COL362 Project 1 - The European Soccer League

Pratyush Pandey, 2017EE10938
Sunil Saharan 2016CS10314

February 24, 2019 - 4:48 am

Contents

1 Project Description

Our project involved making a website which will help a football and data enthusiast visualise European League Football Statistics from the 2006 to 2016. We compiled player performance and match statistics from more than 25,000 Games involving 10,000+ Players which happened in the major European Football Leagues. As you might have guessed, the creators of this project are huge football nuts who wear the jersey of our teams on day of matches and complete assignments early so as to not miss a game, so the primary motivation behind this project was trying to make sense of the data from all the matches we have seen till at least the 2010 season, and attempt to successfully predict things like which team has been on a steady decline and which team is likely to do well in the future.

In an attempt to create a FIFA like system and give the user more ownership of his/her favourite clubs, the user assumes a managerial position and is able to alter the current state of the club by making Player Transfers. He will be able to view the team and player stats, along with the match statistics, to (say) purchase a better defender for his club in order to increase his chances for winning the league cup. The system lets you analyse aggregate data from all the matches to offer insights like the top performing teams and players per season, based on various attributes like attack, defense and overall ratings. If you are a manager of an up and coming team who has a shot at 2020 UEFA Title, the system accommodates you as well. As a manager you have the option of creating new clubs, players, and teams, complete with attributes of all the same, to add to the existing database. You also have the power to change entire match lineups, position players, and decide squad formation for league matches.

The things the system allows you with this data can only be limited by your creativity. Of course, the holy grail is obviously to predict the outcome of the game. The bookies use 3 classes (Home Win, Draw, Away Win) to tip the scales slightly in their favour and get it right about 53% of the time. We attempted to support this number with our own data using SVM but including any ML might have deviated us from the purpose of this project as a database management system and hence the angle was left unexplored. However, we did take the liberty to include some interesting data aggregates. After all, with access to players and teams attributes, team formations and in-game events you should be able to produce some interesting insights into The Beautiful Game. Who knows, Guardiola himself may hire one of you some day!

Table 1: List of Entities and Atrributes

Entity	**Attributes**
Country	id, name
League	id, country_id, name
Manager	id, first name, last name, email, password, team id, country_id
Match	id, country_id, league_id, season, stage, date, match_api_id, home_team_api_id, away_team_api_id, home_team_goal, away_team_goal, home_player_X1, home_player_X2, home_player_X3, home_player_X4, home_player_X5, home_player_X6, home_player_X7, home_player_X8, home_player_X9, home_player_X10, home_player_X11, away_player_X1, away_player_X2, away_player_X3, away_player_X4, away_player_X5, away_player_X6, away_player_X7, away_player_X8, away_player_X9, away_player_X10, away_player_X11, home_player_Y1, home_player_Y2, home_player_Y3, home_player_Y4, home_player_Y5, home_player_Y6, home_player_Y7, home_player_Y8, home_player_Y9, home_player_Y10, home_player_Y11, away_player_Y1, away_player_Y2, away_player_Y3, away_player_Y4, away_player_Y5, away_player_Y6, away_player_Y7, away_player_Y8, away_player_Y9, away_player_Y10, away_player_Y11, home_player_1, home_player_2, home_player_3, home_player_4, home_player_5, home_player_6, home_player_7, home_player_8, home_player_9, home_player_10, home_player_11, away_player_1, away_player_2, away_player_3, away_player_4, away_player_5, away_player_6, away_player_7, away_player_8, away_player_9, away_player_10, away_player_11, goal, shoton, shotoff, foulcommit, card, cross, corner, possession, B365H, B365D, B365A, BWH, BWD, BWA, IWH, IWD, IWA, LBH, LBD, LBA, PSH, PSD, PSA, WHH, WHD, WHA, SJH, SJD, SJA, VCH, VCD, VCA, GBH, GBD, GBA, BSH, BSD, BSA
Player	id, player_api_id, player_name, player_fifa_api_id, birthday, height, weight
Player_Attributes	id, player_fifa_api_id, player_api_id, date, overall_rating, potential, preferred_foot, attacking_work_rate, defensive_work_rate, crossing, finishing, heading_accuracy, short_passing, volleys, dribbling, curve, free_kick_accuracy, long_passing, ball_control, acceleration, sprint_speed, agility, reactions, balance, shot_power, jumping, stamina, strength, long_shots, aggression, interceptions, positioning, vision, penalties, marking, standing_tackle, sliding_tackle, gk_diving, gk_handling, gk_kicking, gk_positioning, gk_reflexes
Team	id, team_api_id, team_fifa_api_id, team_long_name, team_short_name,
Team_Attributes	id, team_fifa_api_id, team_api_id, date, buildUpPlaySpeed, buildUpPlaySpeedClass, buildUpPlayDribbling, buildUpPlayDribblingClass, buildUpPlayPassing, buildUpPlayPassingClass, buildUpPlayPositioningClass, chanceCreationPassing, chanceCreationPassingClass, chanceCreationCrossing, chanceCreationCrossingClass, chanceCreationShooting, chanceCreationShootingClass, chanceCreationPositioningClass, defencePressure, defencePressureClass, defenceAggression, defenceAggressionClass, defenceTeamWidth, defenceTeamWidthClass, defenceDefenderLineClass,

Table 2: Checks for downloaded raw and

Query Number	Average runnning time
Country	11
Country	11
Country	11
Country	11
Country	11

2 Data Sources

What the data offers you -

- +25,000 matches
- +10,000 players
- 11 European Countries with their lead championship
- Seasons 2006 to 2016
- Players and Teams' attributes sourced from EA Sports' FIFA video game series, including the weekly updates
- Team line up with squad formation (X, Y coordinates)
- Betting odds from up to 10 providers. Click here to understand the column naming system for betting odds:
- Detailed match events (goal types, possession, corner, cross, fouls, cards etc...) for +10,000 matches

One can easily find data about soccer matches but they are usually scattered across different websites. We attempted to combine this data from various sources in an attempt to form a coherent database. The data was sourced from:

- `https://www.kaggle.com/hugomathien/soccer` : scores, lineup, team formation and events (available readymade as an sqlite database, needs cleanup to be compatible with postgres)
- `http://www.football-data.co.uk/` : data source for match betting odds. Click here to understand the column naming system for betting odds:
- `http://sofifa.com/` : players and teams attributes from EA Sports FIFA games. Disclaimer - FIFA series and all FIFA assets are property of EA Sports.

When you have a look at the database, you will notice foreign keys for players and matches are the same as the original data sources. We have called those foreign keys "api_id". The data which has been sourced from FIFA official databases is identified with the column name "fifa_api_id".

Improving the DataSet - You will notice that some players are missing from the lineup (NULL values). This is because we have not been able to source their attributes from FIFA. This can be fixed using a more sophisticated crawling algorithm.

Data CleanUp -

- Cleaning HTML
- Strip whitespace
- Converting numbers to number types:
- Converting Boolean values: 'Yes' -¿ True
- Converting dates to machine-readable formats: "24 June 2004" -¿ "2004-06-24"
- Converting an sqlite3 Database to a Postgres Database. This involved -
- Eliminating Duplicates
- Removing unwanted symbol for numeric types like $ or ,

1. Each individual table was extracted as a CSV table. The sqlite default " values for empty rows were replaced by NULL. The sqlite AUTOINCREMENT for primary key ids was changed to SERIAL. The datetime was changed to timestamp, the boolean values for 1 and 0 were made 1::boolean and 0::boolean (respectively).
2. Since we had foreign keys in sqlite database we used set constraints all deferred to avoid insert ordering problems, placing the command inside the BEGIN/COMMIT pair while loading.
3. we also had ‘ on our column names, as generated by some SQLite3 clients, so we needed to remove those to make it postgres compatible.

(1).png

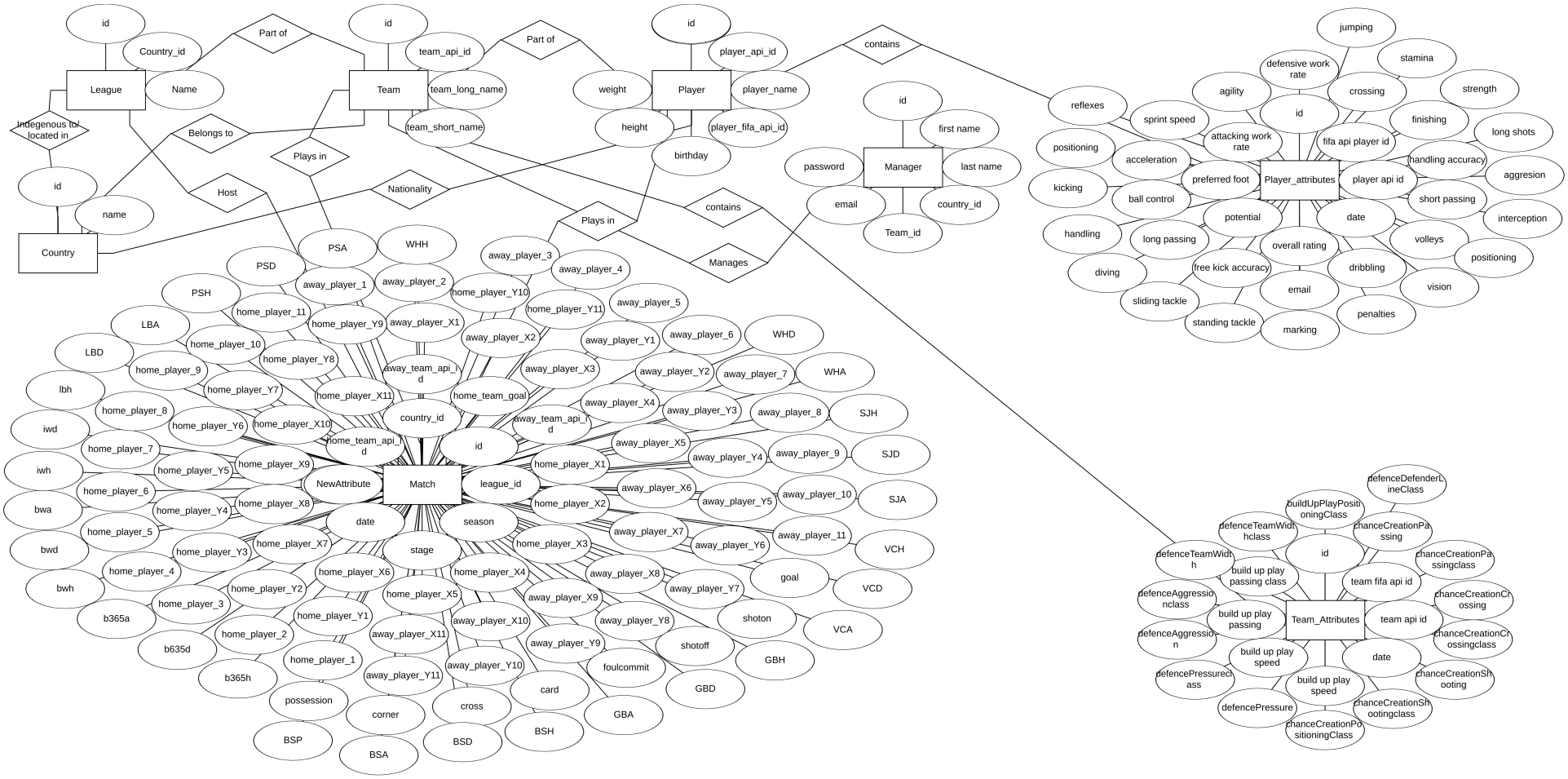

Figure 1: Entity Realtion Diagram of entire Database

Table 3: Data Statistics

Table	No of Tuples	Time to Load	Raw Dataset Size	Raw DataSet Size
Country	11			171 B
League	11			400 B
Match	26,000			297.4 MB
Player	11,100			785.5 KB
Player_Attributes	1,84,000			30.1 MB
Team	299			10.7KB
Team_Attributes	1,458			220.1 KB
Manager				(dynamic table)

3 Functionality and Working

3.1 User's View of the System

1. Home -
   - The User is greeted by an interactive welcome screen listing the system name and its creators. There are seven columns at the top for the user to navigate through different sections of the website. The website offers two functionalities, viewing the site as an unregistered user and as a manager.
   - As the Manager you can make changes in existing team lineups, register new teams and leagues and buy new players from other teams. As a user of the site, you can access match data, team data, player stats and analytics on various team and player attributes (Like top performing teams in a particular season, Top goal scorers in a team, and Number of matches played in a league in a particular season ordered by match averages.
2. Login and Signup Pages -
   It allows for new managers to register on the portal by entering basic information and for old ones to log in.
3. Country and Leagues -
   List down Country and League information that the portal contains. Also offers Season wise Analytics of matches that happened in a particular league.
4. Team and Team Attributes -
   This part of the site allows the user to view various team statistics and analytics on data, supported by visual graphs and charts. Some of the things user can view is -
   - Top Teams in each season, and of all times based on number of wins, losses and draws.
   - Top Teams in each season, and of all times based on number of home wins, losses and draws, as well as away wins, losses and draws.
   - Best teams that a league contains season wise and of all times.
   - Team information and attributes (See the attribute table for more details on which information is available.), which the user can sort based on all attributes and limit his search to say teams of a particular league.
   - Top performing team in a particular country, league and season. (Specific sorting)
5. Match Attributes -
   This part of the site allows the user to view various match statistics and analytics on data, supported by visual graphs and charts. Some of the things user can view is -
   - Detailed Match Information of a particular country ordered by date.
   - Number of matches played in each league by season.
   - Teams ordered by number of home matches played per season and of all times.
   - Teams ordered by number of away matches played per season and of all times.
   - Detailed info about matches per season (for a particular country)

6. Player Attributes -
   This part of the site allows the user to view various player statistics and analytics on data, supported by visual graphs and charts. Some of the things user can view is -
   - Top players in a team of a league ordered by various attributes (or top players of a particular country if the user so desires).
   - group player match attributes by height, weight, age, nationality, or other attributes (Category providing).
   - Player info aggregates grouped by birth year.
   - Number of players born in the same year, which have overall rating greater than a user inputted value.
   - Search accessing various player by searching for various player attributes
   - Birth Distribution - Number of players born in the same year, which are born after 1990, Number of players grouped by birth year, year wise.
7. Special Manager Access only features -
   As a manager, you get special priviledges not offered to a regular user, which include -
   - Create new Teams or take charge of old ones.
   - Create new players for your team.
   - Change team lineups.
   - Update various player and team attributes.
   - Organise new matches and decide win/loss.

3.2 Special Features

1. Indexes:
   Our player attributes table has over 200 thousand rows and 100+ coulmns, so we indexed the table on playerid using a B-Tree index (the PostgreSQL website indicates that hash indexes are not recommended for use) for fast lookups on attributes of a particular player.
2. Constraints:
   Here are the constrints we have placed on various attributes across our tables -
   - NOT NULL constraint
   - FOREIGN KEY constraint
   - UNIQUE constraint
   - ON DELETE CASCADE
   - ON UPDATE CASCADE
   - SET DEFAULT VALUES
   - ADD CHECK CONSTRAINT
   - REFERENTIAL constraint
3. Sequences:
   We have several sequences which keep track of the next manager id, player id, team id and Match id that is to be handed out when a new player is created or when a new manager signs up. Using at the PHP front-end, we extract a new unique value that is next in the sequence. This also ensures that no 2 players are assigned the same ids due to concurrency issues as the sequence lock is first acquired and then the sequence is incremented in this statement.

```
    CREATE SEQUENCE player_player_api_id_seq
owned by player.player_api_id;
SELECT setval('player_player_api_id_seq', 750585, FALSE);
alter table player
alter column player_api_id set default nextval('player_player_api_id_seq');

CREATE SEQUENCE player_player_fifa_api_id_seq
owned by player.player_fifa_api_id;
SELECT setval('player_player_fifa_api_id_seq',  234142, FALSE);
alter table player
alter column player_fifa_api_id set default nextval('player_player_fifa_api_id_seq');
```

4. Triggers:
We have created triggers to achieve things like - Updating the overall rating of the player on updating various player attributes like attacking, defending, passing, etc. and changing the class of a particular skill on updating it to fall within certain brackets. An Example from both these triggers are demonstrated below.

```
    --update class on updating buildUpPlaySpeed

CREATE OR REPLACE FUNCTION upd_buildUpPlaySpeedClass()
RETURNS trigger AS
$$
BEGIN
IF NEW.buildUpPlaySpeed >=67 THEN
NEW.buildUpPlaySpeedClass = 'Fast';
ELSEIF NEW.buildUpPlaySpeed>=34 THEN
NEW.buildUpPlaySpeedClass = 'Balanced';
ELSEIF NEW.buildUpPlaySpeed>=0 THEN
NEW.buildUpPlaySpeedClass = 'Slow';
ELSE
NEW.buildUpPlaySpeed = OLD.buildUpPlaySpeed;
END IF;
RETURN NEW;
END;
$$
LANGUAGE 'plpgsql';

CREATE TRIGGER upd_buildUpPlaySpeed
AFTER UPDATE OF buildUpPlaySpeed
ON Team_Attributes
FOR EACH ROW
EXECUTE PROCEDURE upd_buildUpPlaySpeedClass();

--update overall ratings on updating crossing

CREATE OR REPLACE FUNCTION upd_overallRatings_crossing()
RETURNS trigger AS
$$
BEGIN
NEW.overall_rating = (((NEW.crossing - OLD.crossing)+100)/100)*OLD.overall_rating;
RETURN NEW;
END;
$$
LANGUAGE 'plpgsql';

CREATE TRIGGER upd_player_crossing
AFTER UPDATE OF crossing
ON Player_Attributes
FOR EACH ROW
EXECUTE PROCEDURE upd_overallRatings_crossing();
```

5. Access Privileges:
We implemented 2 user modes - Manager and regular. Regular user can just view data analytics while manager can create new matches, teams, players, lineups and update their attributes.

6. Checks

- All numeric player and team attributes should lie between 0 and 100
- All attribute classes in team attribute can usually take one of two or three values so various checks were put in place to deal with that. For eg - Preferred foot can either be left or right,
- Season should belong in ('2012/2013', '2010/2011', '2013/2014' , '2015/2016', '2008/2009' , '2014/2015' ,'2011/2012', '2009/2010')
- Stage should lie between 1 and 40.

7. Defaults
    - The weights default to 70 kg when creating a new player. The height defaults to 170 cm.
    - All player and team averages when creating a new player or team default to dynamically calculated aggregate averages.
    - Player positioning, betting informations default to 0.

4 List of Queries Used

1. League Queries :

    a. Leagues by Season Analytics

```
            SELECT Country.name AS country_name,
        League.name AS league_name,
        season,
        count(distinct stage) AS number_of_stages,
        count(distinct HT.team_long_name) AS number_of_teams,
        avg(home_team_goal) AS avg_home_team_scors,
        avg(away_team_goal) AS avg_away_team_goals,
        avg(home_team_goal-away_team_goal) AS avg_goal_dif,
        avg(home_team_goal+away_team_goal) AS avg_goals,
        sum(home_team_goal+away_team_goal) AS total_goals
FROM Match
JOIN Country on Country.id = Match.country_id
JOIN League on League.id = Match.league_id
LEFT JOIN Team AS HT on HT.team_api_id = Match.home_team_api_id
LEFT JOIN Team AS AT on AT.team_api_id = Match.away_team_api_id
WHERE country.name in ('Spain', 'Germany', 'France', 'Italy', 'England')
GROUP BY Country.name, League.name, season
HAVING count(distinct stage) > 10
ORDER BY Country.name, League.name, season DESC
;
```

    b. Participating League and their Countries

```
SELECT Country.id, League.name AS League_name, Country.name AS Country_name
FROM League, Country
WHERE Country.id=League.country_id;
```

2. Match Queries

    a. Detailed Match Information of a particular country ordered by date

```
            SELECT  Match.id,
        Country.name AS country_name,
        League.name AS league_name,
        season,
        stage,
        date,
        HT.team_long_name AS  home_team,
        AT.team_long_name AS away_team,
        home_team_goal,
        away_team_goal
FROM Match
JOIN Country on Country.id = Match.country_id
JOIN League on League.id = Match.league_id
LEFT JOIN Team AS HT on HT.team_api_id = Match.home_team_api_id
LEFT JOIN Team AS AT on AT.team_api_id = Match.away_team_api_id
WHERE country.name = 'Spain'
ORDER by date
```

```
LIMIT 10;
```

b. Detailed info about matches per season (for a particular country)

```
            SELECT Match.id,
        Country.name AS country_name,
        League.name AS league_name,
        season,
        stage,
        date,
        HT.team_long_name AS home_team,
        AT.team_long_name AS away_team,
        CASE WHEN home_team_goal > away_team_goal then HT.team_long_name
            ELSE AT.team_long_name
            END AS Winning_team,
        home_team_goal,
        away_team_goal
FROM Match
JOIN Country on Country.id = Match.country_id
JOIN League on League.id = Match.league_id
LEFT JOIN Team AS HT on HT.team_api_id = Match.home_team_api_id
LEFT JOIN Team AS AT on AT.team_api_id = Match.away_team_api_id
WHERE country.name = 'Spain'
--WHERE country.name in ('Spain', 'Germany', 'France', 'Italy', 'England')
ORDER by date
LIMIT 10
;
```

c. MATCHES PLAYED IN EACH LEAGUE BY SEASON

```
            SELECT
        Country.name AS country_name,
        League.name AS league_name,
        season,
        COUNT(distinct Match.id) AS Matches_Played
FROM Match
JOIN Country on Country.id = Match.country_id
JOIN League on League.id = Match.league_id
LEFT JOIN Team AS HT on HT.team_api_id = Match.home_team_api_id
LEFT JOIN Team AS AT on AT.team_api_id = Match.away_team_api_id
--WHERE country.name = 'Spain'
WHERE country.name in ('Spain', 'Germany', 'France', 'Italy', 'England')
GROUP BY Country.name, League.name, season
ORDER by Country.name, League.name, season
LIMIT 20
;
```

d. Teams ordered by number of home matches played (Season)

```
            SELECT
        Country.name AS country_name,
        League.name AS league_name,
        HT.team_long_name AS home_team,
        season,
        COUNT(distinct Match.id) AS Matches_Played
FROM Match
JOIN Country on Country.id = Match.country_id
JOIN League on League.id = Match.league_id
LEFT JOIN Team AS HT on HT.team_api_id = Match.home_team_api_id
LEFT JOIN Team AS AT on AT.team_api_id = Match.away_team_api_id
--WHERE country.name = 'Spain'
WHERE country.name in ('Spain', 'Germany', 'France', 'Italy', 'England')
```

```
GROUP BY HT.team_long_name, Country.name, League.name, season
ORDER by Matches_Played DESC, Country.name, League.name, season, HT.team_long_name
LIMIT 20
;
```

3. Team Queries -

   a. Top Winning Teams per season

```
        SELECT
    Country.name AS country_name,
    League.name AS league_name,
    season,
    CASE WHEN home_team_goal > away_team_goal then HT.team_long_name
        ELSE AT.team_long_name
        END AS Winning_team,
    COUNT(*) as num_wins
FROM Match
JOIN Country on Country.id = Match.country_id
JOIN League on League.id = Match.league_id
LEFT JOIN Team AS HT on HT.team_api_id = Match.home_team_api_id
LEFT JOIN Team AS AT on AT.team_api_id = Match.away_team_api_id
WHERE country.name = 'Spain'
--WHERE country.name in ('Spain', 'Germany', 'France', 'Italy', 'England')
GROUP BY Winning_team, Country.name, League.name, season
ORDER BY num_wins DESC, Country.name, League.name, season DESC
LIMIT 20
;
```

   b. best teams in a league season wise

```
        SELECT Country.name AS country_name,
    League.name AS league_name,
    HT.team_long_name,
    season,
    count(distinct stage) AS number_of_stages,
    -- count(distinct HT.team_long_name) AS number_of_teams,
    avg(home_team_goal) AS avg_home_team_goals,
    avg(away_team_goal) AS avg_away_team_goals,
    avg(home_team_goal-away_team_goal) AS avg_goal_dif,
    avg(home_team_goal+away_team_goal) AS avg_goals,
    sum(home_team_goal+away_team_goal) AS total_goals
FROM Match
JOIN Country on Country.id = Match.country_id
JOIN League on League.id = Match.league_id
LEFT JOIN Team AS HT on HT.team_api_id = Match.home_team_api_id
LEFT JOIN Team AS AT on AT.team_api_id = Match.away_team_api_id
WHERE country.name in ('Spain', 'Germany', 'France', 'Italy', 'England')
--WHERE league.name in () --best teams in a given league
--WHERE season in () --best teams in a given season
GROUP BY HT.team_long_name, Country.name, League.name, season
--HAVING count(distinct stage) > 10
ORDER BY total_goals DESC, Country.name, League.name, season DESC
LIMIT 10
;
```

   c. top teams by home goals (Season Wise)

```
        SELECT
    Country.name AS country_name,
    League.name AS league_name,
    season,
```

```
        HT.team_long_name AS home_team,
        SUM(home_team_goal) AS "home_goals_total"
FROM Match
JOIN Country on Country.id = Match.country_id
JOIN League on League.id = Match.league_id
LEFT JOIN Team AS HT on HT.team_api_id = Match.home_team_api_id
LEFT JOIN Team AS AT on AT.team_api_id = Match.away_team_api_id
WHERE country.name = 'Spain'
--WHERE country.name in ('Spain', 'Germany', 'France', 'Italy', 'England')
GROUP BY HT.team_long_name, Country.name, League.name, season
ORDER BY home_goals_total DESC, Country.name, League.name, season DESC
LIMIT 20
;
```

d. Team Information

```
            SELECT * FROM Team, Team_Attributes
WHERE Team.team_api_id = Team_Attributes.team_api_id
ORDER BY Team.team_long_name;
```

4. Player Queries

a. Top Players based on various attributes

```
            -- Top Players by overall_rating

SELECT player_name, AVG(overall_rating) AS ratings
FROM Player, Player_Attributes
WHERE Player.player_api_id = Player_Attributes.player_api_id AND overall_rating > 0
GROUP BY 1
ORDER BY 2 DESC
LIMIT 10
;

--Top Players by potential

SELECT player_name, AVG(potential) AS potential_avg
FROM Player, Player_Attributes
WHERE Player.player_api_id = Player_Attributes.player_api_id AND potential > 0
GROUP BY 1
ORDER BY 2 DESC
LIMIT 10
;

--Top Players by crossing

SELECT player_name, AVG(crossing) AS crossing_avg
FROM Player, Player_Attributes
WHERE Player.player_api_id = Player_Attributes.player_api_id AND crossing > 0
GROUP BY 1
ORDER BY 2 DESC
LIMIT 10
;

--Top Players by finishing

SELECT player_name, AVG(finishing) AS finishing_avg
FROM Player, Player_Attributes
WHERE Player.player_api_id = Player_Attributes.player_api_id AND finishing > 0
GROUP BY 1
ORDER BY 2 DESC
LIMIT 10
```

```
;

--Top Players by heading_accuracy

SELECT player_name, AVG(heading_accuracy) AS heading_accuracy_avg
FROM Player, Player_Attributes
WHERE Player.player_api_id = Player_Attributes.player_api_id AND heading_accuracy > 0
GROUP BY 1
ORDER BY 2 DESC
LIMIT 10
;

-- AND SO ON
```

b. Search accessing player attributing

```
            SELECT
p.player_name,
--*,
EXTRACT(YEAR FROM to_timestamp(p.birthday, 'YYYY-MM-DD hh24:mi:ss'))::int AS "year_born",
-- 2019-(CAST(coalesce(year_born, '0') AS integer)) AS "player_age"
2019::int-(select EXTRACT(YEAR FROM to_timestamp(p.birthday, 'YYYY-MM-DD hh24:mi:ss'))::int) AS
pa.overall_rating,pa.potential,pa.preferred_foot,pa.attacking_work_rate,pa.defensive_work_rate,
FROM Player AS p
INNER JOIN Player_Attributes AS pa
ON p.player_api_id = pa.player_api_id
--WHERE (Insert Sorting Attribute here > Insert threshhold value)
ORDER BY p.player_name
LIMIT 10
;
```

c. Number of players born in the same year, which have overall$_{r}atinggreaterthan80$

```
SELECT
COUNT(p.player_name) AS number_of_players,
EXTRACT(YEAR FROM to_timestamp(p.birthday, 'YYYY-MM-DD hh24:mi:ss'))::int AS "year_born"
FROM Player AS p
INNER JOIN Player_Attributes AS pa
ON p.player_api_id = pa.player_api_id
WHERE pa.overall_rating > 90
GROUP BY year_born;
```

d. Player info aggregates grouped by birth year

```
            SELECT
COUNT(p.player_name) AS number_of_players,
EXTRACT(YEAR FROM to_timestamp(p.birthday, 'YYYY-MM-DD hh24:mi:ss'))::int AS "year_born",
MIN(pa.overall_rating) AS min_overall_rating,
MAX(pa.overall_rating) AS max_overall_rating,
AVG(pa.overall_rating) AS average_overall_rating
FROM Player AS p
INNER JOIN Player_Attributes AS pa
ON p.player_api_id = pa.player_api_id
GROUP BY year_born;
```

e. group players by height (Category providing)

```
            SELECT player_name, height,
    CASE
```

```
        WHEN height < 170.00 THEN 'Short'
        WHEN height BETWEEN 170.00 AND 185.00 THEN 'Medium'
        WHEN height > 185.00 THEN 'Tall'
    END AS height_class
FROM Player;
```

f. country-league-season level analytics

```
            SELECT Country.name AS country_name,
        League.name AS league_name,
        season,
        count(distinct stage) AS number_of_stages,
        count(distinct HT.team_long_name) AS number_of_teams,
        avg(home_team_goal) AS avg_home_team_goals,
        avg(away_team_goal) AS avg_away_team_goals,
        avg(home_team_goal-away_team_goal) AS avg_goal_dif,
        avg(home_team_goal+away_team_goal) AS avg_goals,
        sum(home_team_goal+away_team_goal) AS total_goals
FROM Match
JOIN Country on Country.id = Match.country_id
JOIN League on League.id = Match.league_id
LEFT JOIN Team AS HT on HT.team_api_id = Match.home_team_api_id
LEFT JOIN Team AS AT on AT.team_api_id = Match.away_team_api_id
WHERE country.name in ('Spain', 'Germany', 'France', 'Italy', 'England')
GROUP BY Country.name, League.name, season
HAVING count(distinct stage) > 10
ORDER BY Country.name, League.name, season DESC
;
```

g. Number of players born in the same year, which are born after 1990

```
            SELECT
COUNT(p.player_name) AS number_of_players,
EXTRACT(YEAR FROM to_timestamp(p.birthday, 'YYYY-MM-DD hh24:mi:ss'))::int AS "year_born"
FROM Player AS p
INNER JOIN Player_Attributes AS pa
ON p.player_api_id = pa.player_api_id
GROUP BY year_born
HAVING EXTRACT(YEAR FROM to_timestamp(p.birthday, 'YYYY-MM-DD hh24:mi:ss'))::int > '1990';
```

h. Player Information by Height

```
            SELECT CASE
    WHEN ROUND(height)<165 then 165
    WHEN ROUND(height)>195 then 195
    ELSE ROUND(height)
    END AS calc_height,
    COUNT(height) AS distribution,
    (avg(PA_Grouped.avg_overall_rating)) AS avg_overall_rating,
    (avg(PA_Grouped.avg_potential)) AS avg_potential,
    AVG(weight) AS avg_weight
FROM PLAYER
LEFT JOIN (SELECT Player_Attributes.player_api_id,
    avg(Player_Attributes.overall_rating) AS avg_overall_rating,
    avg(Player_Attributes.potential) AS avg_potential
    FROM Player_Attributes
    GROUP BY Player_Attributes.player_api_id)
    AS PA_Grouped ON PLAYER.player_api_id = PA_Grouped.player_api_id
GROUP BY calc_height
ORDER BY calc_height
;
```

i. Number of players grouped by birth year year wise

```
            SELECT COUNT(p.player_name) AS number_of_players,
--strftime('%Y',p.birthday) AS "year_born"
EXTRACT(YEAR FROM to_timestamp(p.birthday, 'YYYY-MM-DD hh24:mi:ss')) AS "year_born"
FROM Player AS p
INNER JOIN Player_Attributes AS pa
ON p.player_api_id = pa.player_api_id
GROUP BY year_born;
```

j. Query the entire Players and attributes table

```
            SELECT *
FROM Player,Player_Attributes WHERE Player.Player_api_id=Player_Attributes.player_api_id;
```

4.1 Query Runtimes

Following is a list of query runtimes times (averaged over 3 runs) of queries from the previous Section.

Table 4: Data Statistics

Query Number	Average running time
1.a	47.086 ms
1.b	0.741 ms
2.a	17.274 ms
2.b	12.520 ms
2.c	12.986 ms
2.d	45.617 ms
3.a	18.472 ms
3.b	58.160 ms
3.c	15.815 ms
3.d	43.617 ms
4.a	70.472 ms
4.b	314.160 ms
4.c	15.815 ms
4.d	73.282 ms
4.e	8.654 ms
4.f	45.458 ms
4.g	60.232 ms
4.h	58.282 ms
4.g	430.974 ms