PES UNIVERSITY

100 feet Ring Road, BSK 3rd Stage, Bengaluru 560085

Department of Computer Science and Engineering

Jan – May 2020

UE18CS252
Database Management Systems

Project Report

The Beautiful Game

PES1201800110 Tejas Srinivasan
4th Sem Sec. [A] Roll No. 05

PROJECT SUMMARY

"The Beautiful Game" as the title aptly suggests is an effort to create a database implementation of the truly beautiful game of football via the representation of various football leagues around the world along with their entities and the relationship between them. It seeks to provide pertinent and useful information about these like the contract information of the players and managers, important statistics of players (like the number of goals,assists,minutes played and even the number of yellow and red cards) that are used in the actual transfer markets of the footballing world to make important transfer decisions and information about the football matches themselves along with the stadiums that they take place in and the referees that officiate the proceedings. Information about the teams such as their owners,number of trophies won have also been listed. The necessary aforementioned entities and their relationships are modeled with the help of an ER diagram and the relationships were enhanced using key constraints by the plotting of a relational schema. The system also supported the creation of tables and the insertion of values into them through DDL statements and the reasons for the choice of the appropriate keys and the data types are listed. Since the tables were obtained with the help of the relational schema and other efforts were taken to ensure that there is no data loss from joins of the tables, the tables obtained are already in the highest normal form (3NF) and also satisfy the lossless join property. Moreover, a trigger is defined to enforce an important semantic constraint by auditing the INSERT trial and ensures that the details about the matches played with respect to the leagues that they are a part of are accurate. Informative retrieval queries have been created along with a simple description of what they are trying to achieve and images of the data output have been added to show their impact. Finally some important points regarding the capabilities of the current system like its efficient modeling of the relationship between the entities in football and useful query information have been listed. A limitation of the current system has also been listed along with suggestions for future improvements like the addition of security measures, a GUI amongst other things. In conclusion, the database system created performs the clearly defined tasks and provides useful and accurate information about the relationship between the entities of a beautiful footballing universe.

Introduction

The mini world chosen to model the database is a database of football leagues around the world and the entities that make them up (players, managers,teams etc.).The entities present in this mini world and their transactions have been briefly described below -

- Players - They are the professional sports persons that are contracted by teams to play the matches in various leagues.

- Teams - These are sports organizations that compete against other teams from the same league for the league trophy. Each of these teams have multiple players as well as one manager to oversee the team. They have owners who sponsor the stadium in which they play the matches.

- Managers - A manager is an official that is appointed in charge of the players of the team. Each team has their own manager on a contract, and his role to decide the tactics of the team and orchestrate how the team plays in the games.

- Referees - A referee is an official appointed by a league to oversee the various matches played. Based on their experience, they are classified into various categories and play different roles.

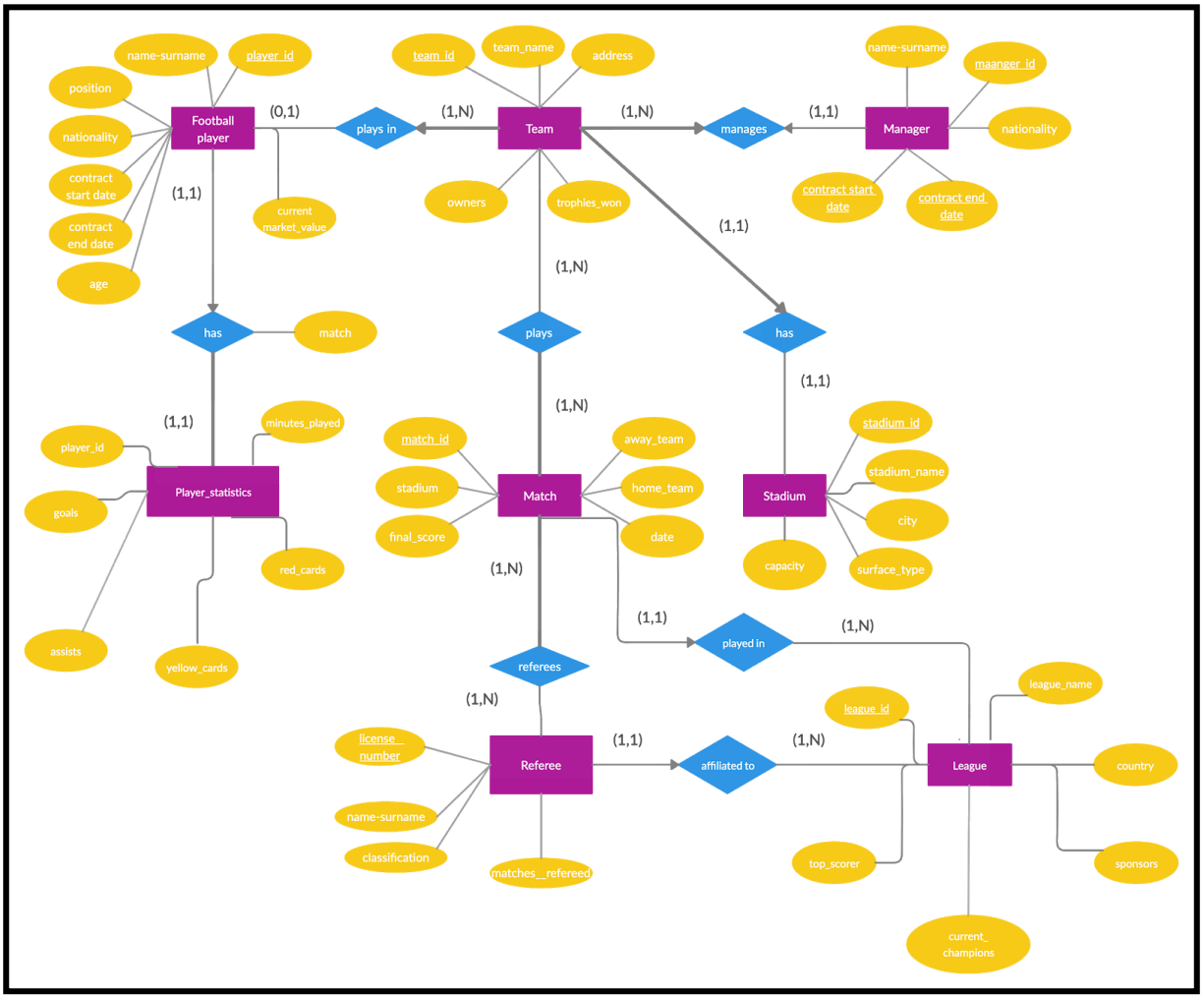
- Matches - A match is played between the players who are representing two teams and usually ends up in a win,loss or a draw. Each of these games are overseen by a referee.

- Stadium - A stadium is the location where the matches are held. Each team has their own stadium and matches are held either in the home stadium of a team or away. Each stadium has a different surface type and capacity.

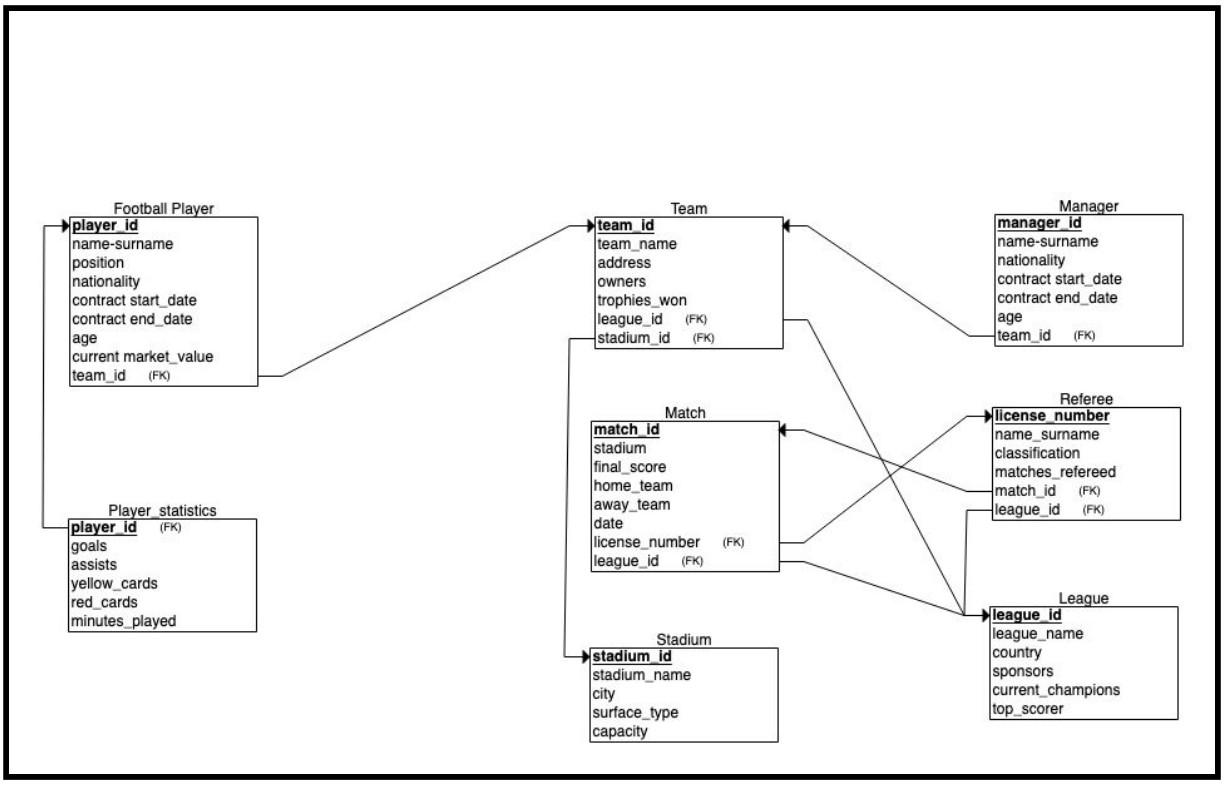
- League - A league is the official yearly competition in football, in which multiple teams compete. Each country has their own major league and several sub-leagues and minor leagues played in a hierarchical format. Each of these leagues have a champion every year and a top scorer amongst other statistics and measures maintained.

Data Model

ER Diagram -



Relational Schema -



The above schema has been generated from the ER diagram upon closely examining the relationships between each entity. The primary key is an uniquely assigned identifier for each of the entities, and is generated in the form of an integer type-attribute to maintain consistency amongst the various tables. For most of the tables identification of the primary keys was just an examination of the determinant of the functional dependencies (which have been discussed in a later section) and the foreign keys were included based on the dependency of an attribute of a table on another attribute from a different table. However, in the Team ,League,Manager and Match tables there were other candidate keys that were taken into consideration as well, but finally only specific keys (one from each table) were chosen as the primary key for the tables.

The reasons for the choice of the above mentioned keys in the relational schema have been listed below -

- The Team relation has two candidate keys, that is the team_id and the combination of {team_name and the foreign key league_id}, since no two teams in the same league can have the same name. However, in this case the team_id is chosen as the primary key since it is a singular value and does not depend on other relations (league_id is a foreign key referenced from the league table).

- The League relation has two candidate keys, that is the league_id and the composite value of {league_name,country}. The league id is chosen as it is better to have a singular value as the primary key instead of a composite value.
- The Manager relation actually has two candidate keys both of which are singular values, that are the manager_id and the team_id. But since team_id is the foreign key referenced in the 1:1 relationship between the team and the manager, it is discarded as a choice and the manager_id is chosen as the primary key instead.
- The Match relation can be identified using either the match_id attribute or a combination of the home and away teams (given by their ids or their names) along with the date of the match. However since the match_id is only a singular value that needs to be used to reference other attributes, it is chosen as the primary key.

The data types chosen are -

- **INT** - The integer type is chosen for the *primary key attributes* of the tables like the various id numbers of players, managers etc. and the license number of the referees.
  It was also chosen for -
  1. Various player statistics like minutes played, number of goals,assists, red and yellow cards received etc.
  2. Number of trophies that a certain team has won in the league.

- **VARCHAR(n)** - This type was chosen for attributes like -
  1. Name and surname of the players, managers and others.
  2. Team and Stadium names and similar attributes like country of the league and nationality of the players and managers.
  3. League attributes like top_scorer, current_champions and sponsors.

- **DATE** - The date type is chosen for attributes like -
  1. Starting and ending dates of contracts of players and managers.
  2. Date of the matches played

FD and Normalization

Functional Dependencies (FDs) -

The functional dependencies for each of the tables are given below -

Football_Player
player_id -> name-surname, position, nationality, contract_start_date, contract_end_date, age, current market_value, team_id

Player_Stats
player_id -> goals,assists,yellow_cards,red_cards,minutes_played

Team
team_id -> team_name, address, trophies_won, league_id, stadium_id
team_name, league_id -> tean_id, address, trophies_won, stadium_id

Match
match_id -> stadium, final_score, home_team, away_team, date, license_number, league_id
home_team, away_team, date -> final_score,license_number, league_id

Manager
manager_id -> name-surname, nationality, contract_start_date, contract_end_date, age, team_id
team_id -> name-surname, nationality, contract_start_date, contract_end_date, age, manager_id

Referee
license_number -> name-surname, classification, matches_refereed, match_id, league_id

Stadium
stadium_id -> stadium_name, city, surface_type, capacity

League
league_id -> league_name, country, sponsors, current_champions, top_scorer

Normalization and Lossless Join Property -

All of the relations obtained were from the ER diagram and subsequently the relational schema generated from it and hence were already in the 3rd Normal Form (3NF). However, if the structure of the schema were changed, then there is a chance that there could be a violation. Such a violation that can occur is if the columns for the home and away teams (home_team and away_team) from the Match table were included as a part of the Team table, then the 3rd normal form (3NF) would be violated as the team names would be functionally dependent on the team_id even

though it is not a primary key of the table. If we had IDs for each of these home and away teams and considered their composite key along with the composite key of ‘Date’ then it would mean that adding team names along with them would result in a violation of the 2ndnormal form (2NF) since they would be functionally dependent on the ID which is a part of the primary key and is not allowed under the second normal form.

The relations chosen were based on the ER diagram and hence it can be seen that the decomposition of the tables would be lossless, and would therefore satisfy the lossless join property. However there is a possibility of the existence of spurious tuples since the name-surname attributes of both the Player and the Manager table are the same and a natural join of the two tables would lead to them being joined by an arbitrary non-distinct attribute. This problem can be overcome by clearly specifying the distinction as player_name-surname and manager_name-surname for these attributes and since the joins would only occur with primary keys and foreign keys, this will ensure that the lossless join property is also satisfied.

DDL

Table creation queries along with the definitions of the entity integrity and the referential integrity constraints -

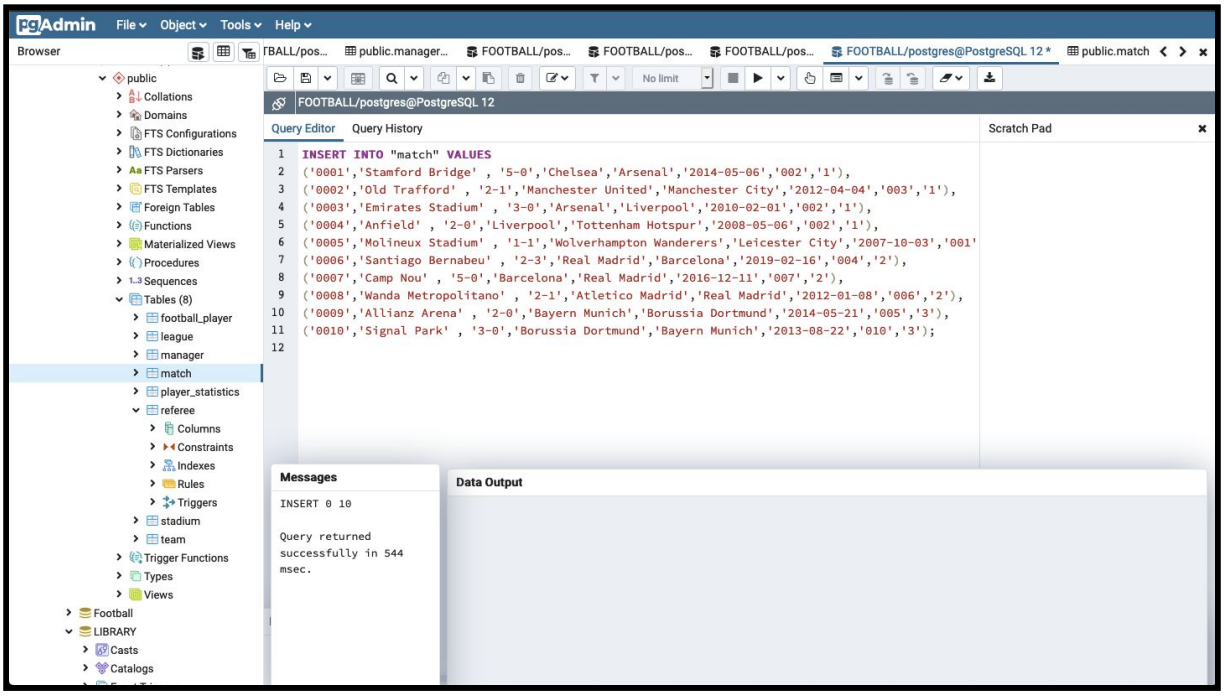
```
CREATE TABLE League
(
  league_id INT,
  league_name VARCHAR,
  country VARCHAR,
  sponsors VARCHAR,
  current_champions VARCHAR,
  top_scorer VARCHAR,
  PRIMARY KEY (league_id)
);

CREATE TABLE Stadium
(
  stadium_id INT,
  stadium_name VARCHAR,
  city VARCHAR,
  surface_type VARCHAR,
  capacity INT,
  PRIMARY KEY (stadium_id)
```

```
);

CREATE TABLE Team
(
  team_id INT,
  team_name VARCHAR,
  address VARCHAR,
  owners VARCHAR,
  trophies_won INT,
  league_id INT,
  stadium_id INT,
  PRIMARY KEY (team_id),
  FOREIGN KEY (league_id) REFERENCES League(league_id),
  FOREIGN KEY (stadium_id) REFERENCES Stadium(stadium_id)
);

CREATE TABLE Manager
(
  manager_id INT,
  manager_name_surname VARCHAR,
  nationality VARCHAR,
  manager_contract_start_date DATE,
  manager_contract_end_date DATE,
  age INT,
  team_id INT,
  PRIMARY KEY (manager_id),
  CHECK(manager_contract_end_date>manager_contract_start_date),
  FOREIGN KEY (team_id) REFERENCES Team(team_id)
);

CREATE TABLE Football_Player
(
  player_id INT,
  player_name_surname VARCHAR,
  position VARCHAR,
  nationality VARCHAR,
  player_contract_start_date DATE,
  player_contract_end_date DATE,
  age INT,
  current_market_value FLOAT,
  team_id INT,
  PRIMARY KEY (player_id),
  CHECK(player_contract_end_date>player_contract_start_date),
```

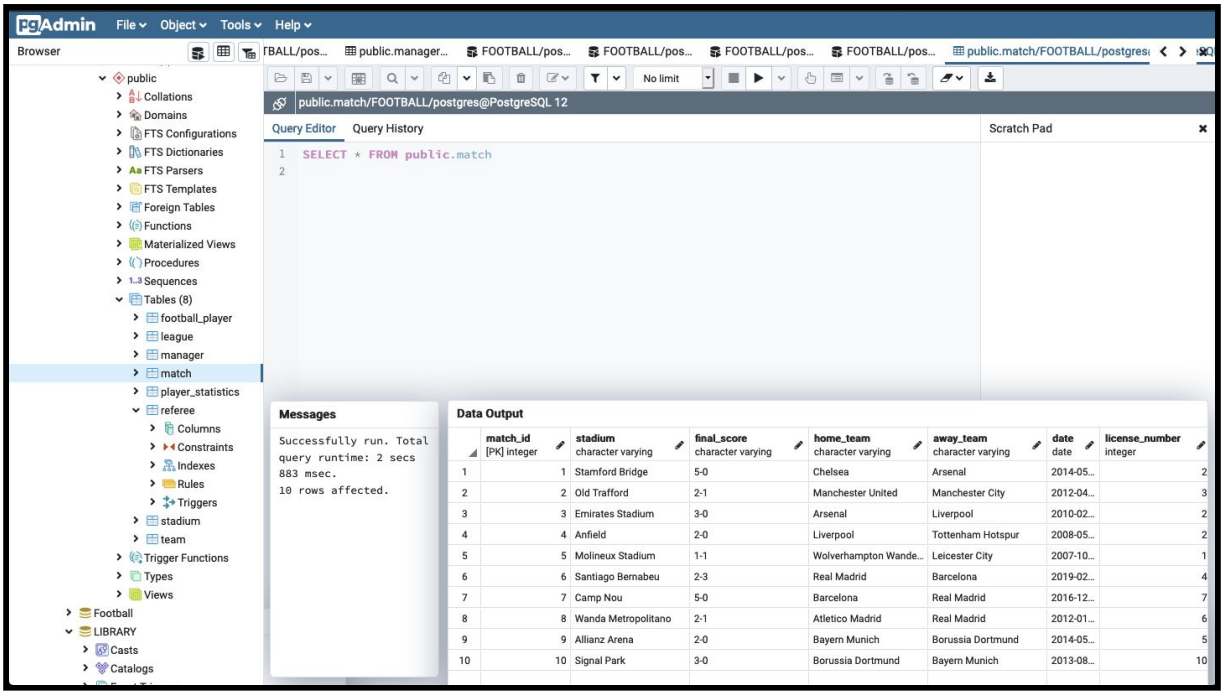
```
  FOREIGN KEY (team_id) REFERENCES Team(team_id)
);

CREATE TABLE Player_statistics
(
  goals INT,
  assists INT,
  yellow_cards INT,
  red_cards INT,
  minutes_played INT,
  player_id INT,
  PRIMARY KEY (player_id),
  FOREIGN KEY (player_id) REFERENCES Football_Player(player_id)
);

CREATE TABLE Referee
(
  license_number INT,
  referee_name_surname VARCHAR,
  classification VARCHAR,
  matches_refereed INT,
  league_id INT,
  PRIMARY KEY (license_number),
  FOREIGN KEY (league_id) REFERENCES League(league_id)
);

CREATE TABLE Match
(
  match_id INT,
  stadium VARCHAR,
  final_score VARCHAR,
  home_team VARCHAR,
  away_team VARCHAR,
  date DATE,
  license_number INT,
  league_id INT,
  PRIMARY KEY (match_id),
  FOREIGN KEY (license_number) REFERENCES Referee(license_number),
  FOREIGN KEY (league_id) REFERENCES League(league_id)
);
```

Sample INSERT queries along with their outputs -

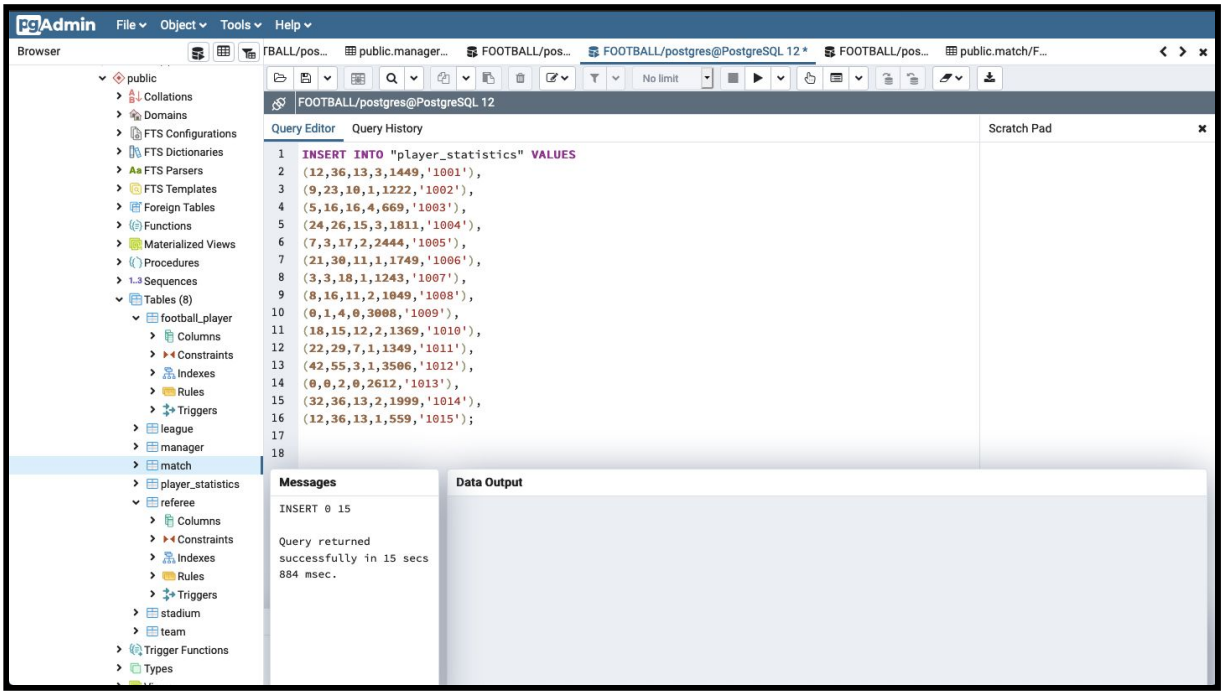
1. INSERT INTO “match” VALUES



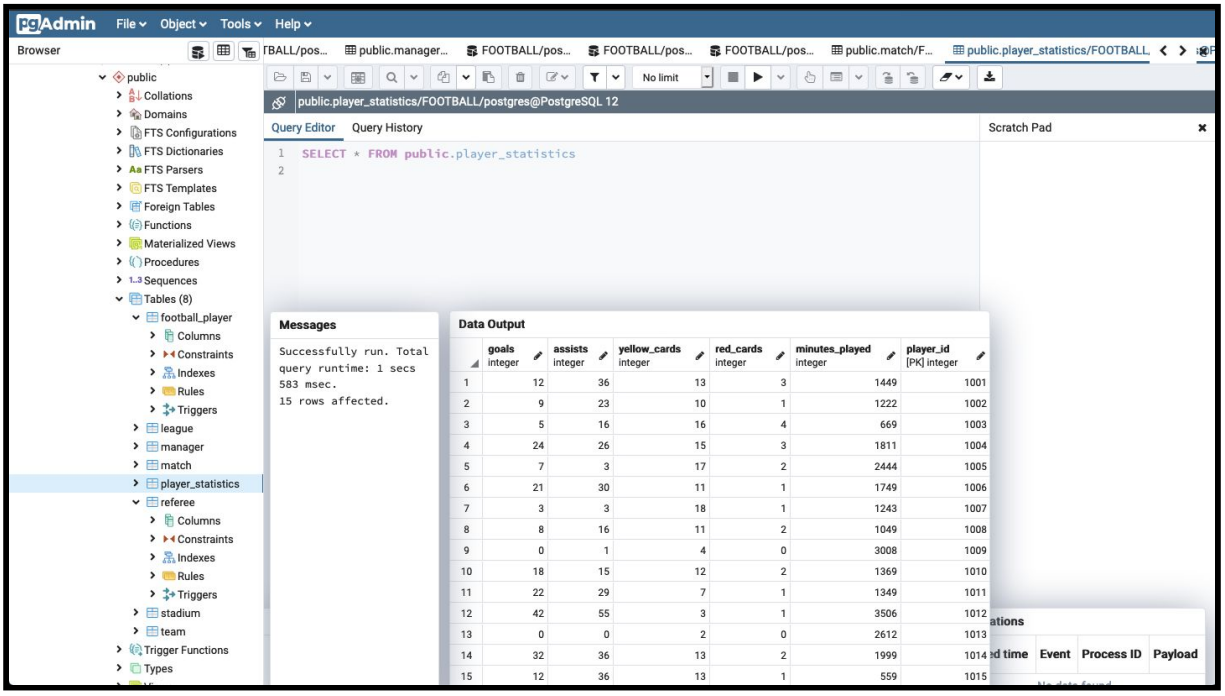
Output -



2. INSERT INTO “player_” VALUES



Output -

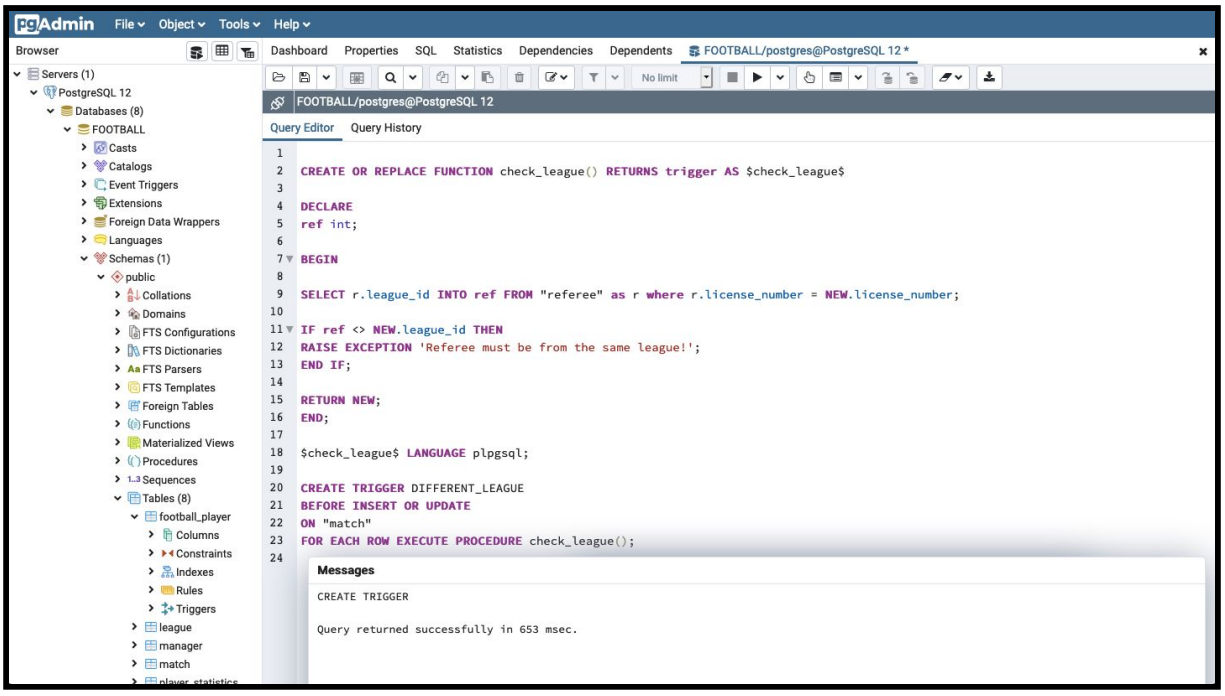


Triggers

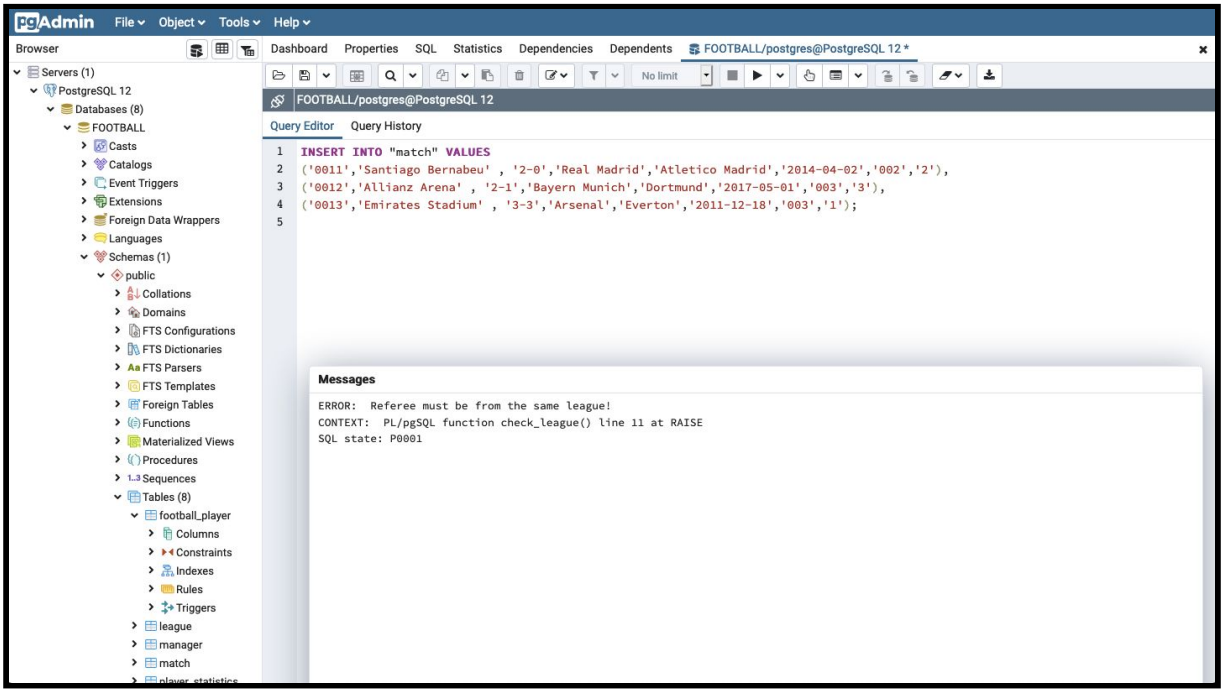
A trigger is a stored procedure in a database which is automatically invoked whenever a special event in the database occurs.

For the Match table, the referee must belong to the same league as the match that is being overseen by him/her. To ensure that this semantic constraint is upheld, the following trigger is used -

```
CREATE OR REPLACE FUNCTION check_league() RETURNS trigger AS
$check_league$

DECLARE
ref int;

BEGIN

SELECT r.league_id INTO ref FROM "referee" as r where r.license_number =
NEW.license_number;

IF ref <> NEW.league_id THEN
RAISE EXCEPTION 'Referee must be from the same league!';
END IF;

RETURN NEW;
END;

$check_league$ LANGUAGE plpgsql;

CREATE TRIGGER DIFFERENT_LEAGUE
BEFORE INSERT OR UPDATE
ON "match"
FOR EACH ROW EXECUTE PROCEDURE check_league();
```



Case where the trigger is violated -



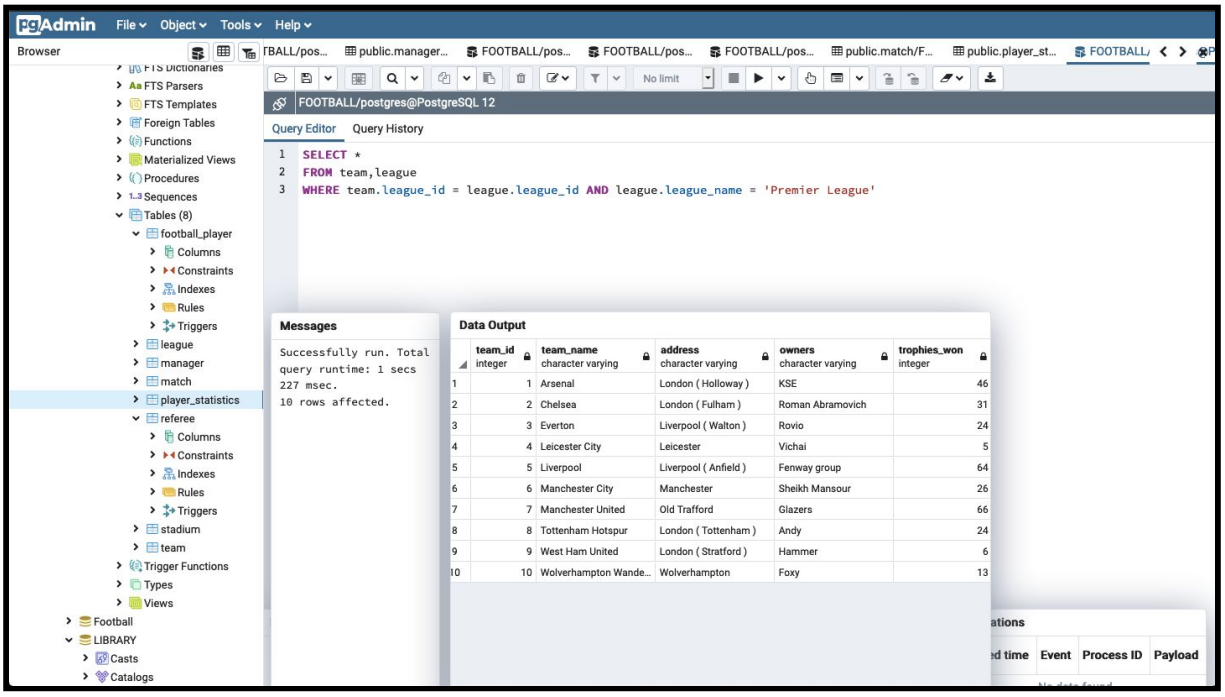
As we can see, an attempt to insert values to a match of a certain league with a referee from another league throws an error. The violation can be resolved by ensuring that the referee is from the same league as that of the match being played through his/her license_number. This can also be extended to ensure that the home and away teams playing the match belong to the same league via the same process of creating additional triggers. Other semantic constraints can also be identified and enforced in a similar manner.

SQL Queries

1. Find the total number of teams playing in the Premier league and their details.

```
SELECT *
FROM team,league
WHERE team.league_id = league.league_id AND league.league_name = 'Premier League'
```

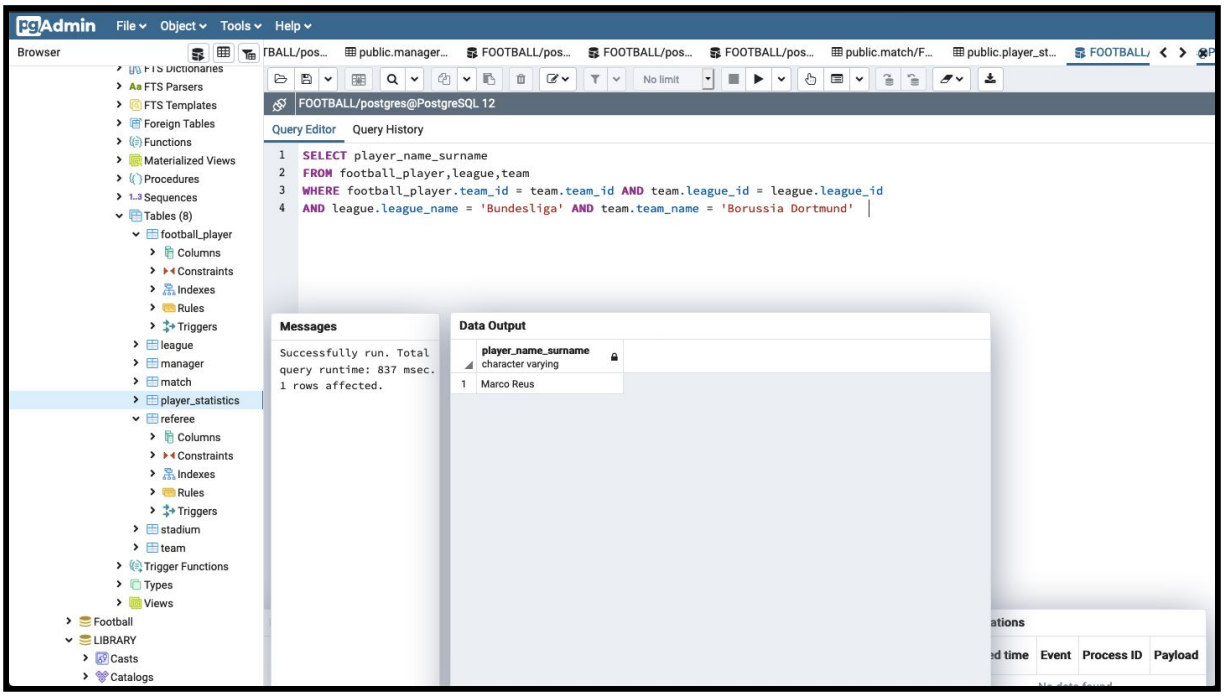
Output -



2. Find the names of the players playing in the Bundesliga that play for Borussia Dortmund.

```
SELECT player_name_surname
FROM football_player,league,team
WHERE football_player.team_id = team.team_id AND team.league_id = league.league_id
AND league.league_name = 'Bundesliga' AND team.team_name = 'Borussia Dortmund'
```

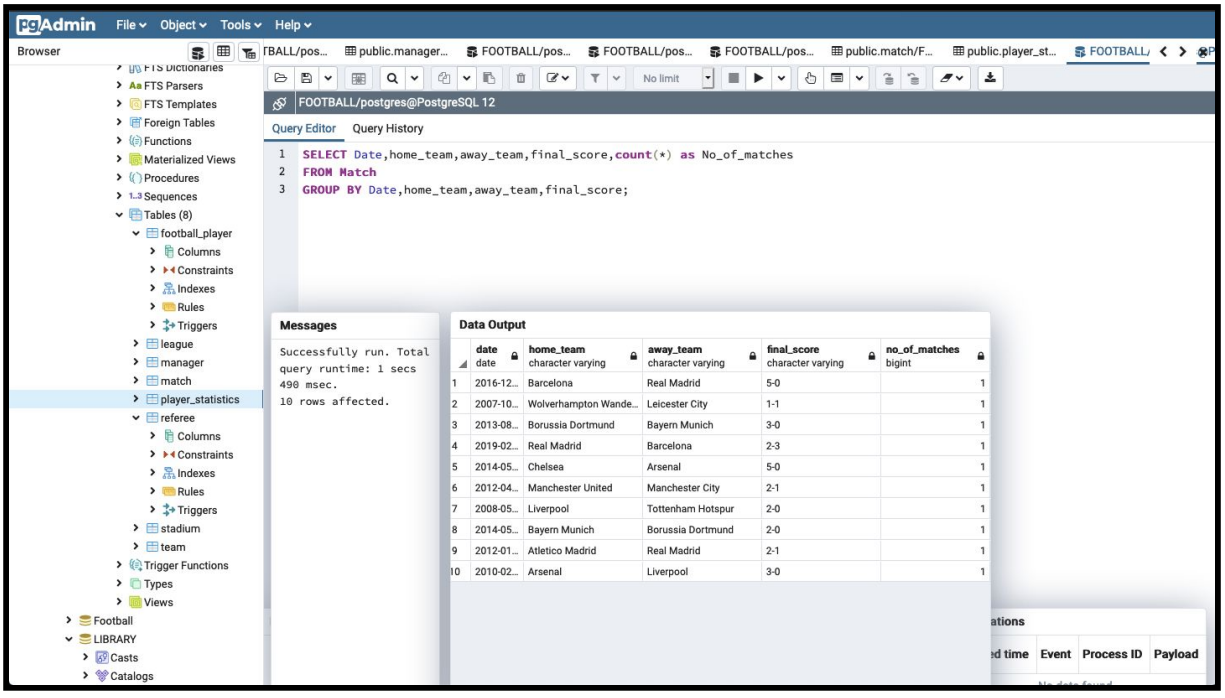
Output -



3. Find the number of matches played on each date, and return details like the teams that played, the final score of the game etc.

```
SELECT Date,home_team,away_team,final_score,count(*) as No_of_matches
FROM Match
GROUP BY Date,home_team,away_team,final_score;
```

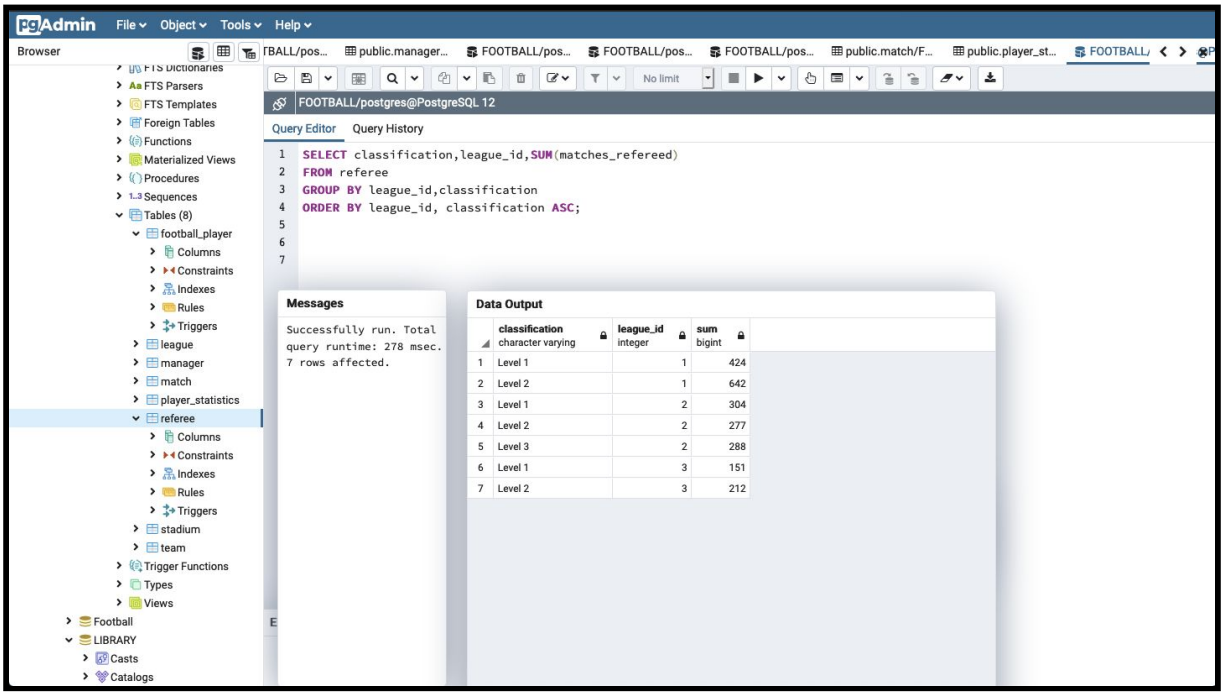
Output -



4. Find the total number of matches refereed by all referees from each country. List the classification for each group of referees and sort them by their league_ids.

```
SELECT classification,league_id,SUM(matches_refereed)
FROM referee
GROUP BY league_id,classification
ORDER BY league_id, classification ASC;
```

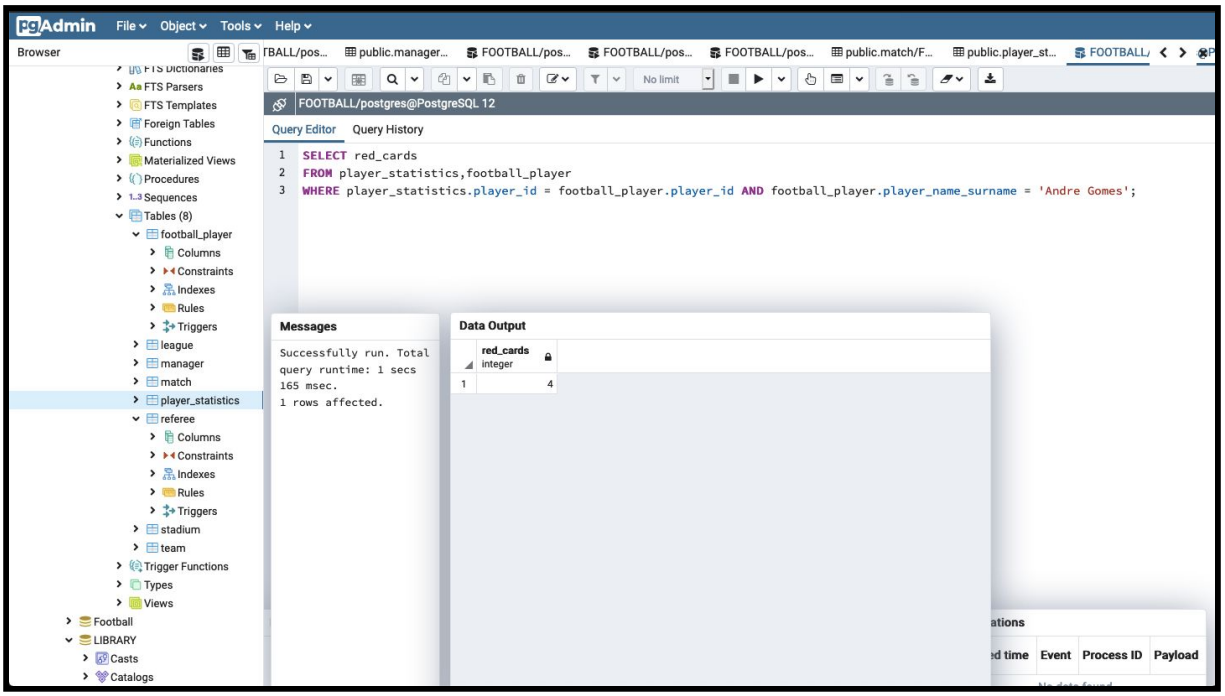
Output -



5. Find the total number of red cards received by a player named “Andre Gomes” in the Premier League.

```
SELECT red_cards
FROM player_statistics,football_player
WHERE player_statistics.player_id = football_player.player_id AND
football_player.player_name_surname = 'Andre Gomes';
```

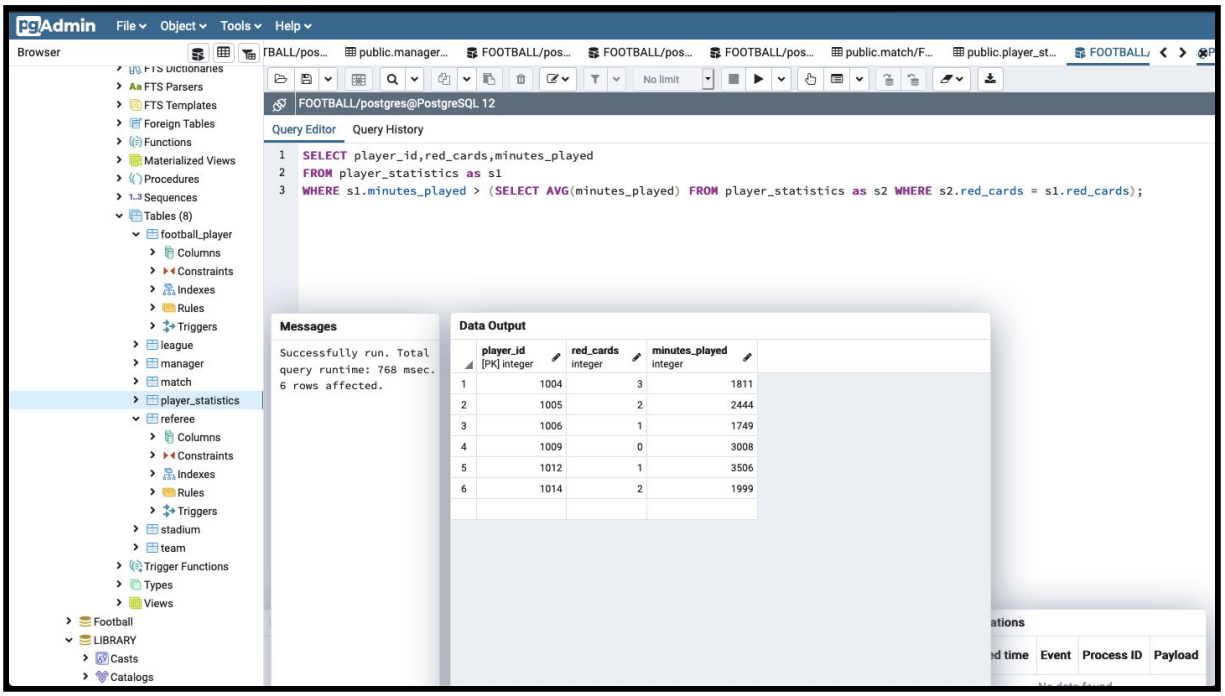
Output -



6. Find the player_ids of the players who have played for more average minutes amongst the players that have the same number of red cards.

SELECT player_id,red_cards,minutes_played
FROM player_statistics as s1
WHERE s1.minutes_played > (SELECT AVG(minutes_played) FROM player_statistics as s2 WHERE s2.red_cards = s1.red_cards);

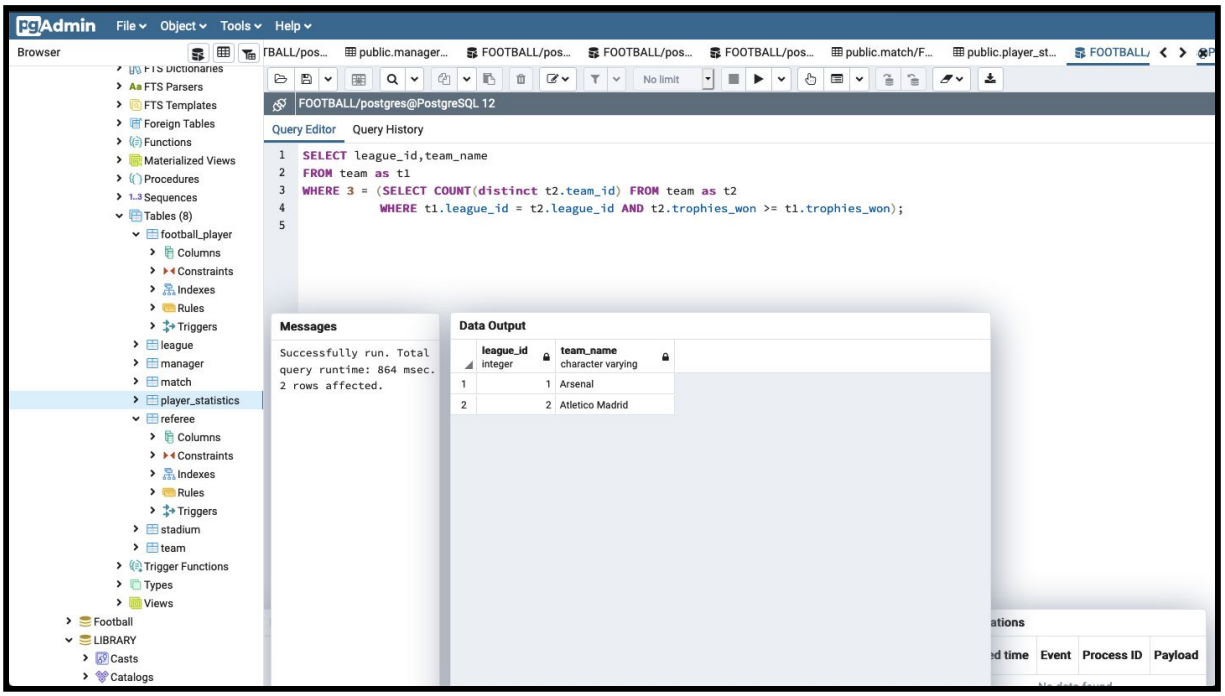
Output -



7. Find the names of the teams that have won the third highest number of trophies in their respective leagues.

```
SELECT league_id,team_name
FROM team as t1
WHERE 3 = (SELECT COUNT(distinct t2.team_id) FROM team as t2
                    WHERE t1.league_id = t2.league_id AND t2.trophies_won >=
t1.trophies_won);
```

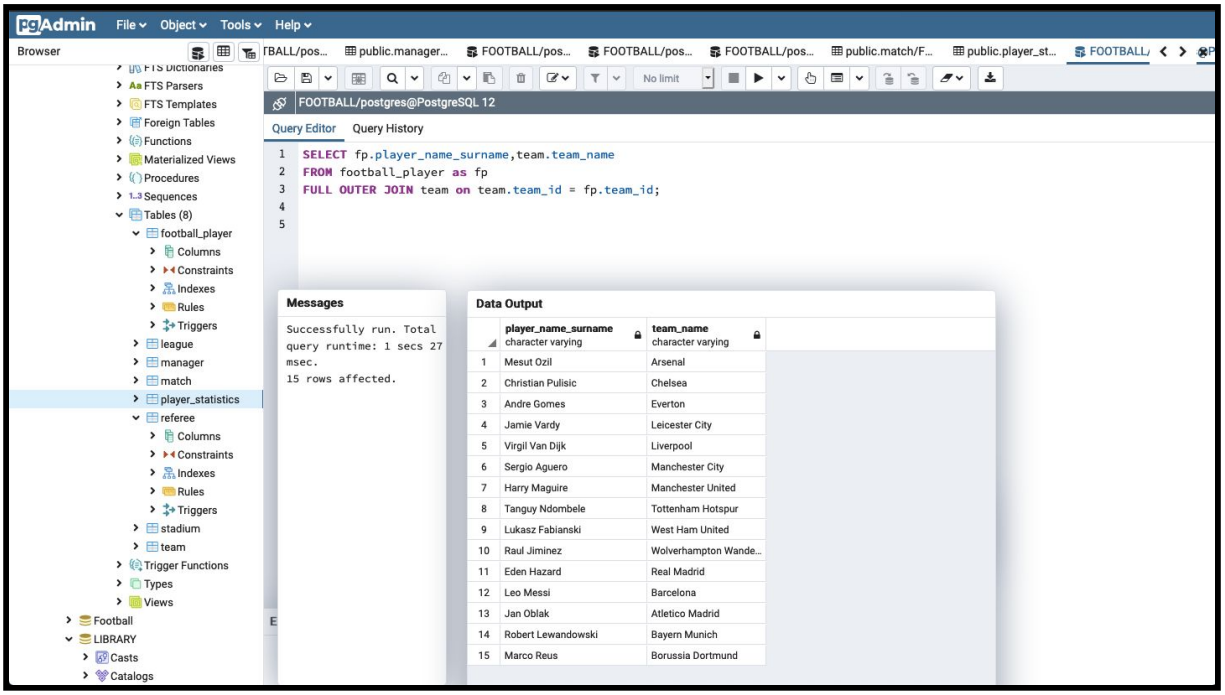
Output -



8. Find all the players and their team names.

```
SELECT fp.player_name_surname,team.team_name
FROM football_player as fp
FULL OUTER JOIN team on team.team_id = fp.team_id;
```

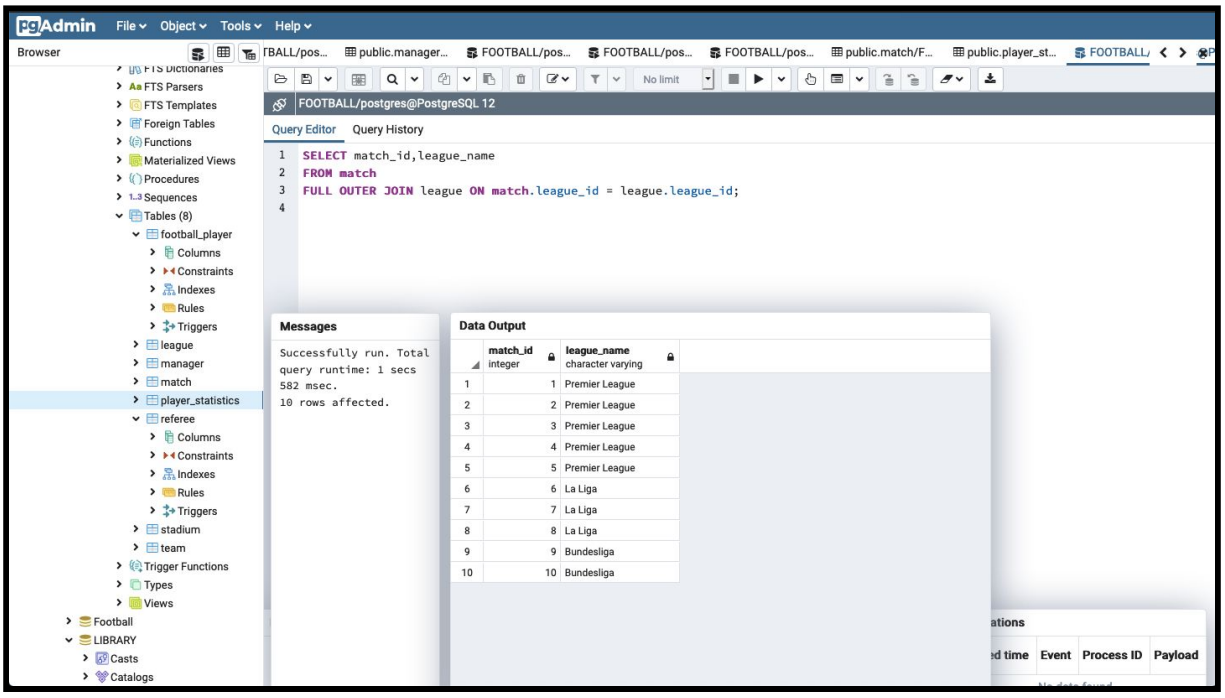
Output -



9. Find all the id's of all matches with league names.

```
SELECT match_id,league_name
FROM match
FULL OUTER JOIN league ON match.league_id = league.league_id;
```

Output -



Conclusion

The database system created quite efficiently performs the useful task of collecting data about the various entities and displaying their relationship in the mini world. We can see how these relationships affect each other and change the way the data is represented in the database. The database currently has some data on the three chosen major leagues in Europe, but it can be easily expanded to other leagues as well as the relationships between the entities is clearly defined and stored in the third normal forms. Some useful information can be extracted from the database in the form of retrieval queries, some of which have been mentioned in the above sections and the semantic constraints has been upheld through the use of a trigger. Additional data can be added to the tables, and the tables can even be normalized to higher normal forms (like Boyce-Codd Normal Form) to ensure no data redundancy or mutli-valued dependencies are present. Real time data can be obtained using web scraping from online sources to give an accurate representation about the current data in the tables. A transfer market to represent transactions between the various entities like the transfer of players between various teams and leagues based on their current market value could be implemented for a better understanding of the fast paced changes in the footballing world. One limitation of the current system is that there aren't any particular security measures implemented or permissions granted/revoked and hence implementing database security measures for the same can be done as well by giving the appropriate permissions, which would enable the administrator to ensure that the integrity of the database is maintained. A user friendly and interactive UI can also be made for the application, which would enable the user to visualize the information being represented and insert and make changes to the data more easily.

In conclusion, the database designed in the project performs the required applications quite well while keeping in mind the design constraints and provides useful information about the various football teams and leagues around the world.