Toronto Metropolitan University

Department of Electrical, Computer, & Biomedical Engineering
Faculty of Engineering & Architectural Science

Course Title:	Database Systems I
Course Number:	CPS 510
Semester/Year (e.g.F2016)	F2024

Instructor:	Dr. Soheila Bashardoust Tajali

Assignment/Lab Number:	Assignment 10
Assignment/Lab Title:	Soccer League Management System Technical Report

Submission Date:	November 29, 2024
Due Date:	November 29, 2024

Student LAST Name	**Student FIRST Name**	**Student Number**	**Section**	**Signature***
Guirguis	Joseph	501172194	10	
Gurakuqi	Sindi	501197737	10	
Vallipuranathan	Arathi	501168322	07	

*By signing above you attest that you have contributed to this written lab report and confirm that all work you have contributed to this lab report is your own work. Any suspicion of copying or plagiarism in this work will result in an investigation of Academic Misconduct and may result in a "0" on the work, an "F" in the course, or possibly more severe penalties, as well as a Disciplinary Notice on your academic record under the Student Code of Academic Conduct, which can be found online at: http://www.ryerson.ca/senate/current/pol60.pdf

Table of Contents

1 Application Description

1.1 Introduction

1.1.1 Overview

The soccer league management system is designed to assist in organizing and managing soccer league championships. This system allows users to manage teams, schedule matches, track player performance, and maintain standings. The system aims to improve operational efficiency and provide real-time updates for the performances of teams and players.

1.2.1 Objectives

The main objectives of the soccer league management system are as follows:

1. **Schedule matches** between teams in a league, including assigning referees, dates, and locations and recording match results.
2. **Track statistics** of teams and individual players, including goals, assists, wins, and losses.
3. **Update standings** within a league based on metrics such as wins, draws, and points.

1.3.1 Users

The different types of users who can use this system are as follows:

1. **Player:** Plays the matches and can see their contribution for the team.
2. **Team Management:** Manages the operation of a team, including president, coach and manager.
3. **Referees:** Officiates matches and inputs match results.
4. **Confederation Admin:** Manages the overall operation of a league, including the qualification of teams, creation of matches, and updating of player and match statistics.

1.2 Functions of system

1.2.1 Competition Admin

Function: Competition admin may create and administer leagues, create and qualify teams, and schedule and update matches.
Description: The competition admin entity has permissions (based on their role) to manage overarching operations, including creating new leagues, updating league details, adding or updating teams within the leagues, and scheduling and updating matches.

- **Relationships:**
    - Competition admin *administrates* League

    - Competition admin *qualifies* Teams
    - Competition admin *schedules* Match
- **Data/tables required:**
    - League table with field league ID to store and update league information.
    - Team table and fields team ID, league ID to store and manage team information.
    - Match table with fields match ID, home team ID, away team ID, and referee ID to schedule and update match details.

1.2.2 Player Management

Function: All players are members of a single team.
Description: Players in the system are associated with only one team at any given time, but a team can have multiple players. The system tracks their individual profiles and performance statistics.

- **Relationships**: Player is *member of* Team.
- **Data/tables required:**
    - Player table with fields player ID and team ID to store player information.
    - Team table with field team ID to associate each player with a team.

1.2.3 Referee Management

Function: Referees officiate matches and update the results of that match.
Description: Referees are assigned to matches to officiate them. A referee can only officiate one match at a time.

- **Relationships:** Referee *officiates* Match
- **Data/tables required:**
    - Referee table with field referee ID to store referee information.
    - Match table with field match ID and referee ID to assign referees to matches.

1.2.4 Team Management

Function: Management oversees the operations of their team.
Description: Each team has a management that is responsible for its operations. Management can only belong to one team, but a team can have multiple management. They are responsible for managing players, team details, and performance.

- **Relationships:** Management *manages* Team
- **Data/tables required:**
    - Management table with fields management ID and team ID to store management details.
    - Team table with field team ID to associate each management with a team.

1.2.5 Match Management

Function: The system can schedule matches between two teams and updates the performance of the teams and players based on the results.
Description: Matches are scheduled by the Confederation Admin. Once a match is completed, the referee will update the performance statistics.

- **Relationships:**
  - Competition Admin *schedules* Match
  - Referee *officiates* Match
  - Match *updates stats for* Team
  - Match *updates stats for* Player
- **Data/tables required:**
  - Match table with fields match ID, team ID to store match details.
  - Team table with field team ID to update the team's stats.
  - Player table with field player ID to update the player's stats.

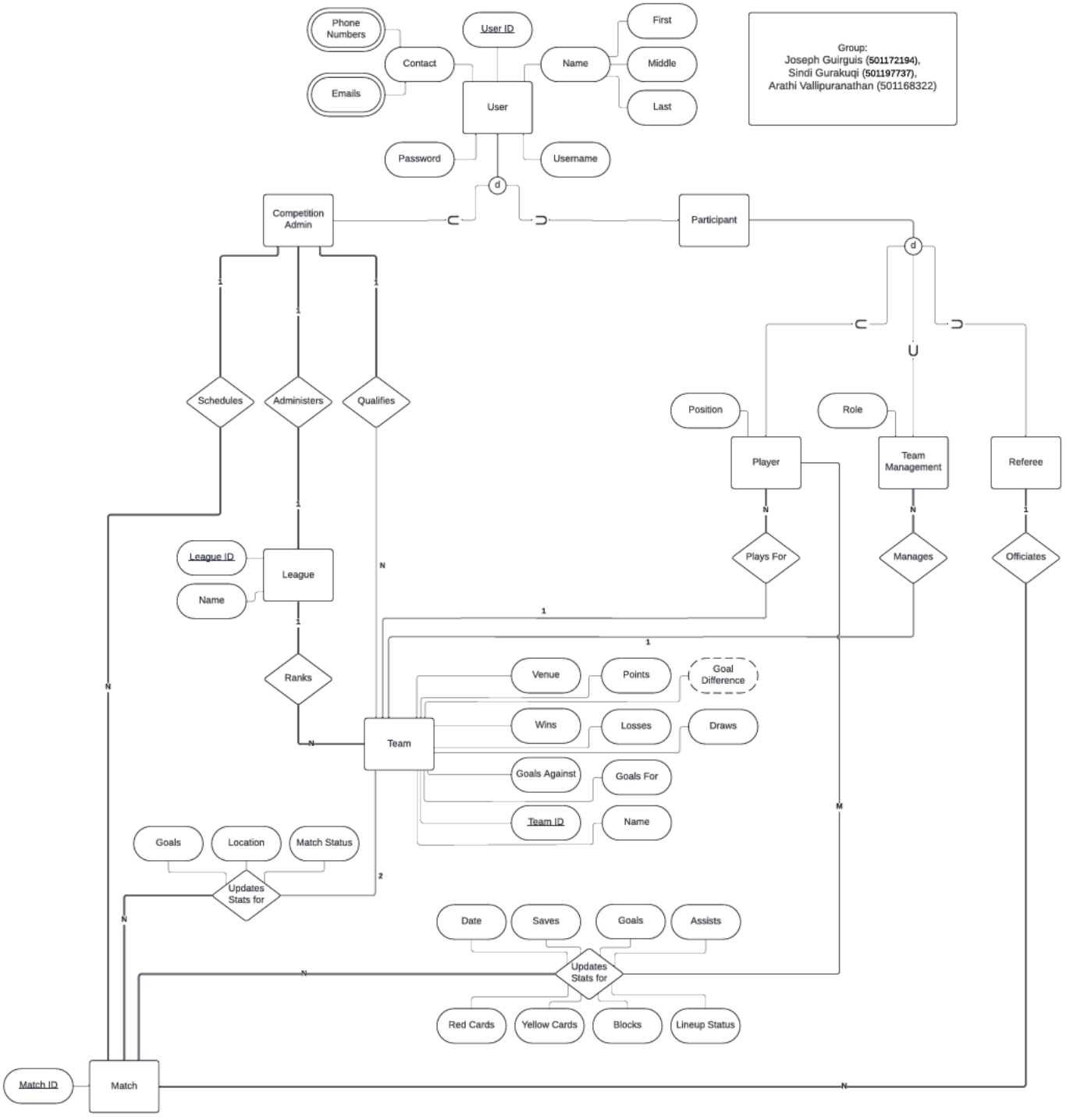
1.2.6 League Ranking

Function: The system can rank teams within a league based on their match performances.
Description: The league system calculates and updates the rankings of the teams based on the points they have earned through matches.

- **Relationships:**
  - League *ranks* Team
- **Data/tables required:**
  - League table with fields league ID and team ID to manage the league.
  - Team table with field team ID to track and rank teams.

2 Non-Normalized ER Model

2.1 Entity Relationship Diagram



3 Schema Design

3.1 Create CompetitionAdmin

```
-- Creating the tables derived from the ER diagram
CREATE TABLE CompetitionAdmin (
   CompetitionAdminID NUMBER PRIMARY KEY,
   FirstName VARCHAR2(25),
   LastName VARCHAR2(25),
   Email VARCHAR2(50) UNIQUE,
   PhoneNumber VARCHAR2(20) UNIQUE,
   Username VARCHAR2(24) NOT NULL UNIQUE,
   AdminPassword VARCHAR2(255) NOT NULL
);
```

3.2 Create League

```
CREATE TABLE League (
   LeagueID NUMBER PRIMARY KEY,
   LeagueName VARCHAR2(30) NOT NULL,
   CompetitionAdminID NUMBER,
   CONSTRAINT fk_admin_for_league FOREIGN KEY (CompetitionAdminID)
   REFERENCES CompetitionAdmin(CompetitionAdminID)
);
```

3.3 Create Team

```
CREATE TABLE Team (
   TeamID NUMBER PRIMARY KEY,
   LeagueID NUMBER NOT NULL,
   CompetitionAdminID NUMBER NOT NULL,
   TeamName VARCHAR2(50) NOT NULL,
   Points NUMBER DEFAULT 0,
   Wins NUMBER DEFAULT 0,
   Losses NUMBER DEFAULT 0,
   Draws NUMBER DEFAULT 0,
   GoalsFor NUMBER DEFAULT 0,
   GoalsAgainst NUMBER DEFAULT 0,
   Venue VARCHAR2(50),
   GoalDifference NUMBER AS (GoalsFor - GoalsAgainst),
```

```
    CONSTRAINT fk_league_for_team FOREIGN KEY (LeagueID)
      REFERENCES League(LeagueID),
    CONSTRAINT fk_admin_for_team FOREIGN KEY (CompetitionAdminID)
      REFERENCES CompetitionAdmin(CompetitionAdminID),
    CHECK (GoalsFor >= 0),
    CHECK (GoalsAgainst >= 0)
);
```

3.4 Create Player

```
CREATE TABLE Player (
    PlayerID NUMBER PRIMARY KEY,
    TeamID NUMBER NOT NULL,
    FirstName VARCHAR2(25) NOT NULL,
    LastName VARCHAR2(25) NOT NULL,
    Email VARCHAR2(50) UNIQUE,
    PhoneNumber VARCHAR2(20) UNIQUE,
    Username VARCHAR2(24) NOT NULL UNIQUE,
    PlayerPassword VARCHAR2(255) NOT NULL,
    PlayerPosition VARCHAR2(15),
    CHECK (PlayerPosition IN ('Goalkeeper', 'Defender', 'Midfielder', 'Forward')),
    CONSTRAINT fk_team_for_player FOREIGN KEY (TeamID) REFERENCES
Team(TeamID)
);
```

3.5 Create TeamManagement

```
CREATE TABLE TeamManagement (
    TeamManagementID NUMBER PRIMARY KEY,
    TeamID NUMBER NOT NULL,
    FirstName VARCHAR2(25) NOT NULL,
    LastName VARCHAR2(25) NOT NULL,
    Email VARCHAR2(50) UNIQUE,
    PhoneNumber VARCHAR2(20) UNIQUE,
    Username VARCHAR2(24) NOT NULL UNIQUE,
    ManagementPassword VARCHAR2(255) NOT NULL,
    TeamRole VARCHAR2(15),
    CONSTRAINT fk_team_management_for_team FOREIGN KEY (TeamID) REFERENCES
Team(TeamID)
);
```

3.6 Create Referee

```
CREATE TABLE Referee (
   RefereeID NUMBER PRIMARY KEY,
   FirstName VARCHAR2(25) NOT NULL,
   LastName VARCHAR2(25) NOT NULL,
   Email VARCHAR2(50) NOT NULL UNIQUE,
   PhoneNumber VARCHAR2(20) UNIQUE,
   Username VARCHAR2(24) NOT NULL UNIQUE,
   RefereePassword VARCHAR2(255) NOT NULL
);
```

3.7 Create Game

```
CREATE TABLE Game (
   GameID NUMBER PRIMARY KEY,
   CompetitionAdminID NUMBER NOT NULL,
   RefereeID NUMBER NOT NULL,
   GameLocation VARCHAR2(50) NOT NULL,
   GameDate DATE NOT NULL,
   GameTime TIMESTAMP,
   MatchStatus VARCHAR2(10) DEFAULT 'SCHEDULED',
   CONSTRAINT fk_admin_for_game FOREIGN KEY (CompetitionAdminID)
      REFERENCES CompetitionAdmin(CompetitionAdminID),
   CONSTRAINT fk_referee_for_game FOREIGN KEY (RefereeID)
      REFERENCES Referee(RefereeID),
   CHECK (MatchStatus IN ('SCHEDULED', 'COMPLETED', 'CANCELLED','DELAYED'))
);
```

3.8 Create GameTeamStats

```
CREATE TABLE GameTeamStats (
   GameID NUMBER NOT NULL,
   TeamID NUMBER NOT NULL,
   Goals NUMBER DEFAULT 0 NOT NULL,
   PRIMARY KEY (GameID, TeamID),
   CONSTRAINT fk_game_for_gameTeamStats FOREIGN KEY (GameID) REFERENCES
Game(GameID),
   CONSTRAINT fk_team_for_gameTeamStats FOREIGN KEY (TeamID) REFERENCES
Team(TeamID)
);
```

3.9 Create GamePlayerStats

```
CREATE TABLE GamePlayerStats (
   GameID NUMBER NOT NULL,
   PlayerID NUMBER NOT NULL,
   PlayerSaves NUMBER DEFAULT 0,
   PlayerGoals NUMBER DEFAULT 0,
   PlayerAssists NUMBER DEFAULT 0,
   PlayerBlocks NUMBER DEFAULT 0,
   PlayerRedCards NUMBER DEFAULT 0,
   PlayerYellowCards NUMBER DEFAULT 0,
   PlayerLineupStatus VARCHAR2(12) DEFAULT 'RESERVES',
   PRIMARY KEY (GameID, PlayerID),
   CONSTRAINT fk_game_for_gamePlayerStats FOREIGN KEY (GameID) REFERENCES
Game(GameID),
   CONSTRAINT fk_player_for_gamePlayerStats FOREIGN KEY (PlayerID) REFERENCES
Player(PlayerID),
   CHECK (PlayerLineupStatus IN ('FIRSTTEAM', 'SUBSTITUTE', 'RESERVES'))
);
```

4 Views/Queries

4.1 All Views

VIEW #1

A general fan wants to see all the games that were played for his favorite league, and the league is like a parameter that the user can choose with the view abstraction.

```
CREATE VIEW fixtures AS
SELECT LEAST(t1.TeamName,t2.TeamName) AS
Team1,GREATEST(t1.TeamName,t2.TeamName) AS Team2,
gts1.Goals AS Team1Goals, gts2.Goals AS Team2Goals,
g.GameDate,g.RefereeID,g.MatchStatus,l.LeagueName,l.LeagueID
FROM Game g, GameTeamStats gts1, GameTeamStats gts2, Team t1, Team t2,League
l,CompetitionAdmin c
WHERE gts1.TeamID = t1.TeamID AND gts2.TeamID = t2.TeamID AND
g.CompetitionAdminID = l.CompetitionAdminID
AND g.GameID = gts1.GameID AND g.GameID = gts2.GameID AND gts1.TeamID !=
gts2.TeamID
AND (t1gi.TeamName != LEAST(t1.TeamName,t2.TeamName)) AND l.CompetitionAdminID =
c.CompetitionAdminID
ORDER BY g.GameDate;
```

VIEW #2

The Competition Admin (Joseph Guirguis) wants to make sure all referees in the league (Mississauga League) have officiated an equal amount of games. Users can choose whatever league they want in the view abstraction.

```
CREATE VIEW EachRefereeOfficiatedTotal AS
SELECT c.Username AS
AdminUsername,l.LeagueName,r.FirstName,r.LastName,COUNT(r.RefereeID) AS
GamesOfficiated,l.LeagueID
FROM CompetitionAdmin c, League l, Referee r, Game g
WHERE c.CompetitionAdminID = l.CompetitionAdminID AND g.RefereeID = r.RefereeID
AND g.CompetitionAdminID = l.CompetitionAdminID
GROUP BY c.Username,l.LeagueName,r.FirstName,r.LastName,l.LeagueID
ORDER BY GamesOfficiated;
```

VIEW #3
A general fan wants to go on the website and check the player with the most goals scored in the league to see if his favorite player is doing well this season. Users will be able to choose the league they want with the view abstraction.

```
CREATE VIEW playersRankedByGoals AS
SELECT p.PlayerID,p.FirstName, p.LastName,p.PlayerPosition, t.TeamName,
SUM(gps.PlayerGoals) AS TotalGoals,l.LeagueID
FROM Player p, GamePlayerStats gps, Team t, League l
WHERE p.PlayerID = gps.PlayerID AND p.TeamID = t.TeamID AND l.LeagueID = t.LeagueID
GROUP BY p.PlayerID,p.FirstName, p.LastName,p.PlayerPosition, t.TeamName,l.LeagueID
ORDER BY TotalGoals DESC;
```

VIEW #4
I am the Manager and I want to see the players who have an average goal scoring rate higher than 0.5 to see who is providing the most value playing in the 'FIRSTTEAM' so that I can later adjust salaries and the First Team lineup. Specific manager's team data can be selected using view abstraction.

```
CREATE VIEW GoodPlayerPerformance AS
SELECT tm.Username,p.FirstName, p.LastName,p.PlayerPosition, t.TeamName,
AVG(gps.PlayerGoals) AS GoalsPerGame
FROM Player p, GamePlayerStats gps, Team t, TeamManagement tm
WHERE p.PlayerID = gps.PlayerID AND p.TeamID = t.TeamID AND tm.TeamID = p.TeamID
GROUP BY tm.Username,p.FirstName, p.LastName,p.PlayerPosition, t.TeamName
HAVING AVG(gps.PlayerGoals) > 0.5
ORDER BY GoalsPerGame DESC;
```

4.2 Results of All Queries that are not Views

Query 1

```
/*
1) The Manager (Sindi1 with username sindig1) of Team 17 (Mississauga Warriors) wants to get all the players in her team to see how the lineup looks like for the next game.
I want the database to search username currently using the website to get all players of the team Sindi1 manages.
Maybe Sindi1 wants to update the lineup status or position of a player for next game because a player has poor performance or is injured or has a red card.
```

```
*/

SELECT FirstName, LastName, PlayerPosition
FROM Player
WHERE TeamID = (
   SELECT TeamID
   FROM TeamManagement
   WHERE Username = 'sindig1'
);
```

	FIRSTNAME	LASTNAME	PLAYERPOSITION
1	Aiden	Aiden	Goalkeeper
2	Andrew	Andrew	Defender
3	Aaron	Aaron	Midfielder
4	Adam	Adam	Forward

Query 2

```
/*
2) I am the Competition Admin (Joseph Guirguis with username joe027) and I want to see how the standings look, because by the end of the league, I will transfer the top 4 teams to another competition admin who is controlling a league that is more prestigious.
*/

SELECT TeamName, Wins, Draws, Losses, GoalsFor, GoalsAgainst, GoalDifference, Points
FROM Team
WHERE LeagueID = (
   SELECT l.LeagueID
   FROM CompetitionAdmin c, League l
   WHERE c.Username = 'joe027' AND l.CompetitionAdminID = c.CompetitionAdminID
)
ORDER BY Points DESC, GoalDifference DESC, GoalsFor DESC;
```

	TEAMNAME	WINS	DRAWS	LOSSES	GOALSFOR	GOALSAGAINST	GOALDIFFERENCE	POINTS
1	Mississauga Warriors	3	1	2	14	7	7	10
2	Mississauga Dragons	2	2	2	14	12	2	8
3	Mississauga Bull Dogs	2	2	2	14	14	0	8
4	Mississauga Eagles	2	1	3	11	20	-9	7

Query 3

/*
3) A special trophy is given to those who got the most hat tricks or higher (3 goals or more in a single game). The competition Admin (Joseph Guirguis) wants to find who got the most number of hat tricks (more than 3 in a game counts as a hattrick).
*/

```
SELECT p.PlayerID, p.FirstName, p.LastName, p.PlayerPosition,
t.TeamName,COUNT(gps.PlayerID) AS hattricks
FROM GamePlayerStats gps, Player p, Game g, Team t
WHERE p.PlayerID = gps.PlayerID AND gps.PlayerGoals >= 3 AND g.GameID = gps.GameID
AND p.TeamID = t.TeamID
GROUP BY p.PlayerID, p.FirstName, p.LastName, p.PlayerPosition, t.TeamName
ORDER BY hattricks DESC;
```

	PLAYERID	FIRSTNAME	LASTNAME	PLAYERPOSITION	TEAMNAME	HATTRICKS
1	52	Darren	Darren	Forward	Mississauga Bull Dogs	2
2	39	Aaron	Aaron	Midfielder	Mississauga Warriors	1
3	40	Adam	Adam	Forward	Mississauga Warriors	1

Query 4 **NOTE:** This query also shows that VIEW #1 works

/*
4) A general fan wants to see all the games happening in october except the first week for the Mississauga League
*/

```
SELECT *
FROM (
   SELECT *
   FROM fixtures
   WHERE LeagueID = 4
   MINUS
```

```
    SELECT *
    FROM fixtures
    WHERE GameDate BETWEEN TO_DATE('2024-10-01', 'YYYY-MM-DD') AND
TO_DATE('2024-10-07', 'YYYY-MM-DD') AND LeagueID = 4
)
ORDER BY GameDate;
```

	TEAM1	TEAM2	TEAM1GOALS	TEAM2GOALS	GAMEDATE	REFEREEID	MATCHSTATUS	LEAGUENAME	LEAGUEID
1	Mississauga Dragons	Mississauga Eagles	1	2	24-10-08	4	SCHEDULED	Mississauga League	4
2	Mississauga Bull Dogs	Mississauga Eagles	2	3	24-10-09	1	SCHEDULED	Mississauga League	4
3	Mississauga Bull Dogs	Mississauga Warriors	0	1	24-10-10	2	SCHEDULED	Mississauga League	4
4	Mississauga Bull Dogs	Mississauga Dragons	2	2	24-10-11	3	SCHEDULED	Mississauga League	4
5	Mississauga Bull Dogs	Mississauga Eagles	3	2	24-10-12	4	SCHEDULED	Mississauga League	4

Query 5

```
/*
5) I want to find all the players with more than 2 yellow cards for the Mississauga League
*/
SELECT p.PlayerID, p.FirstName, p.LastName, p.TeamID, SUM(gps.PlayerYellowCards) as
TotalYellowCards
FROM Player p, GamePlayerStats gps, Team t, League l
WHERE p.PlayerID = gps.PlayerID
AND p.TeamID = t.TeamID
AND t.LeagueID = l.LeagueID
AND l.LeagueID = 4
GROUP BY p.PlayerID, p.FirstName, p.LastName, p.TeamID
HAVING SUM(gps.PlayerYellowCards) >= 2;
```

	PLAYERID	FIRSTNAME	LASTNAME	TEAMID	TOTALYELLOWCARDS
1	37	Aiden	Aiden	10	5
2	38	Andrew	Andrew	10	2
3	45	Caleb	Caleb	12	2

Query 6

```
/*
6) Get players who received yellow cards using the EXISTS operator.
*/
SELECT p.PlayerID, p.FirstName, p.LastName, t.TeamName
```

```
FROM Player p
JOIN Team t ON p.TeamID = t.TeamID
WHERE EXISTS (
   SELECT *
   FROM GamePlayerStats gps
   JOIN Player p2 ON gps.PlayerID = p2.PlayerID
   WHERE p2.TeamID = t.TeamID
   AND gps.PlayerYellowCards > 0
);
```

	PLAYERID	FIRSTNAME	LASTNAME	TEAMNAME
1	37	Aiden	Aiden	Mississauga Warriors
2	38	Andrew	Andrew	Mississauga Warriors
3	39	Aaron	Aaron	Mississauga Warriors
4	40	Adam	Adam	Mississauga Warriors
5	45	Caleb	Caleb	Mississauga Eagles
6	46	Colton	Colton	Mississauga Eagles
7	47	Charles	Charles	Mississauga Eagles
8	48	Collin	Collin	Mississauga Eagles
9	49	Drake	Drake	Mississauga Bull Dogs
10	50	David	David	Mississauga Bull Dogs
11	51	Dorris	Dorris	Mississauga Bull Dogs
12	52	Darren	Darren	Mississauga Bull Dogs

Query 7

```
/*
7) Get players who have scored but never received a yellow card using MINUS operator.
*/
SELECT p.PlayerID, p.FirstName, p.LastName
FROM Player p
JOIN GamePlayerStats gps ON p.PlayerID = gps.PlayerID
WHERE gps.PlayerGoals > 0
MINUS
SELECT p.PlayerID, p.FirstName, p.LastName
FROM Player p
JOIN GamePlayerStats gps ON p.PlayerID = gps.PlayerID
WHERE gps.PlayerYellowCards > 0;
```

	PLAYERID	FIRSTNAME	LASTNAME
1	40	Adam	Adam
2	42	Blake	Blake
3	43	Belle	Belle
4	44	Boron	Boron
5	46	Colton	Colton
6	47	Charles	Charles
7	48	Collin	Collin
8	51	Dorris	Dorris
9	52	Darren	Darren

Query 8

```
/*
8) Retrieve all players who either scored more than 2 goals and/or assisted 1 goal or more in a
game
*/
SELECT p.PlayerID,p.FirstName,p.LastName,gps.PlayerGoals,gps.PlayerAssists
FROM GamePlayerStats gps,Player p
WHERE gps.PlayerGoals > 2 and p.PlayerID = gps.PlayerID
UNION
SELECT p.PlayerID,p.FirstName,p.LastName,gps.PlayerGoals,gps.PlayerAssists
FROM GamePlayerStats gps, Player p
WHERE gps.PlayerAssists > 0 and p.PlayerID = gps.PlayerID;
```

	PLAYERID	FIRSTNAME	LASTNAME	PLAYERGOALS	PLAYERASSISTS
1	39	Aaron	Aaron	5	0
2	40	Adam	Adam	4	0
3	43	Belle	Belle	0	1
4	43	Belle	Belle	1	1
5	47	Charles	Charles	0	1
6	47	Charles	Charles	1	1
7	48	Collin	Collin	0	1
8	51	Dorris	Dorris	0	1
9	52	Darren	Darren	3	0

4.3 Results of All Queries that are Views

Query 1

NOTE: VIEW #1 was tested in query 4.

Query 2

```
SELECT FirstName,LastName,GamesOfficiated
FROM EachRefereeOfficiatedTotal
WHERE LeagueID = 4
```

	FIRSTNAME	LASTNAME	GAMESOFFICIATED
1	Michael	Johnson	3
2	Jane	Doe	3
3	David	Williams	3
4	James	Brown	3

Query 3

```
SELECT PlayerID,FirstName, LastName,PlayerPosition,TeamName, TotalGoals
FROM playersRankedByGoals
WHERE LeagueID = 4;
```

	PLAYERID	FIRSTNAME	LASTNAME	PLAYERPOSITION	TEAMNAME	TOTALGOALS
1	52	Darren	Darren	Forward	Mississauga Bull Dogs	9
2	39	Aaron	Aaron	Midfielder	Mississauga Warriors	6
3	40	Adam	Adam	Forward	Mississauga Warriors	6
4	44	Boron	Boron	Forward	Mississauga Dragons	5
5	47	Charles	Charles	Midfielder	Mississauga Eagles	4
6	48	Collin	Collin	Forward	Mississauga Eagles	4
7	42	Blake	Blake	Defender	Mississauga Dragons	4
8	43	Belle	Belle	Midfielder	Mississauga Dragons	3
9	50	David	David	Defender	Mississauga Bull Dogs	3
10	51	Dorris	Dorris	Midfielder	Mississauga Bull Dogs	2
11	46	Colton	Colton	Defender	Mississauga Eagles	2
12	38	Andrew	Andrew	Defender	Mississauga Warriors	2
13	37	Aiden	Aiden	Goalkeeper	Mississauga Warriors	0
14	41	Beau	Beau	Goalkeeper	Mississauga Dragons	0
15	45	Caleb	Caleb	Goalkeeper	Mississauga Eagles	0
16	49	Drake	Drake	Goalkeeper	Mississauga Bull Dogs	0

Query 4

```
SELECT Username,FirstName AS PlayerFirstName, LastName AS
PlayerLastName,PlayerPosition, TeamName, GoalsPerGame
FROM GoodPlayerPerformance
WHERE Username = 'arathiv2';
```

	USERNAME	PLAYERFIRSTNAME	PLAYERLASTNAME	PLAYERPOSITION	TEAMNAME	GOALSPERGAME
1	arathiv2	Boron	Boron	Forward	Mississauga Dragons	0.833333333333333333333333333333333333333
2	arathiv2	Blake	Blake	Defender	Mississauga Dragons	0.666666666666666666666666666666666666667

5 Unix Shell Implementation

5.1 Query 1

```
-- Retrieve all players who either scored more than 2 goals and/or assisted 1 goal or more in a
game using UNION
```

```
#!/bin/bash

export LD_LIBRARY_PATH=/usr/lib/oracle/12.1/client64/lib
sqlplus64 "username/password@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(Host=orac
le.scs.ryerson.ca)(Port=1521))(CONNECT_DATA=(SID=orcl)))" <<EOF

-- Retrieve all players who either scored more than 2 goals and/or assisted
 1 goal or more in a game using UNION

SELECT p.PlayerID,p.FirstName,p.LastName,gps.PlayerGoals,gps.PlayerAssists
FROM GamePlayerStats gps,Player p
WHERE gps.PlayerGoals > 2 and p.PlayerID = gps.PlayerID
UNION
SELECT p.PlayerID,p.FirstName,p.LastName,gps.PlayerGoals,gps.PlayerAssists
FROM GamePlayerStats gps, Player p
WHERE gps.PlayerAssists > 0 and p.PlayerID = gps.PlayerID;


exit;
EOF
~
```

Output:

```
SQL> SQL> SQL> SQL>   2    3    4    5    6    7
  PLAYERID FIRSTNAME                LASTNAME                  PLAYERGOALS
---------- ------------------------ ------------------------- -----------
PLAYERASSISTS
-------------
         3 Aaron                    Aaron                               5
           0

         4 Adam                     Adam                                4
           0

         7 Belle                    Belle                               0
           1


  PLAYERID FIRSTNAME                LASTNAME                  PLAYERGOALS
---------- ------------------------ ------------------------- -----------
PLAYERASSISTS
-------------
         7 Belle                    Belle                               1
           1

        11 Charles                  Charles                             0
           1

        11 Charles                  Charles                             1
           1


  PLAYERID FIRSTNAME                LASTNAME                  PLAYERGOALS
---------- ------------------------ ------------------------- -----------
PLAYERASSISTS
-------------
        12 Collin                   Collin                              0
           1

        15 Dorris                   Dorris                              0
           1

        16 Darren                   Darren                              3
           0
```

5.2 Query 2

-- See the players who scored higher than 0.5, used GROUP BY

```
#!/bin/bash

export LD_LIBRARY_PATH=/usr/lib/oracle/12.1/client64/lib
sqlplus64 "username/password@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(Host=orac
le.scs.ryerson.ca)(Port=1521))(CONNECT_DATA=(SID=orcl)))" <<EOF

-- To see the players who scored higher than 0.5, used GROUP BY

CREATE VIEW GoodPlayerPerformances AS
SELECT tm.Username,p.FirstName, p.LastName,p.PlayerPosition, t.TeamName, AV
G(gps.PlayerGoals) AS GoalsPerGame
FROM Player p, GamePlayerStats gps, Team t, TeamManagement tm
WHERE p.PlayerID = gps.PlayerID AND p.TeamID = t.TeamID AND tm.TeamID = p.T
eamID
GROUP BY tm.Username,p.FirstName, p.LastName,p.PlayerPosition, t.TeamName
HAVING AVG(gps.PlayerGoals) > 0.5
ORDER BY GoalsPerGame DESC;

exit;
EOF
```

Output:

```
SQL> SQL> SQL> SQL>   2    3    4    5    6
USERNAME                FIRSTNAME                LASTNAME
----------------------- ------------------------ ------------------------
PLAYERPOSITION  TEAMNAME                                 GOALSPERGAME
--------------- ---------------------------------------- ------------
arathiv4                Darren                   Darren
Forward         Mississauga Bull Dogs                             1.5

arathiv3                Darren                   Darren
Forward         Mississauga Bull Dogs                             1.5

sindig2                 Aaron                    Aaron
Midfielder      Mississauga Warriors                                1


USERNAME                FIRSTNAME                LASTNAME
----------------------- ------------------------ ------------------------
PLAYERPOSITION  TEAMNAME                                 GOALSPERGAME
--------------- ---------------------------------------- ------------
sindig1                 Adam                     Adam
Forward         Mississauga Warriors                                1

sindig2                 Adam                     Adam
Forward         Mississauga Warriors                                1

sindig1                 Aaron                    Aaron
Midfielder      Mississauga Warriors                                1


USERNAME                FIRSTNAME                LASTNAME
----------------------- ------------------------ ------------------------
PLAYERPOSITION  TEAMNAME                                 GOALSPERGAME
--------------- ---------------------------------------- ------------
arathiv1                Boron                    Boron
Forward         Mississauga Dragons                        .833333333

arathiv2                Boron                    Boron
Forward         Mississauga Dragons                        .833333333

sindig4                 Charles                  Charles
Midfielder      Mississauga Eagles                         .666666667
```

```
USERNAME                FIRSTNAME                LASTNAME
----------------------- ------------------------ ------------------------
PLAYERPOSITION  TEAMNAME                                 GOALSPERGAME
--------------- ---------------------------------------- ------------
arathiv2                Blake                    Blake
Defender        Mississauga Dragons                        .666666667

sindig3                 Collin                   Collin
Forward         Mississauga Eagles                         .666666667

sindig3                 Charles                  Charles
Midfielder      Mississauga Eagles                         .666666667


USERNAME                FIRSTNAME                LASTNAME
----------------------- ------------------------ ------------------------
PLAYERPOSITION  TEAMNAME                                 GOALSPERGAME
--------------- ---------------------------------------- ------------
sindig4                 Collin                   Collin
Forward         Mississauga Eagles                         .666666667

arathiv1                Blake                    Blake
Defender        Mississauga Dragons                        .666666667


14 rows selected.
```

5.3 Query 3

-- The Competition Admin wants to make sure all referees in the league (Mississauga League) have officiated an equal amount of games, uses COUNT and GROUP BY

```
#!/bin/bash

export LD_LIBRARY_PATH=/usr/lib/oracle/12.1/client64/lib
sqlplus64 "username/password@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(Host=orac
le.scs.ryerson.ca)(Port=1521))(CONNECT_DATA=(SID=orcl)))" <<EOF

-- The Competition Admin wants to make sure all referees in the league (Mis
sissauga League) have officiated an equal amount of games, uses COUNT and G
ROUP BY

CREATE VIEW EachRefereesOfficiatedTotal AS
SELECT c.Username AS AdminUsername,l.LeagueName,r.FirstName,r.LastName,COUN
T(r.RefereeID) AS GamesOfficiated,l.LeagueID
FROM CompetitionAdmin c, League l, Referee r, Game g
WHERE c.CompetitionAdminID = l.CompetitionAdminID AND g.RefereeID = r.Refer
eeID
AND g.CompetitionAdminID = l.CompetitionAdminID
GROUP BY c.Username,l.LeagueName,r.FirstName,r.LastName,l.LeagueID
ORDER BY GamesOfficiated;

exit;
EOF
```

Output:

```
SQL> SQL> SQL> SQL>   2    3    4    5    6
ADMINUSERNAME            LEAGUENAME
------------------------ ----------------------------------------
FIRSTNAME                 LASTNAME                  GAMESOFFICIATED   LEAGUEID
------------------------- ------------------------- --------------- ----------
joe027                   Mississauga League
James                     Brown                                   3          1

joe027                   Mississauga League
David                     Williams                                3          1

joe027                   Mississauga League
Jane                      Doe                                     3          1


ADMINUSERNAME            LEAGUENAME
------------------------ ----------------------------------------
FIRSTNAME                 LASTNAME                  GAMESOFFICIATED   LEAGUEID
------------------------- ------------------------- --------------- ----------
joe027                   Mississauga League
Michael                   Johnson                                 3          1
```

5.4 Query 4

Get players who have scored but never received a yellow card using MINUS operator.

Code:

```
#!/bin/bash

export LD_LIBRARY_PATH=/usr/lib/oracle/12.1/client64/lib
sqlplus64 "username/passsword@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(Host=oracle.scs.ryerson.ca)(Port=1521))
(CONNECT_DATA=(SID=orcl)))" <<EOF

-- 4) Get players who have scored but never received a yellow card using MINUS operator.

SELECT p.PlayerID, p.FirstName, p.LastName
FROM Player p
JOIN GamePlayerStats gps ON p.PlayerID = gps.PlayerID
WHERE gps.PlayerGoals > 0
MINUS
SELECT p.PlayerID, p.FirstName, p.LastName
FROM Player p
JOIN GamePlayerStats gps ON p.PlayerID = gps.PlayerID
WHERE gps.PlayerYellowCards > 0;

exit;
EOF
```

Output:

```
avallipu@europa:~/cps510$ bash interesting_query4.sh

SQL*Plus: Release 12.1.0.2.0 Production on Thu Oct 24 13:21:31 2024

Copyright (c) 1982, 2014, Oracle.  All rights reserved.


Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options

SQL> SQL> SQL> SQL>   2    3    4    5    6    7    8    9
  PLAYERID FIRSTNAME                LASTNAME
---------- ------------------------ ------------------------
         4 Adam                     Adam
         6 Blake                    Blake
         7 Belle                    Belle
         8 Boron                    Boron
        10 Colton                   Colton
        11 Charles                  Charles
        12 Collin                   Collin
        15 Dorris                   Dorris
        16 Darren                   Darren

9 rows selected.
```

5.5 Query 5

-- Get players who received no red cards using the NOT EXISTS operator.

Code:

```
#!/bin/bash

export LD_LIBRARY_PATH=/usr/lib/oracle/12.1/client64/lib
sqlplus64 "username/password@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(Host=oracle.scs.ryerson.ca)(Port=1521))
(CONNECT_DATA=(SID=orcl)))" <<EOF

-- 5) Get players who receive no red cards using the NOT EXISTS operator.

SELECT p.PlayerID, p.FirstName, p.LastName, t.TeamName
FROM Player p
JOIN Team t ON p.TeamID = t.TeamID
WHERE NOT EXISTS (
    SELECT *
    FROM GamePlayerStats gps
    JOIN Player p2 ON gps.PlayerID = p2.PlayerID
    WHERE p2.TeamID = t.TeamID
    AND gps.PlayerRedCards > 0
);


exit;
EOF
```

Output:

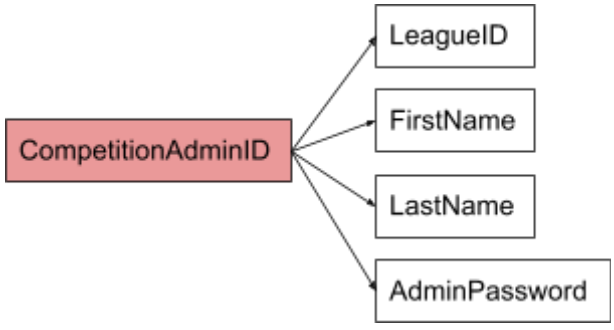
```
Running Query 5...

SQL*Plus: Release 12.1.0.2.0 Production on Fri Oct 25 12:07:04 2024

Copyright (c) 1982, 2014, Oracle.  All rights reserved.


Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options

SQL> SQL> SQL> SQL>   2    3    4    5    6    7    8    9   10
  PLAYERID FIRSTNAME                LASTNAME
---------- ------------------------ ------------------------
TEAMNAME
----------------------------------------
         5 Beau                     Beau
Mississauga Dragons

         6 Blake                    Blake
Mississauga Dragons

         7 Belle                    Belle
Mississauga Dragons
```

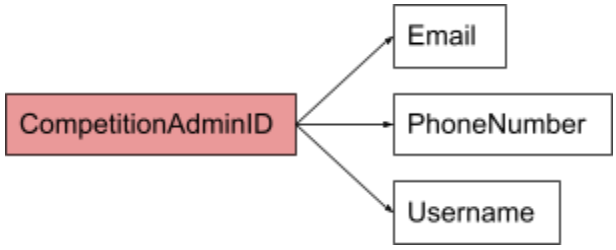
```
  PLAYERID FIRSTNAME                LASTNAME
---------- ------------------------ ------------------------
TEAMNAME
----------------------------------------
         8 Boron                    Boron
Mississauga Dragons

         9 Caleb                    Caleb
Mississauga Eagles

        10 Colton                   Colton
Mississauga Eagles


  PLAYERID FIRSTNAME                LASTNAME
---------- ------------------------ ------------------------
TEAMNAME
----------------------------------------
        11 Charles                  Charles
Mississauga Eagles

        12 Collin                   Collin
Mississauga Eagles

        13 Drake                    Drake
Mississauga Bull Dogs
```

```
  PLAYERID FIRSTNAME                LASTNAME
---------- ------------------------ ------------------------
TEAMNAME
----------------------------------------
        14 David                    David
Mississauga Bull Dogs

        15 Dorris                   Dorris
Mississauga Bull Dogs

        16 Darren                   Darren
Mississauga Bull Dogs


12 rows selected.
```

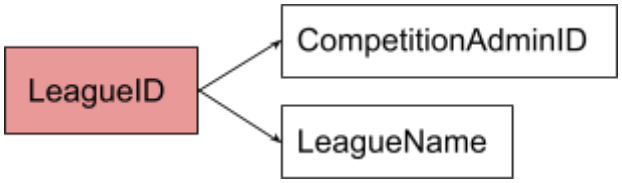
6 Normalization of Database

6.1 Competition Admin

CompetitionAdmin(CompetitionAdminID, LeagueID, FirstName, LastName, Email, PhoneNumber, Username, AdminPassword)	
Functional Dependencies	{CompetitionAdminID} → LeagueID, FirstName, LastName, Email, PhoneNumber, Username, AdminPassword {LeagueID} → CompetitionAdminID {Username} → CompetitionAdminID, FirstName, LastName, Email, PhoneNumber, AdminPassword {PhoneNumber} → CompetitionAdminID, FirstName, LastName, Email, Username, AdminPassword {Email} → CompetitionAdminID, FirstName, LastName, PhoneNumber, Username, AdminPassword
Derive 3NF	Non-key attributes (FirstName, LastName, AdminPassword) are transitively dependent on candidate keys {CompetitionAdminID}, which violates 3NF. This table can be decomposed into two tables: 1. **LeagueAdmin**(CompetitionAdminID, LeagueID, FirstName, LastName, AdminPassword) 2. **AdminContact**(CompetitionAdminID, Email, PhoneNumber, Username)
LeagueAdmin	 Table is: ● 1NF because all values are atomic ● 2NF because all non-key attributes are fully functionally dependent on the primary key ● 3NF because all non-key attributes are non-transitively dependent

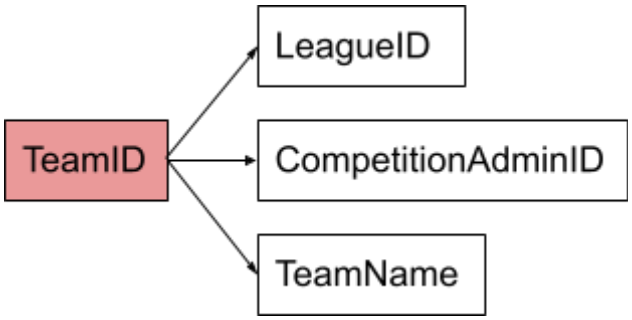
	on the primary key ● BCNF because **CompetitionAdminID → LeagueID, FirstName, LastName, AdminPassword** is the only functional dependency in the table, and CompetitionAdminID is a superkey since it determines all the other attributes in the tuple.
AdminContact	 Table is: ● 1NF because all values are atomic ● 2NF because all non-key attributes are fully functionally dependent on the primary key ● 3NF because all non-key attributes are non-transitively dependent on the primary key ● BCNF because **CompetitionAdminID → Email, PhoneNumber, Username** is the only functional dependency in the table, and CompetitionAdminID is a superkey since it determines all the other attributes in the tuple.

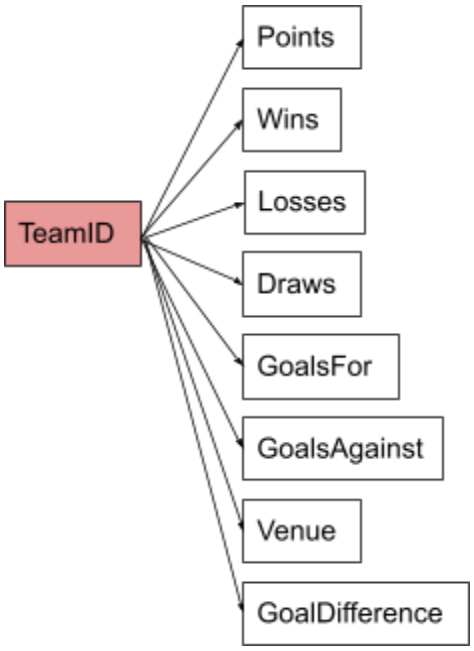
6.2 League

League(LeagueID, CompetitionAdminID, LeagueName)	
Functional Dependencies	{LeagueID} → CompetitionAdminID, LeagueName {CompetitionAdminID} → LeagueID
League	 Table is: ● 1NF because all values are atomic ● 2NF because all non-key attributes are fully functionally dependent on the primary key ● 3NF because all non-key attributes are non-transitively dependent on the primary key ● BCNF because **LeagueID → CompetitionAdminID, LeagueName** is the only functional dependency in the table, and LeagueID is a superkey since it determines all the other attributes

	in the tuple.

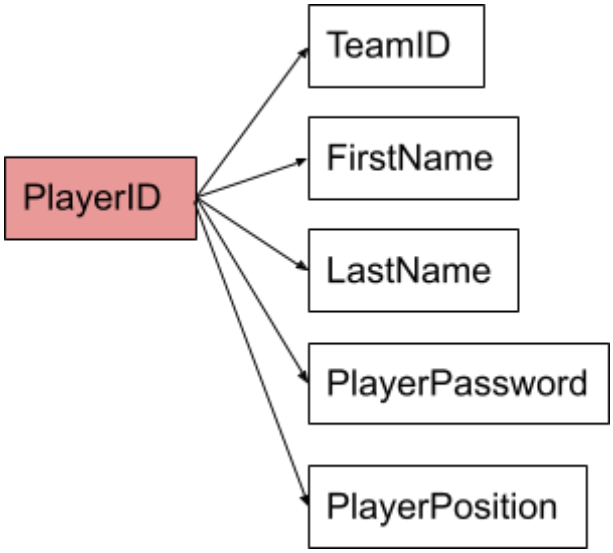
6.3 Team

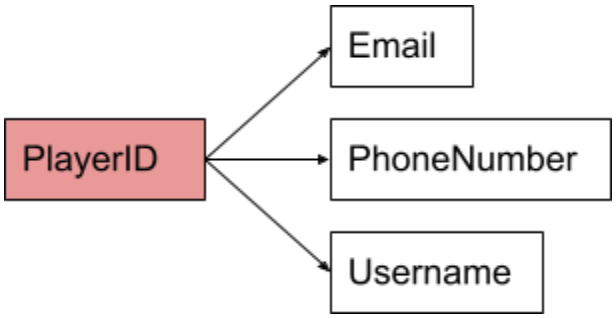
Team(TeamID, LeagueID, CompetitionAdminID, TeamName, Points, Wins, Losses, Draws, GoalsFor, GoalsAgainst, Venue, GoalDifference)	
Functional Dependencies	{TeamID} → LeagueID, CompetitionAdminID, TeamName, Points, Wins, Losses, Draws, GoalsFor, GoalsAgainst, Venue, GoalDifference {TeamName} → TeamID, Points, Wins, Losses, Draws, GoalsFor, GoalsAgainst, Venue, GoalDifference {CompetitionAdminID} → LeagueID
Derive 3NF	Non-key attributes (Points, Wins, Losses, Draws, GoalsFor, GoalsAgainst, Venue, GoalDifference) are transitively dependent on candidate key {TeamID}, which violates 3NF. This table can be decomposed into two tables:: 1. **Team**(TeamID, LeagueID, CompetitionAdminID,TeamName) 2. **TeamStats**(TeamID, Points, Wins, Losses, Draws, GoalsFor, GoalsAgainst, Venue, GoalDifference)
Team	 Table is: ● 1NF because all values are atomic ● 2NF because all non-key attributes are fully functionally dependent on the primary key ● 3NF because all non-key attributes are non-transitively dependent on the primary key ● BCNF because **TeamID→ LeagueID, CompetitionAdminID, TeamName** is the only functional dependency in the table, and TeamID is a superkey since it determines all the other attributes in the tuple.
Derive BCNF	Since {CompetitionID} → LeagueID, and CompetitionID is not a superkey, then this violates BCNF.

	This table can be decomposed into two tables: **TeamDetails**(TeamID, TeamName, LeagueID) **LeagueAdminAssignment**(CompetitionAdminID, LeagueID)
TeamStats	 Table is: ● 1NF because all values are atomic ● 2NF because all non-key attributes are fully functionally dependent on the primary key ● 3NF because all non-key attributes are non-transitively dependent on the primary key ● BCNF because **TeamID → Points, Wins, Losses, Draws, GoalsFor, GoalsAgainst, Venue, GoalDifference** is the only functional dependency in the table, and TeamID is a superkey since it determines all the other attributes in the tuple.

6.4 Player

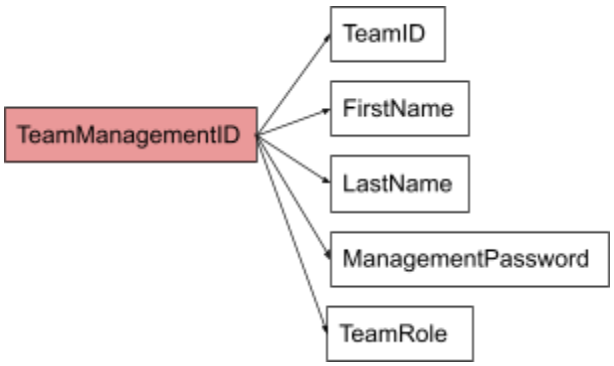
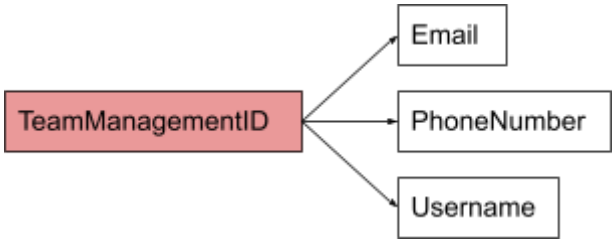
Player(PlayerID, TeamID, FirstName, LastName, Email, PhoneNumber, Username, PlayerPassword, PlayerPosition)	
Functional Dependencies	PlayerID → TeamID, FirstName, LastName, Email, PhoneNumber, Username, PlayerPassword, PlayerPosition, PlayerJerseyColor Username → FirstName, LastName, Email, PhoneNumber, PlayerID, PlayerPassword

	PhoneNumber → FirstName, LastName, Email, Username, PlayerID Email → FirstName, LastName, PhoneNumber, Username, PlayerID
Derive 3NF	Non-key attributes (FirstName, LastName, PlayerPassword) are transitively dependent on {PlayerID}, which violates 3NF. This table can be decomposed into two tables:: 1. **Player**(PlayerID, TeamID, FirstName, LastName, PlayerPassword, PlayerPosition) 2. **PlayerContact**(PlayerID, Email, PhoneNumber, Username)
Player	 Table is: ● 1NF because all values are atomic ● 2NF because all non-key attributes are fully functionally dependent on the primary key ● 3NF because all non-key attributes are non-transitively dependent on the primary key ● BCNF because **PlayerID → TeamID, FirstName, LastName, PlayerPassword, PlayerPosition** is the only functional dependency in the table, and PlayerID is a superkey since it determines all the other attributes in the tuple.

PlayerContact	 Table is: ● 1NF because all values are atomic ● 2NF because all non-key attributes are fully functionally dependent on the primary key ● 3NF because all non-key attributes are non-transitively dependent on the primary key ● BCNF because **PlayerID → Email, PhoneNumber, Username** is the only functional dependency in the table, and PlayerID is a superkey since it determines all the other attributes in the tuple.

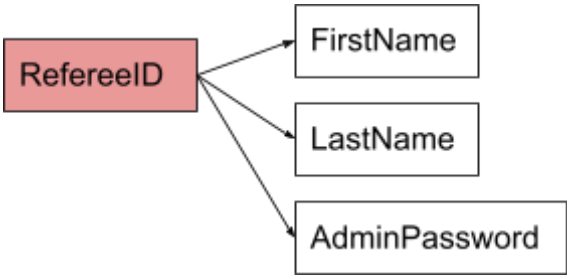
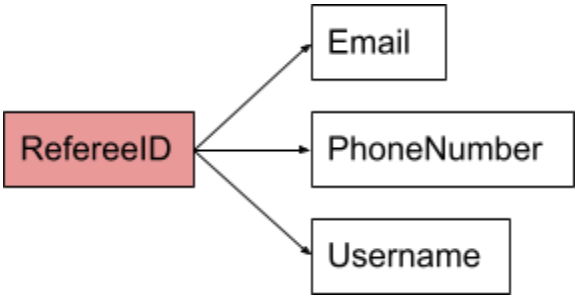
6.5 Team Management

TeamManagement(TeamManagementID, TeamID, FirstName, LastName, Email, PhoneNumber, Username, ManagementPassword, TeamRole)	
Functional Dependencies	{TeamManagementID} → TeamID, FirstName, LastName, Email, PhoneNumber, Username, ManagementPassword, TeamRole {Username} → FirstName, LastName, Email, PhoneNumber, TeamManagementID, ManagementPassword {PhoneNumber} → FirstName, LastName, Email, Username, TeamManagementID, ManagementPassword {Email} → FirstName, LastName, Username, ManagementPassword,TeamManagementID, PhoneNumber
Derive 3NF	Non-key attributes (FirstName, LastName, ManagementPassword, TeamRole) are transitively dependent on {TeamManagementID}, which violates 3NF. This table can be decomposed into two tables:: 1. **TeamManagement**(TeamManagementID, TeamID, FirstName, LastName, ManagementPassword, TeamRole) 2. **TeamManagementContact**(TeamManagementID, Email, PhoneNumber, Username)

TeamManagement	 Table is: ● 1NF because all values are atomic ● 2NF because all non-key attributes are fully functionally dependent on the primary key ● 3NF because all non-key attributes are non-transitively dependent on the primary key ● BCNF because **TeamManagementID → TeamID, FirstName, LastName, ManagementPassword, TeamRole** is the only functional dependency in the table, and TeamManagementID is a superkey since it determines all the other attributes in the table.
TeamManagementContact	 Table is: ● 1NF because all values are atomic ● 2NF because all non-key attributes are fully functionally dependent on the primary key ● 3NF because all non-key attributes are non-transitively dependent on the primary key ● BCNF because **TeamManagementID → Email, PhoneNumber, Username** is the only functional dependency in the table, and TeamManagementID is a superkey since it determines all the other attributes in the tuple.

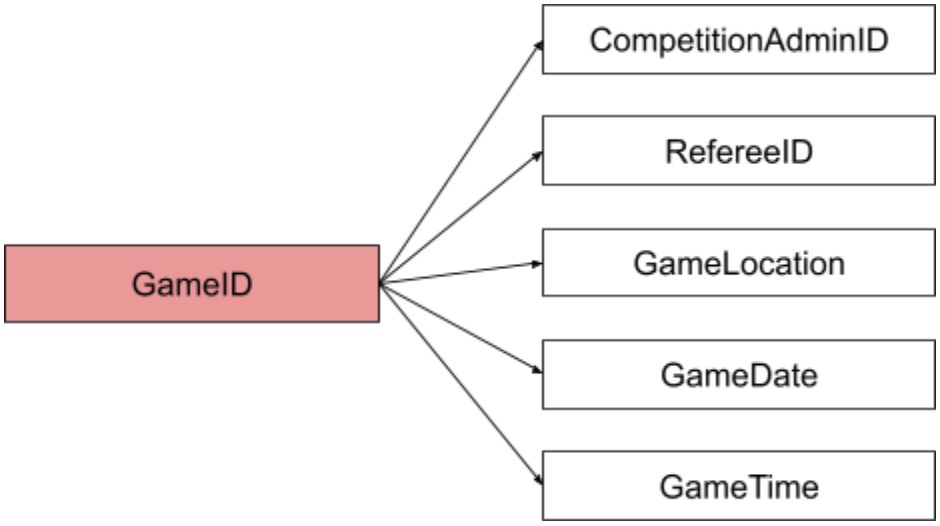
6.6 Referee

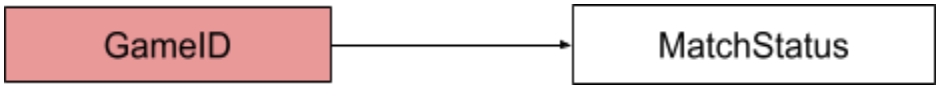
Referee(RefereeID, FirstName, LastName, Email, PhoneNumber, Username, RefereePassword)	
Functional Dependencies	{RefereeID} → FirstName, LastName, Email, PhoneNumber, Username, RefereePassword

	{Username} → FirstName, LastName, Email, PhoneNumber, RefereeID, RefereePassword {PhoneNumber} → FirstName, LastName, Email, Username, RefereeID, RefereePassword {Email} → FirstName, LastName, PhoneNumber, Username, RefereeID, RefereePassword
Derive 3NF	Non-key attributes (FirstName, LastName, RefereePassword) are transitively dependent on {RefereeID}, which violates 3NF. This table can be decomposed into two tables:: 1. **Referee**(RefereeID, FirstName, LastName, RefereePassword) 2. **RefereeContact**(RefereeID, Email, PhoneNumber, Username)
Referee	 Table is: ● 1NF because all values are atomic ● 2NF because all non-key attributes are fully functionally dependent on the primary key ● 3NF because all non-key attributes are non-transitively dependent on the primary key ● BCNF because **RefereeID → FirstName, LastName, AdminPassword** is the only functional dependency in the table, and RefereeID is a superkey since it determines all the other attributes in the tuple.
RefereeContact	 Table is: ● 1NF because all values are atomic

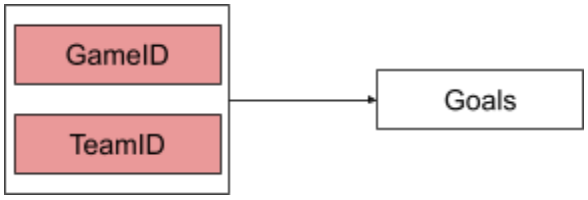
	• 2NF because all non-key attributes are fully functionally dependent on the primary key • 3NF because all non-key attributes are non-transitively dependent on the primary key • BCNF because **RefereeID → Email, PhoneNumber, Username** is the only functional dependency in the table, and RefereeID is a superkey since it determines all the other attributes in the tuple.

6.7 Game

Game(GameID, CompetitionAdminID, RefereeID, GameLocation, GameDate, GameTime, MatchStatus)	
Functional Dependencies	GameID → CompetitionAdminID, RefereeID, GameLocation, GameDate, GameTime, MatchStatus GameID, GameDate, GameTime → MatchStatus
Derive 2NF	There are partial dependencies: • The first functional dependency is a full dependency since GameID is the primary key and it determines all other attributes. • The second dependency is partial since GameID alone can already determine MatchStatus, which violates 2NF. To satisfy the 2NF conditions, the following 2 tables need to be created: **Game**(GameID, CompetitionAdminID, RefereeID, GameLocation, GameDate, GameTime) **GameStatus**(GameID, MatchStatus)
Game	 Table is: • 1NF because all values are atomic • 2NF because all non-key attributes are fully functionally dependent on the primary key

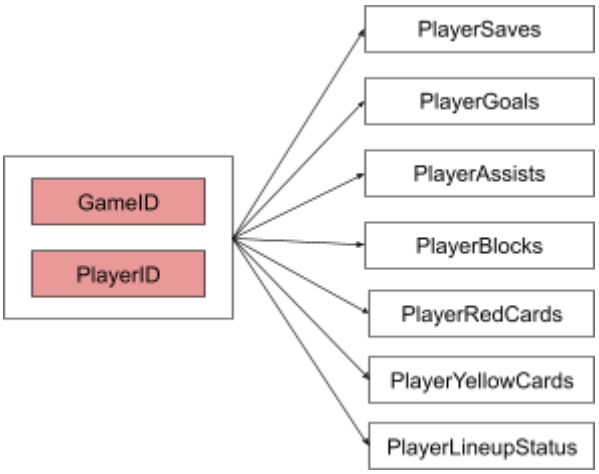
	• 3NF because all non-key attributes are non-transitively dependent on the primary key • BCNF because **GameID → CompetitionAdminID, RefereeID, GameLocation, GameDate, GameTime** is the only functional dependency in the table, and GameID is a superkey since it determines all the other attributes in the tuple.
GameStatus	 Table is: • 1NF because all values are atomic • 2NF because all non-key attributes are fully functionally dependent on the primary key • 3NF because all non-key attributes are non-transitively dependent on the primary key • BCNF because **GameID → MatchStatus** is the only functional dependency in the table, and GameID is a superkey since it determines all the other attributes in the tuple.

6.8 GameTeamStats

GameTeamStats(GameID, TeamID, Goals)	
Functional Dependencies	GameID, TeamID → Goals
GameTeamStats	 Table is: • 1NF because all values are atomic • 2NF because all non-key attributes are fully functionally dependent on the composite primary key • 3NF because all non-key attributes are non-transitively dependent on the composite primary key • BCNF because **{GameID, TeamID}→ Goals** is the only functional dependency in the table, and {GameID, TeamID} is a superkey since it determines all the other attributes in the tuple.

6.9 GamePlayerStats

GamePlayerStats (GameID, PlayerID, PlayerSaves, PlayerGoals, PlayerAssists, PlayerBlocks, PlayerRedCards, PlayerYellowCards, PlayerLineupStatus)

Functional Dependencies	GameID, PlayerID → PlayerSaves, PlayerGoals, PlayerAssists, PlayerBlocks, PlayerRedCards, PlayerYellowCards, PlayerLineupStatus
GamePlayerStats	 Table is: ● 1NF because all values are atomic ● 2NF because all non-key attributes are fully functionally dependent on the composite primary key ● 3NF because all non-key attributes are non-transitively dependent on the composite primary key ● BCNF because **{GameID, PlayerID}→ PlayerSaves, PlayerGoals, PlayerAssists, PlayerBlocks, PlayerRedCards, PlayerYellowCards, PlayerLineupStatus** is the only functional dependency in the table, and {GameID, PlayerID} is a superkey since it determines all the other attributes in the tuple.

6.10 Bernstein's Algorithm (Bonus Marks)

Player Table to be made into 3NF

Player(PlayerID, TeamID, FirstName, LastName, Email, PhoneNumber, Username, PlayerPassword, PlayerPosition, PlayerJerseyColor)	
Functional Dependencies	PlayerID → TeamID, FirstName, LastName, Email, PhoneNumber, Username, PlayerPassword, PlayerPosition, PlayerJerseyColor Username → FirstName, LastName, Email, PhoneNumber, PlayerID, PlayerPassword PhoneNumber → FirstName, LastName, Email, Username, PlayerID Email → FirstName, LastName, PhoneNumber, Username, PlayerID TeamID → PlayerJerseyColor

Using Bernstein's Algorithm

Player(PlayerID, TeamID, FirstName, LastName, Email, PhoneNumber, Username, PlayerPassword, PlayerPosition, PlayerJerseyCol

R(A, B, C, D, E, F, G, H, I, J)

Functional dependencies: A → B, C, D, E, F, G, H, I, J
G → C, D, E, F, A, H
F → C, D, E, G, A
E → C, D, F, G, A
B → J

① Find Redundancies

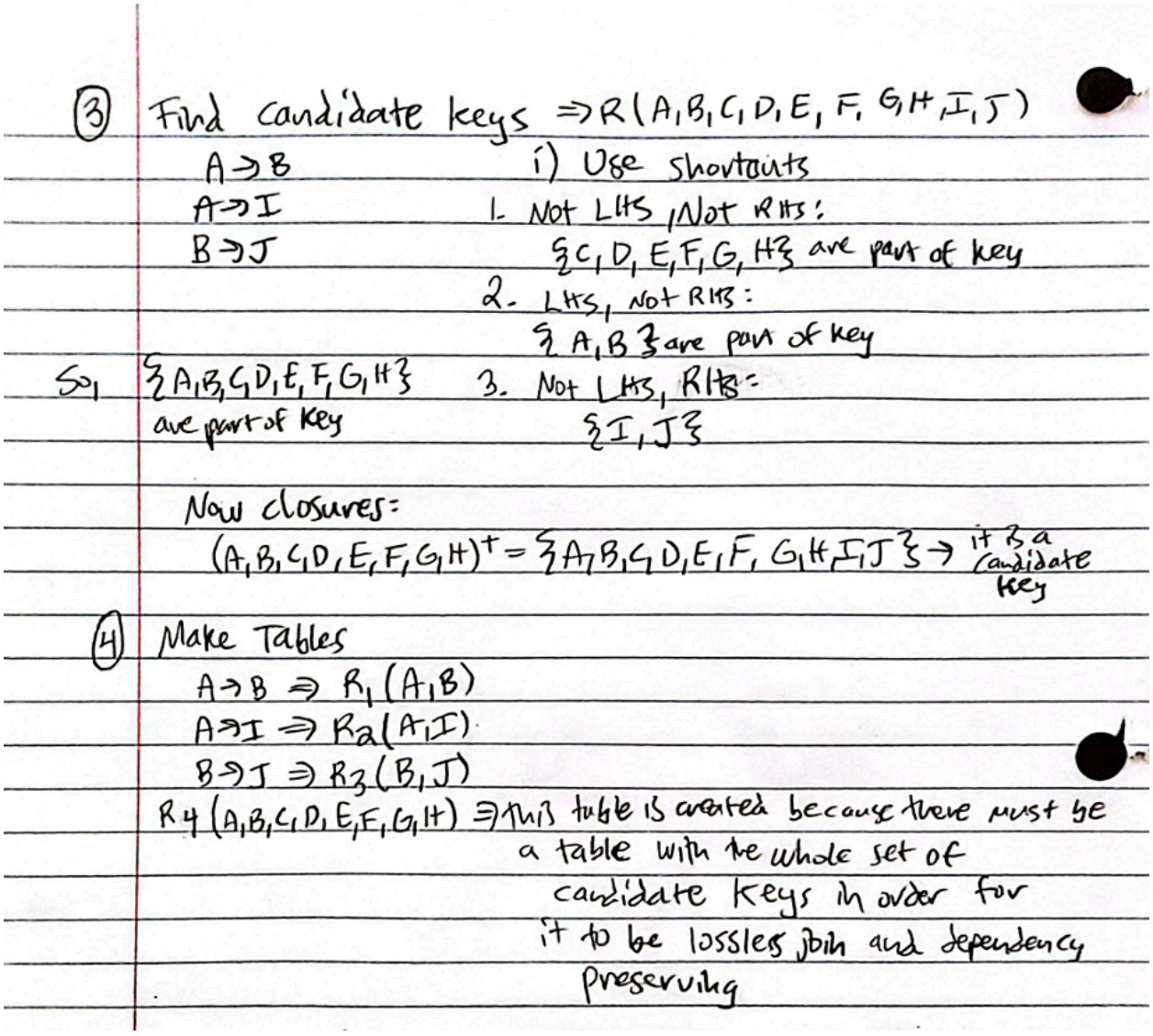
A → B: $A^+ = \{A, C, D, E, F, G, H, I, J\}$ We don't get B, so not redundant
↳ same logic goes for A → I.
for all other letters dependent on A, they appear again, so they are redundant, for example:
A → C: $A^+ = \{A, B, D, E, F, G, H, I, J, C\}$ we got C, so this FD is redundant

G → C: $G^+ = \{D, E, F, A, H, G, C\}$ we got C, so redundant
↳ all G → letter are redundant.
- all F → letter are also redundant
- all E → letter are also redundant

B → J: $B^+ = \{B\}$ we don't get J, so not redundant

After removing redundancies, FD = {A → B, A → I, B → J}

② No partial dependencies since there are no composite keys



The tables that are derived to make it 3NF:

R1(PlayerID, TeamID)

R2(PlayerID, PlayerPosition)

R3(TeamID, PlayerJerseyColor)

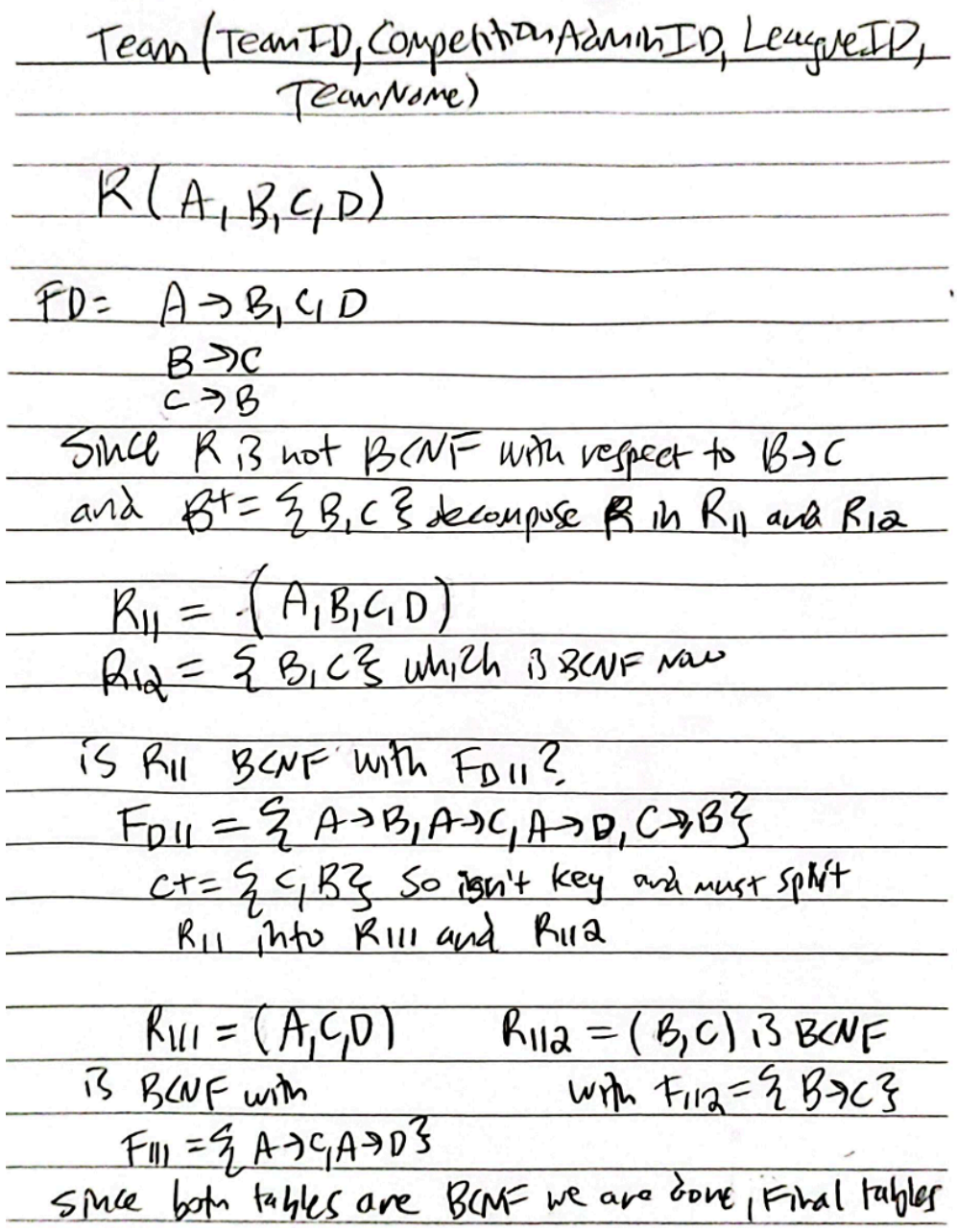
R4(PlayerID, TeamID, FirstName, LastName, Email, PhoneNumber, Username, PlayerPassword)

6.11 Decomposition Algorithm (Bonus Marks)

Making Referee Table into BCNF

Team(TeamID, LeagueID, CompetitionAdminID,TeamName)	
Functional Dependencies	TeamID → LeagueID, CompetitionAdminID, TeamName LeagueID → CompetitionAdminID CompetitionAdminID → LeagueID

Using BCNF Decomposition Algorithm



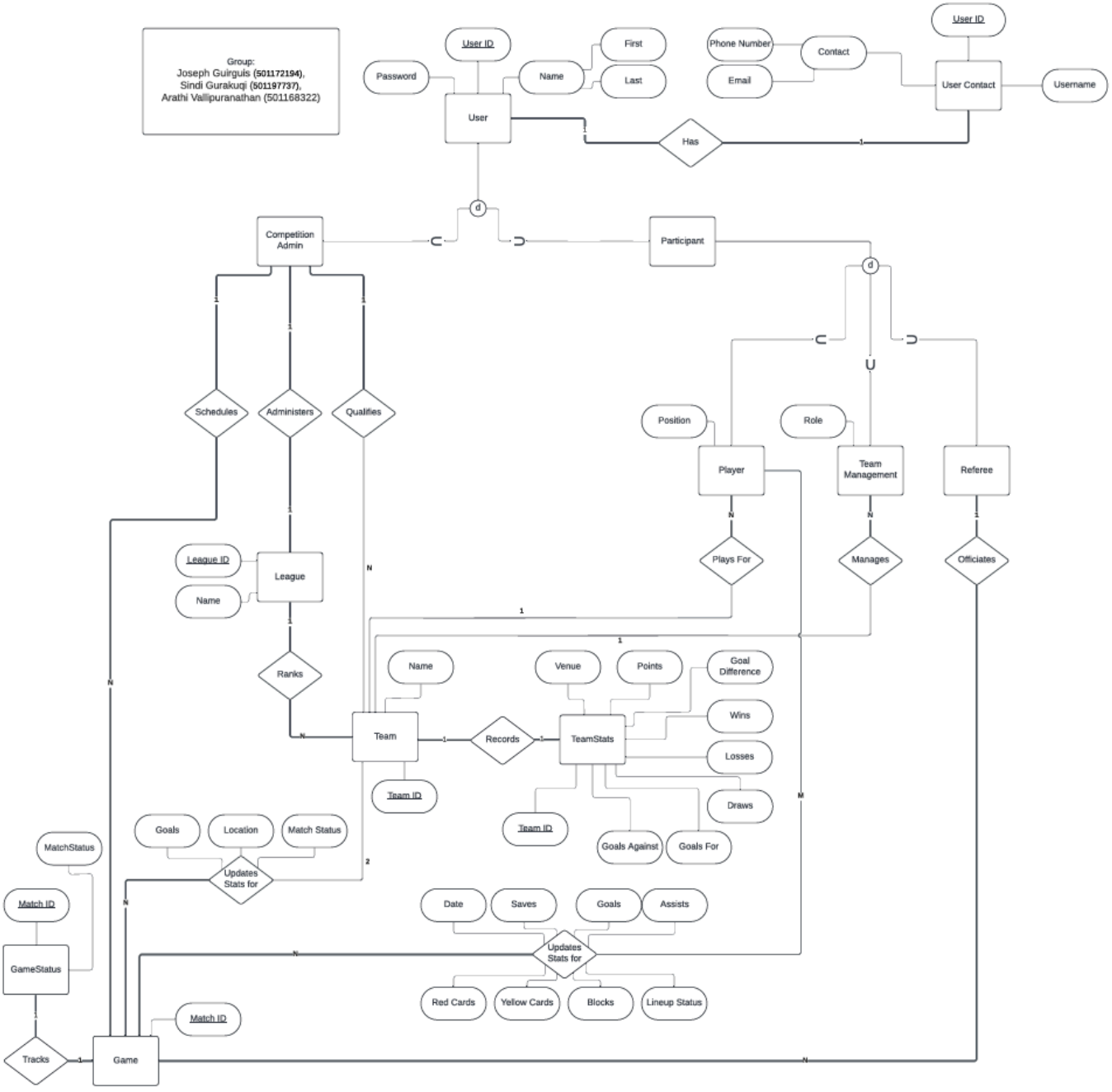
The Final tables to be derived to make it BCNF:

R1(TeamID, LeagueID, TeamName)

R2(CompetitionAdminID, LeagueID)

7 Normalized ER Model

7.1 Normalized Entity Relationship Diagram



7.2 Normalized Database Schema

7.2.1 - Create LeagueAdmin Table

```
CREATE TABLE LeagueAdmin (
   CompetitionAdminID NUMBER PRIMARY KEY,
   FirstName VARCHAR2(25),
   LastName VARCHAR2(25),
   AdminPassword VARCHAR2(255) NOT NULL
);
```

7.2.2 - Create LeagueAdminContact Table

```
CREATE TABLE LeagueAdminContact (
   CompetitionAdminID NUMBER PRIMARY KEY,
   Email VARCHAR2(50) UNIQUE,
   PhoneNumber VARCHAR2(20) UNIQUE,
   Username VARCHAR2(24) NOT NULL UNIQUE
);
```

7.2.3 - Create League Table

```
CREATE TABLE League (
   LeagueID NUMBER PRIMARY KEY,
   LeagueName VARCHAR2(30) NOT NULL,
   CompetitionAdminID NUMBER,
   CONSTRAINT fk_admin_for_league FOREIGN KEY (CompetitionAdminID)
   REFERENCES LeagueAdminContact(CompetitionAdminID)
);
```

7.2.4 - Create Team Table

```
CREATE TABLE Team (
   TeamID NUMBER PRIMARY KEY,
   LeagueID NUMBER NOT NULL,
   CompetitionAdminID NUMBER NOT NULL,
   TeamName VARCHAR2(50) NOT NULL,
   CONSTRAINT fk_league_for_team FOREIGN KEY (LeagueID)
   REFERENCES League(LeagueID),
   CONSTRAINT fk_admin_for_team FOREIGN KEY (CompetitionAdminID)
   REFERENCES LeagueAdminContact(CompetitionAdminID)
```

```
);
```

7.2.5 - Create TeamStatus Table

```
CREATE TABLE TeamStats (
   TeamID NUMBER PRIMARY KEY,
   Points NUMBER DEFAULT 0,
   Wins NUMBER DEFAULT 0,
   Losses NUMBER DEFAULT 0,
   Draws NUMBER DEFAULT 0,
   GoalsFor NUMBER DEFAULT 0,
   GoalsAgainst NUMBER DEFAULT 0,
   Venue VARCHAR2(50),
   GoalDifference NUMBER,
   CHECK (GoalsFor >= 0),
   CHECK (GoalsAgainst >= 0)
);
```

7.2.6 - Create Player Table

```
CREATE TABLE Player (
   PlayerID NUMBER PRIMARY KEY,
   TeamID NUMBER NOT NULL,
   FirstName VARCHAR2(25) NOT NULL,
   LastName VARCHAR2(25) NOT NULL,
   PlayerPassword VARCHAR2(255) NOT NULL,
   PlayerPosition VARCHAR2(15),
   CHECK (PlayerPosition IN ('Goalkeeper', 'Defender', 'Midfielder', 'Forward')),
   CONSTRAINT fk_team_for_player FOREIGN KEY (TeamID) REFERENCES
Team(TeamID)
);
```

7.2.7 - Create PlayerContact Table

```
CREATE TABLE PlayerContact (
   PlayerID NUMBER PRIMARY KEY,
   Email VARCHAR2(50) UNIQUE,
   PhoneNumber VARCHAR2(20) UNIQUE,
   Username VARCHAR2(24) NOT NULL UNIQUE
```

```
);
```

7.2.8 - Create TeamManagement Table

```
CREATE TABLE TeamManagement (
   TeamManagementID NUMBER PRIMARY KEY,
   TeamID NUMBER NOT NULL,
   FirstName VARCHAR2(25) NOT NULL,
   LastName VARCHAR2(25) NOT NULL,
   ManagementPassword VARCHAR2(255) NOT NULL,
   TeamRole VARCHAR2(15),
   CONSTRAINT fk_team_management_for_team FOREIGN KEY (TeamID) REFERENCES
Team(TeamID)
);
```

7.2.9 - Create TeamManagementContact Table

```
CREATE TABLE TeamManagementContact (
   TeamManagementID NUMBER PRIMARY KEY,
   Email VARCHAR2(50) UNIQUE,
   PhoneNumber VARCHAR2(20) UNIQUE,
   Username VARCHAR2(24) NOT NULL UNIQUE
);
```

7.2.10 - Create Referee Table

```
CREATE TABLE Referee (
   RefereeID NUMBER PRIMARY KEY,
   FirstName VARCHAR2(25) NOT NULL,
   LastName VARCHAR2(25) NOT NULL,
   RefereePassword VARCHAR2(255) NOT NULL
);
```

7.2.11 - Create RefereeContact Table

```
CREATE TABLE RefereeContact (
   RefereeID NUMBER PRIMARY KEY,
   Email VARCHAR2(50) NOT NULL UNIQUE,
   PhoneNumber VARCHAR2(20) UNIQUE,
```

```
    Username VARCHAR2(24) NOT NULL UNIQUE
);
```

7.2.12 - Create Game Table

```
CREATE TABLE Game (
    GameID NUMBER PRIMARY KEY,
    CompetitionAdminID NUMBER NOT NULL,
    RefereeID NUMBER NOT NULL,
    GameLocation VARCHAR2(50) NOT NULL,
    GameDate DATE NOT NULL,
    GameTime TIMESTAMP,
    CONSTRAINT fk_admin_for_game FOREIGN KEY (CompetitionAdminID)
        REFERENCES LeagueAdmin(CompetitionAdminID),
    CONSTRAINT fk_referee_for_game FOREIGN KEY (RefereeID)
        REFERENCES Referee(RefereeID)
);
```

7.2.13 - Create GameStatus Table

```
CREATE TABLE GameStatus (
    GameID NUMBER PRIMARY KEY,
    MatchStatus VARCHAR2(10) DEFAULT 'SCHEDULED',
    CHECK (MatchStatus IN ('SCHEDULED', 'COMPLETED', 'CANCELLED','DELAYED'))
);
```

7.2.14 - Create GameTeamStatus Table

```
CREATE TABLE GameTeamStats (
    GameID NUMBER NOT NULL,
    TeamID NUMBER NOT NULL,
    Goals NUMBER DEFAULT 0 NOT NULL,
    PRIMARY KEY (GameID, TeamID),
    CONSTRAINT fk_game_for_gameTeamStats FOREIGN KEY (GameID) REFERENCES
Game(GameID),
    CONSTRAINT fk_team_for_gameTeamStats FOREIGN KEY (TeamID) REFERENCES
Team(TeamID)
);
```

7.2.15 - Create GamePlayerStats Table

```
CREATE TABLE GamePlayerStats (
   GameID NUMBER NOT NULL,
   PlayerID NUMBER NOT NULL,
   PlayerSaves NUMBER DEFAULT 0,
   PlayerGoals NUMBER DEFAULT 0,
   PlayerAssists NUMBER DEFAULT 0,
   PlayerBlocks NUMBER DEFAULT 0,
   PlayerRedCards NUMBER DEFAULT 0,
   PlayerYellowCards NUMBER DEFAULT 0,
   PlayerLineupStatus VARCHAR2(12) DEFAULT 'RESERVES',
   PRIMARY KEY (GameID, PlayerID),
   CONSTRAINT fk_game_for_gamePlayerStats FOREIGN KEY (GameID) REFERENCES
Game(GameID),
   CONSTRAINT fk_player_for_gamePlayerStats FOREIGN KEY (PlayerID) REFERENCES
Player(PlayerID),
   CHECK (PlayerLineupStatus IN ('FIRSTTEAM', 'SUBSTITUTE', 'RESERVES'))
);
```

7.3 Queries Performed On Normalized Database

The queries performed are exactly the same as before the database schema was not normalized.

8 Python UI Application

8.1 Description of UI

The database application features a graphical user interface implemented using the “tkinter” library in Python. This library is used to create key UI components such as menus, buttons, text fields, and dialog boxes. For database operations, the ‘cx_Oracle’ library is used to establish a connection between the UI and OracleSQL database. This allows for dynamic data fetching, deletion, insertion, updates, and querying directly through the UI.

8.2 UI Features

1. **Log in and log out**
    a. Description: Users can securely log in and out of the DBMS by providing valid credentials.
2. **Dashboard**
    a. Description: Serves as main navigation to provide quick access to different pages.
3. **Create tables**
    a. Description: Users can create tables either individually or all at once. Dependency constraints are handled appropriately when creating individual tables.
4. **Drop tables**
    a. Description: Users can drop tables either individually or all at once. Integrity constraints are dropped when dropping individual tables.
5. **Populate tables**
    a. Description: Users can populate tables with data either by using dummy data or by inserting custom entries.
6. **Query data**
    a. Description: Users can fetch and view data from tables using custom queries.

8.3 Screenshots of UI

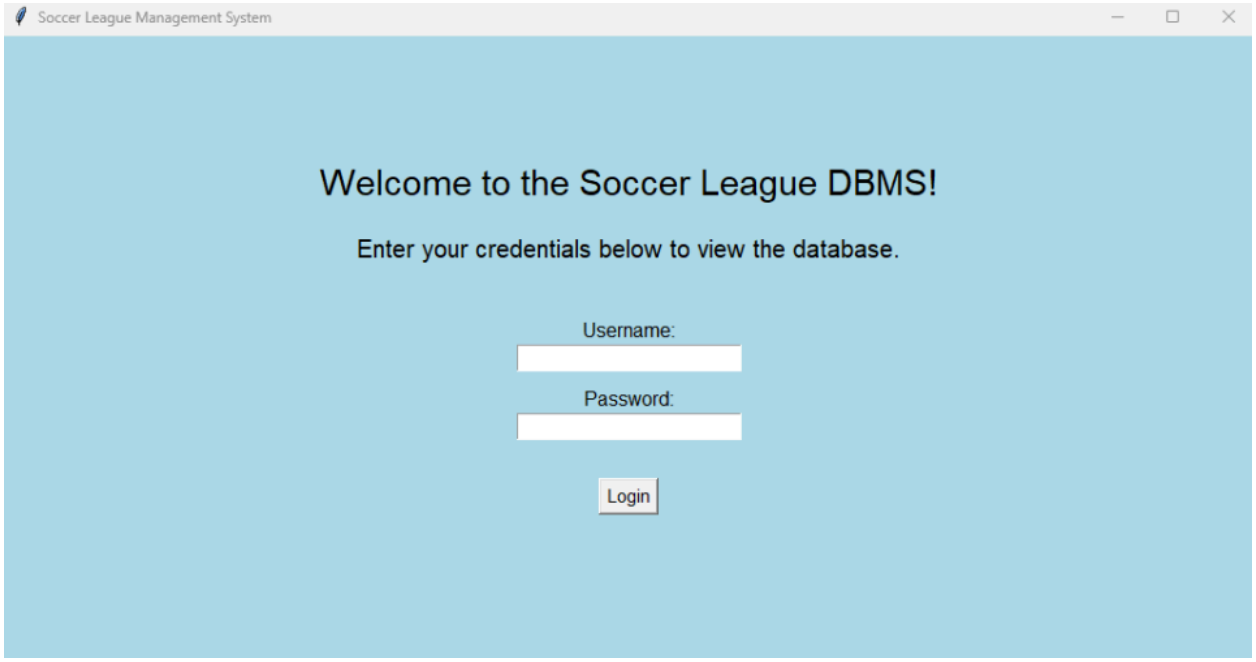


Image 1: Login screen

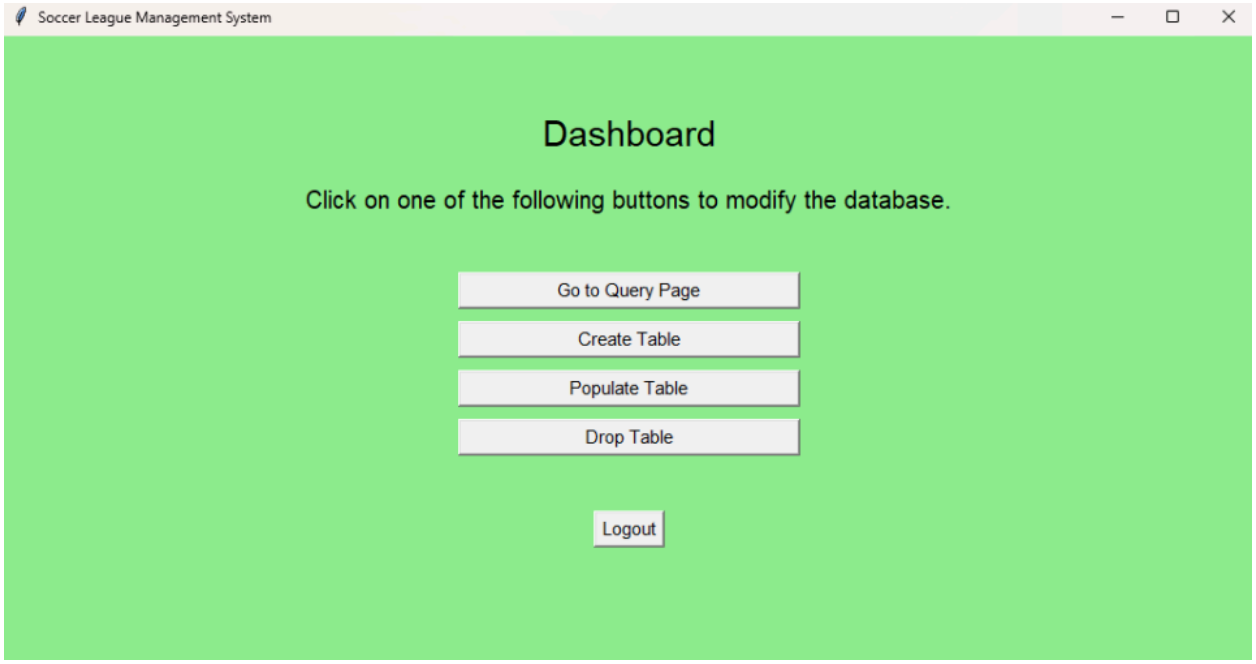


Image 2: Dashboard

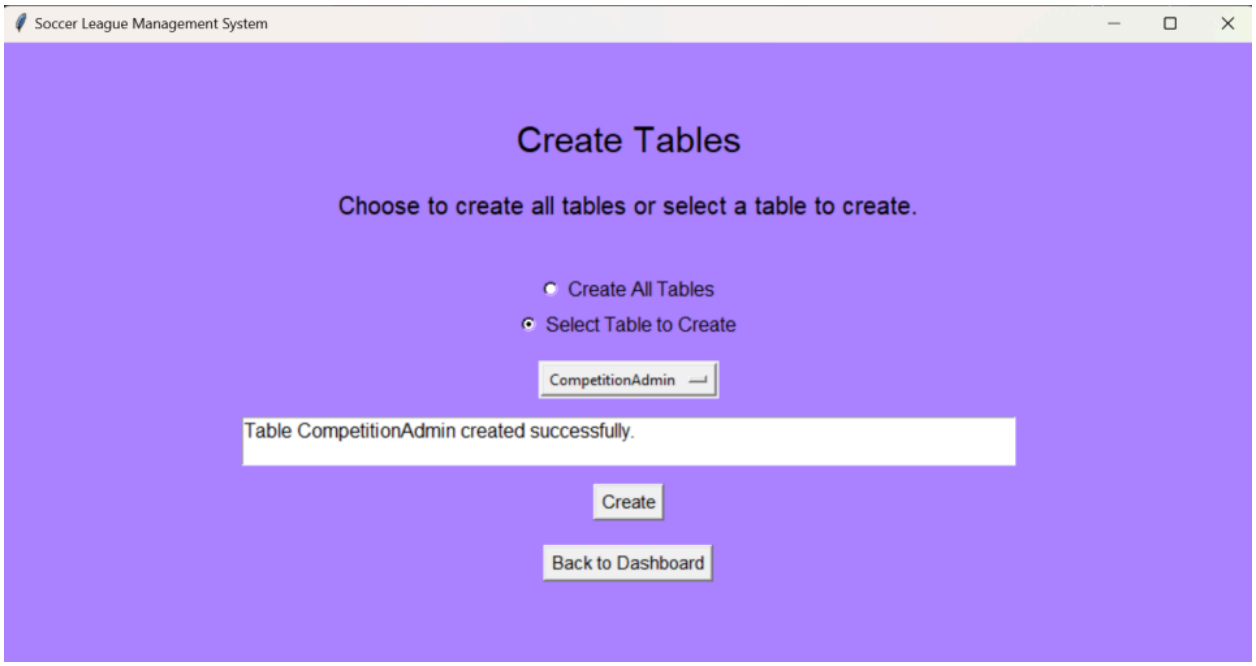


Image 3: Successfully creating individual 'CompetitionAdmin' table

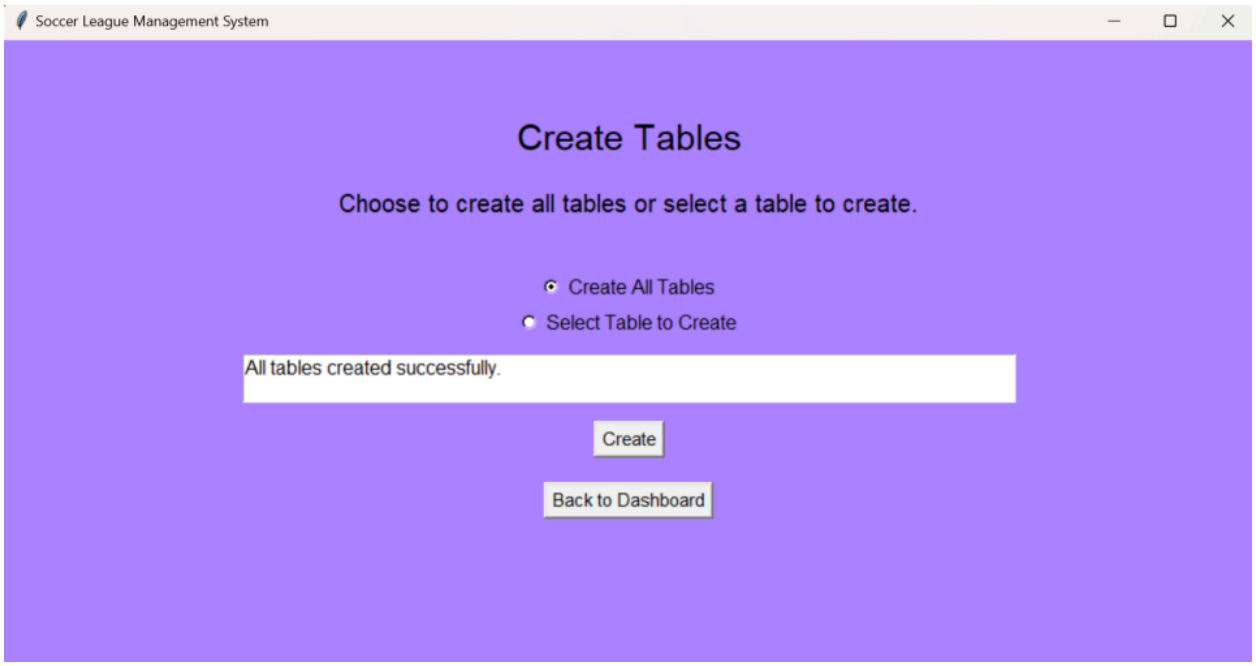


Image 4: Successfully creating all tables

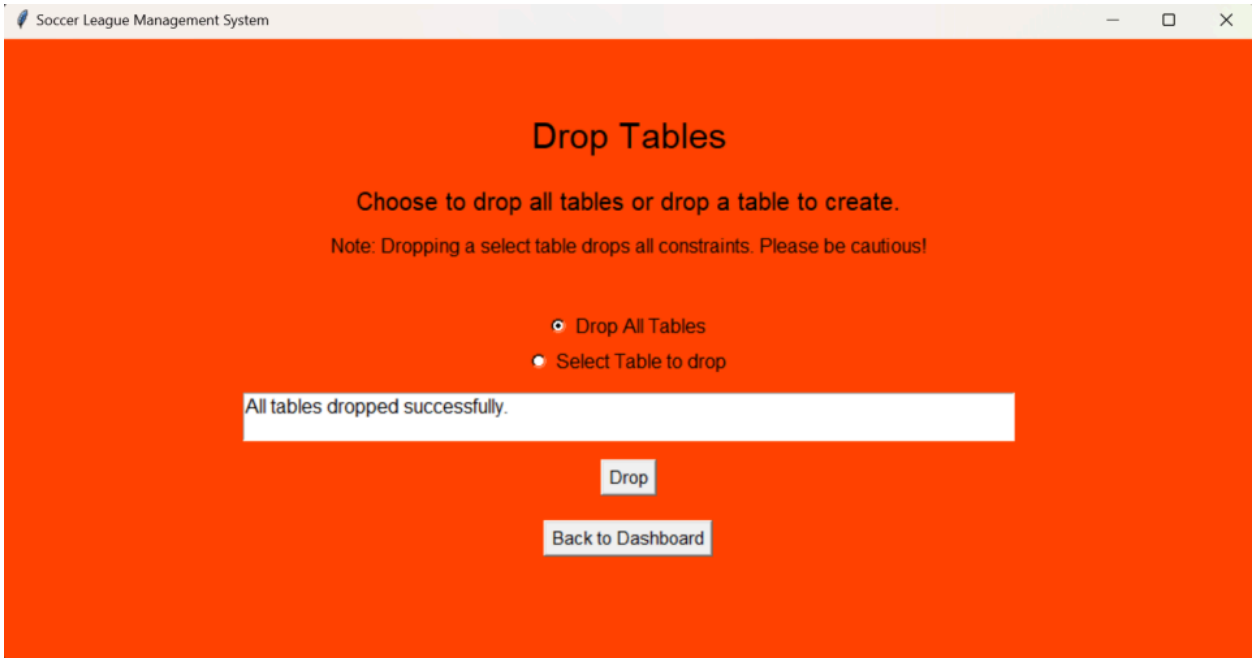


Image 5: Successfully dropping all tables

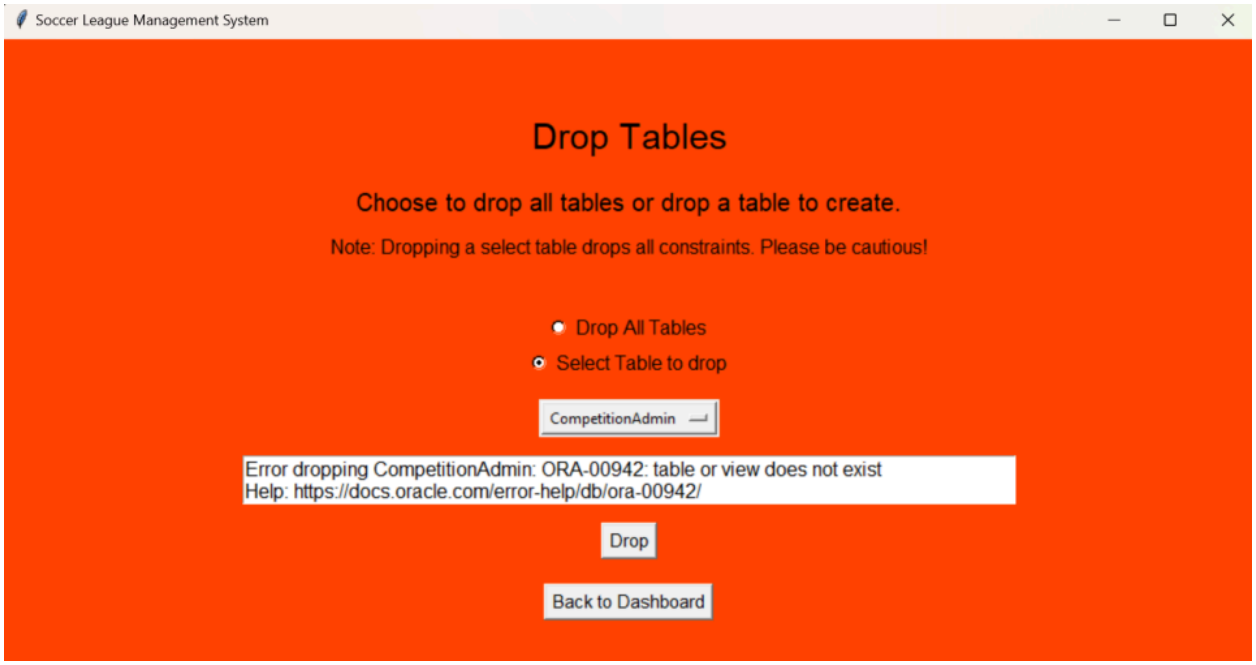


Image 6: Unsuccessfully dropping individual ‘CompetitionAdmin’ table as it does not exist.

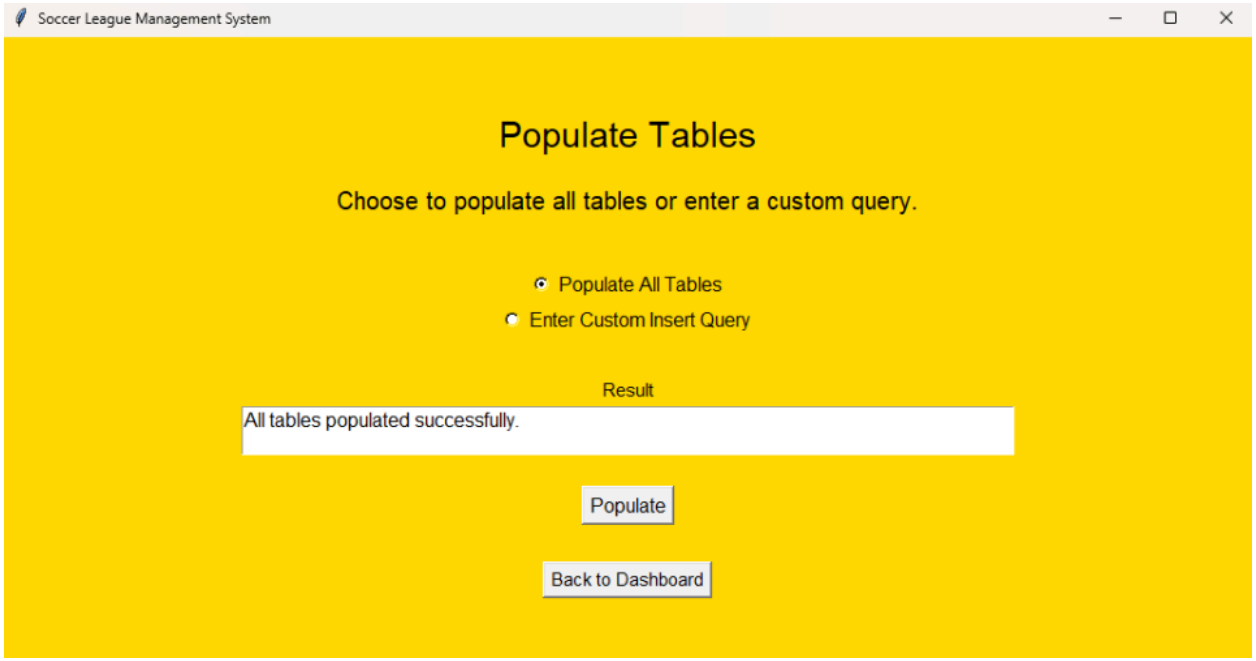


Image 7: Successfully populating all tables with dummy data

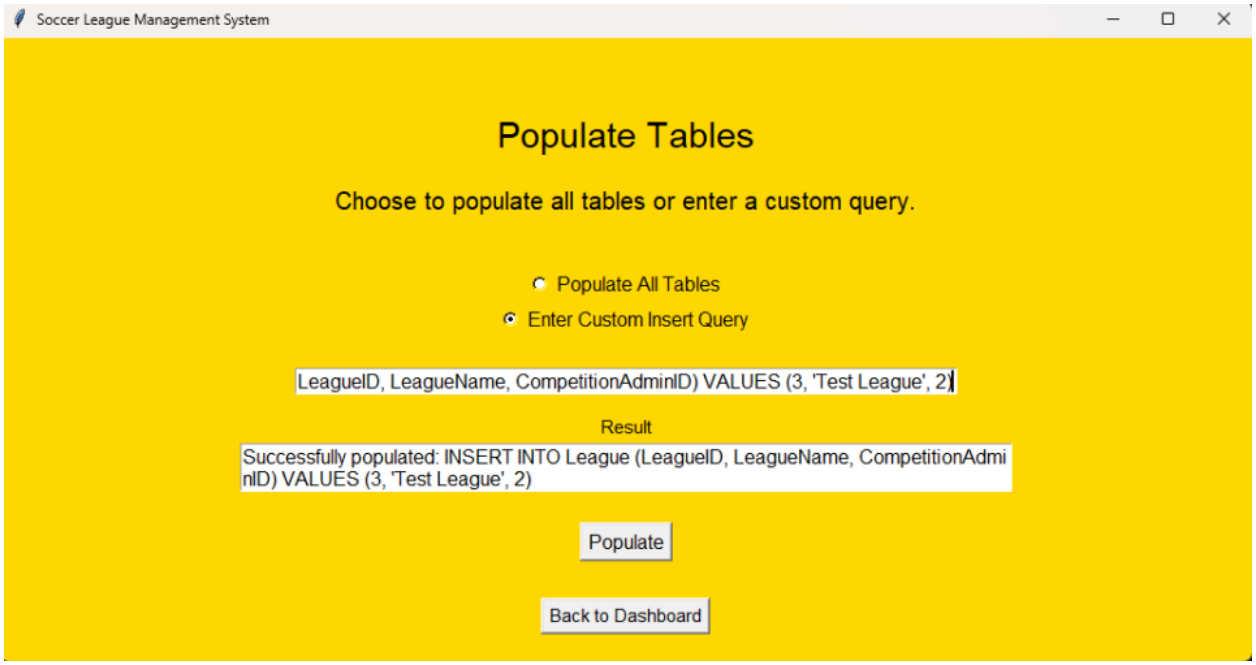


Image 8: Successfully populating ‘League’ table with custom insert entry.

Image 9: Successfully displaying all data satisfying custom query

9 Relational Algebra Notation

9.1 Query 1

$$\pi_{FirstName, LastName, PlayerPosition}(\sigma_{username = 'coach\ sindi'}(TeamManagementContact)\ ,\ Player \bowtie TeamManagement\ )$$

9.2 Query 2

findLeague ← $\pi_{LeagueID}(\sigma_{username = 'joe027'}(LeagueAdminContact) \bowtie League)$

TeamsStats ←
$\pi_{TeamName, Wins, Draws, Losses, GoalsFor, GoalsAgainst, GoalDifference, Points}(\sigma_{LeagueID\ = findLeague}(Team),\ TeamStats\ )$

Result ← $\tau_{Points\ (DESC), GoalDifference(DESC), GoalsFor(DESC)}$(TeamsStats)

9.3 Query 3

Joins ← $Player \bowtie\ GamePlayerStats\ ,\ Player \bowtie Team,\ League \bowtie Team$
Aggregate ←
$PlayerID, FirstName,\ LastName, PlayerPosition, TeamName, LeagueID\ F_{\rho_{TotalGoals}(SUM\ PlayerGoals)}(Joins)$

Result ← $\tau_{TotalGoals\ \ (DESC)}$(Aggregate)

9.4 Query 4

Group_by ← cc.Username,l.LeagueName,r.FirstName,r.LastName,l.LeagueID
k1 ← (g ⋈(g.GameID = gts1.GameID) gts1)
k2 ← (k1 ⋈(g.GameID = gts2.GameID) gts2)
k3 ← (k2 ⋈(gts1.TeamID = t1.TeamID) t1)
k4 ← (k3 ⋈(gts2.TeamID = t2.TeamID) t2)
k5 ← (k4 ⋈(g.CompetitionAdminID = l.CompetitionAdminID) l)
k6 ← (k5 ⋈(l.CompetitionAdminID = c.CompetitionAdminID) c)
k7 ← (k6 ⋈(g.GameID = gs.GameID) gs)
k8 ← σ(gts1.TeamID ≠ gts2.TeamID) (k7)
k9 ← σ(t1.TeamName ≠ Group_by F LEAST(t1.TeamName, t2.TeamName)) (k8)

k10 ← π(F LEAST(t1.TeamName, t2.TeamName), Group_by F GREATEST(t1.TeamName, t2.TeamName), gts1.Goals, gts2.Goals, g.GameDate, g.RefereeID, gs.MatchStatus, l.LeagueName, l.LeagueID) (k9)
Result ← τ(g.GameDate) (k10)

9.5 Query 5

Joins ← LeagueAdmin ⋈ League , Referee ⋈ Game , Game ⋈League, LeagueAdminContact

Aggregate ← Username,LeagueName,FirstName,LastName,LeagueID F $\rho_{TotalGoals}$(COUNT RefereeID) (Joins)

Result ← $\tau_{TotalGoals\ (DESC)}$(Aggregate)

9.6 Query 6

Joins ← Player ⋈ GamePlayerStats, Player ⋈ Team, Team ⋈ League

Aggregate ← PlayerID, FirstName, LastName, TeamID F SUM PlayerYellowCards(Joins)

Result ← $\sigma_{PlayerYellowCards >= 2}$(Aggregate)

9.7 Query 7

Exists ← $\pi_{ALL}(\sigma_{PlayerYellowCards > 0}(GamePlayerStats \bowtie Player,\ Player \bowtie Team))$

Result ← $\pi_{PlayerID, FirstName, LastName, TeamName}(\sigma_{EXISTS}(Player \bowtie Team))$

9.8 Query 8

Scored ← $\pi_{PlayerID, FirstName, LastName}(\sigma_{playerGoals > 0}(Player \bowtie GamePlayerStats))$

GotYellowCard ←
$\pi_{PlayerID, FirstName, LastName}(\sigma_{playerYellowCards > 0}(Player \bowtie GamePlayerStats))$

Result ← Scored - GotYellowCard

9.9 Query 9

scoredMoreThan2 ←

$\pi_{PlayerID, FirstName, LastName, PlayerGoals, PlayerAssists}(\sigma_{playerGoals > 2}(Player \bowtie GamePlayerStats))$

assisstedAtLeast1 ←

$\pi_{PlayerID, FirstName, LastName, PlayerGoals, PlayerAssists}(\sigma_{Assists > 0}(Player \bowtie GamePlayerStats))$

Result ← scoredMoreThan2 U assisstedAtLeast1

9.10 Query 10

Joins ← Player ⋈ GamePlayerStats, GamePlayerStats ⋈ Game, Player ⋈ Team

Aggregate ← PlayerID,FirstName,LastName,PlayerPosition,TeamName F $\rho_{hattricks}$(COUNT PlayerID) (Joins)

Result ← $\tau_{hattricks\ (DESC)}$(Aggregate)

9.11 Query 11

Joins ← Player ⋈ GamelayerStats, Player ⋈ Team, TeamManagement ⋈ Player, TeamManagementContact

Aggregate ← Username,FirstName,LastName,PlayerPosition,TeamName F $\rho_{GoalsPerGame}$(AVG PlayerGoals) (Joins)

Result ← $\tau_{GoalsPerGame\ (DESC)}(\sigma_{PlayerGoals > 0.5}$(Aggregate))