Fantasy Manager Database

Team IX

Final Project Report

Andrew Harrison | ACHarrison1123@gmail.com

Sudhir Ray | raysudhir7733@gmail.com

Table of Contents

1.0 Introduction

1.1 Purpose

The purpose of this document is to review and provide a detailed presentation of our team's Fantasy Soccer Manager Database. This report will outline the creation and management of our team’s database. This document will describe the logical flow of the database by specifying the requirements, detailing the conceptual and logical designs, implementing the SQL code, and, lastly, query execution.

1.2 Overview

Our Fantasy Soccer Management Database offers a user-friendly and simple way to track the users Fantasy Soccer progression. Our team’s current database balances user-friendliness with comprehensive functionality. While the current iteration presents a simplified version of the database, it serves as a solid foundation for an even bigger project.

2.0 Scope

Our team’s client requires a database specifically designed for the Fantasy Soccer Manager application. Our team will develop a database featuring various functioning entities such as players, teams, managers, and more. These entities will have specific interrelationships to facilitate easy-to-use and functional database. Our goal is to create an immersive fantasy soccer experience while ensuring accurate and fair gameplay through real-world statistics.

3.0 Requirement Specifications

The following sections will highlight the specific requirements and business functions essential for our team to develop the Fantasy Soccer Manager database. It will provide details on each entity and outline the business rules associated with each entity and their relationships.

3.1 Business Rules

Each entity must have a corresponding ID that will be used to identify it as a Primary Key.

- A User/Manager must be registered with an email and password.
- A User/Manager can only have one Team in a League.
- A Team must belong to a User/Manager.
- A Team must consist of at least 11 players and no more than 15.
- A Team must consist of at least 2 goalkeepers, 3 defenders, 3 mid-fielders, and 3 forwards. The final 4 players may be any position the User/Manger sees fit.
- A Players Market Value must be determined by Real-World statistics about the Player.
- A Player must belong to a Real-World Team.
- A Player can only be a part of one Team in the League.
- The Transfer Market must update player availability based on the fantasy transfers.
- A User/Manager can claim, trade, or drop players only through the Transfer Market.
- A Transfer Market will close as soon as the first game starts and will re-open 24 hours after the final game.
- A Match must involve exactly 2 Teams, no more no less.
- A Match must be calculated through points generated by Real-World Players statistics.

- A Match's results must be determined no later than one hour after the last game has finalized.
- A Real-World Team must have a roster of Real-World Players.
- Real-World Players may be traded, dropped, or picked up by Real-World Team's, which must be reflected in the Transfer Market.

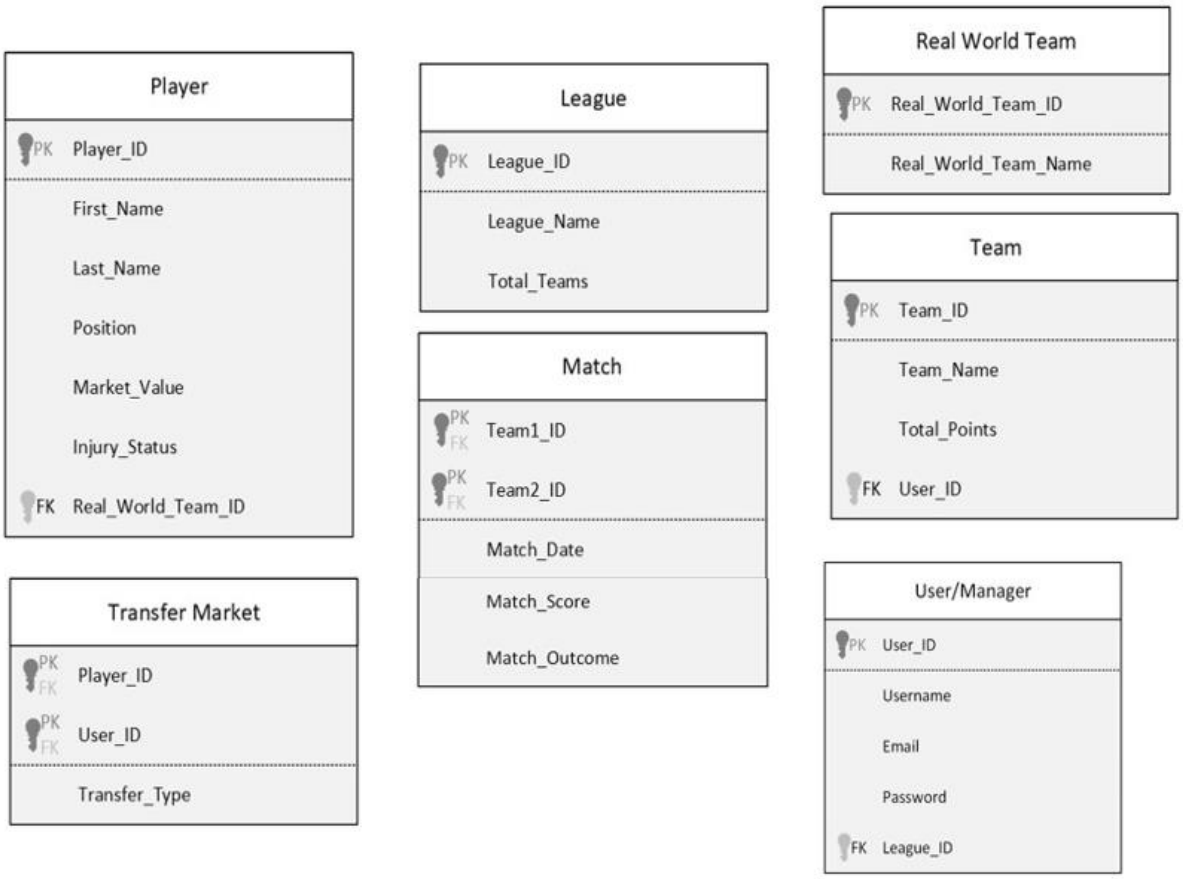
3.2 Entity Relationships

- A League must have one or many Users/Managers. Technically at least 6 Users/Managers.
- A User/Manager can only be associated with one and only one League.
- A User/Manager must be associated with one and only one Team.
- A Team must be owned by one and only one User/Manager.
- A Team must participate in one or many Matches.
- A Match must involve exactly two different Teams.
- A Team must have many Players.
- A Player may be associated with one and only one Team.
- A Player must belong to one Real-World Team.
- A Real-World Team must have many Players.
- A Player may be associated with multiple Transfers.
- A Transfer Market must be associated with exactly one Player.
- A User/Manager may be associated with multiple Transfers.
- A Transfer Market must be associated with exactly one User/Manager.

4.0 Database Design

The following two sections present the conceptual and logical designs for the Fantasy Manager Database. Our team's database comprises seven entities: League, Player, Team, User/Manager, Transfer Market, Match, and Real-World Team. Figure 1.0 will provide a brief look at each entity and their attributes:

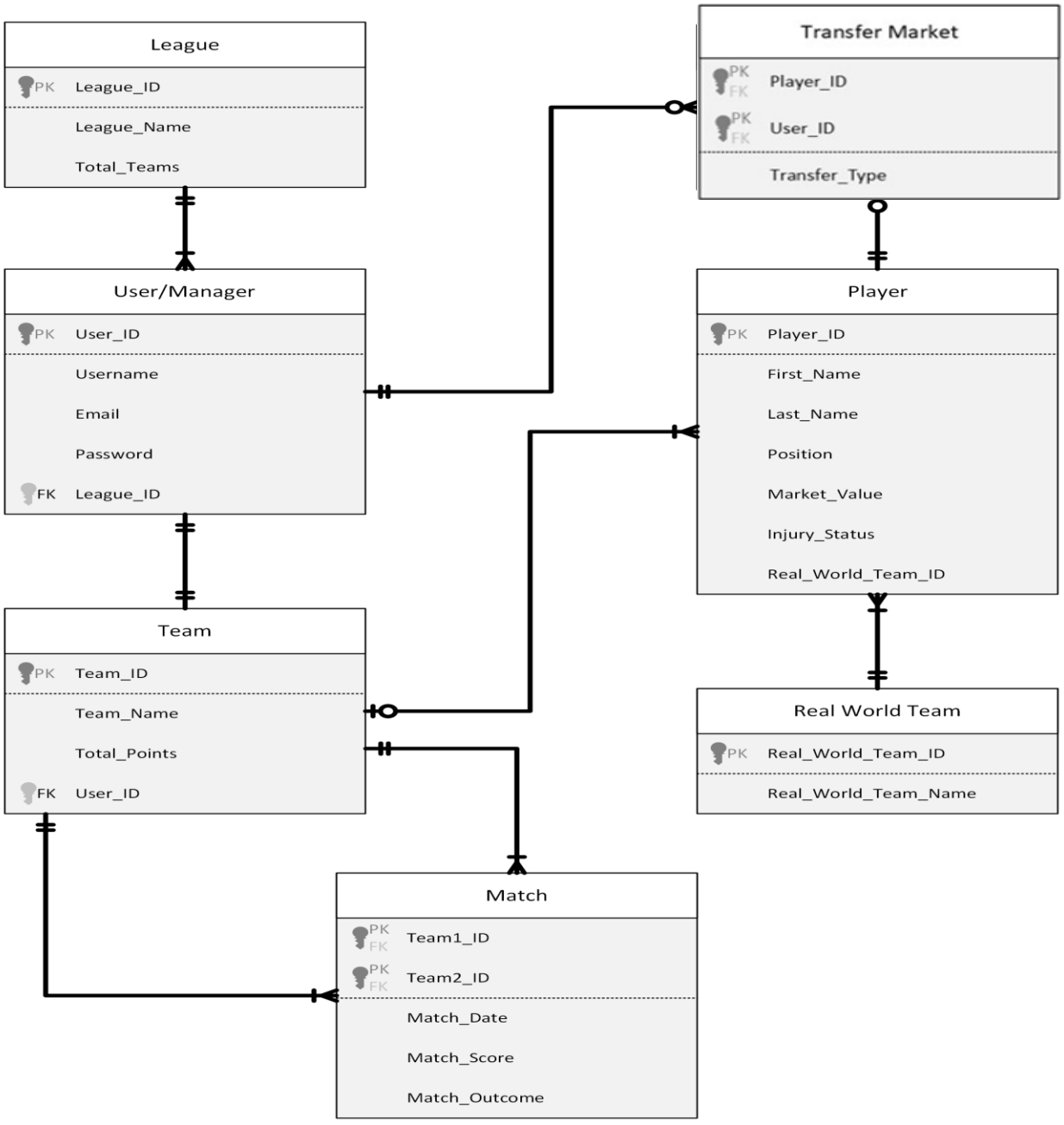
Figure 1.0 Database Entities and Attributes



4.1 Conceptual Design

Figure 2.0 Details the conceptual design of our teams Fantasy Manager Database. This diagram shows the different entities associated with our team's database as well as the relationships between each entity. Section 3.2 goes in detail about the relationships between each entity.

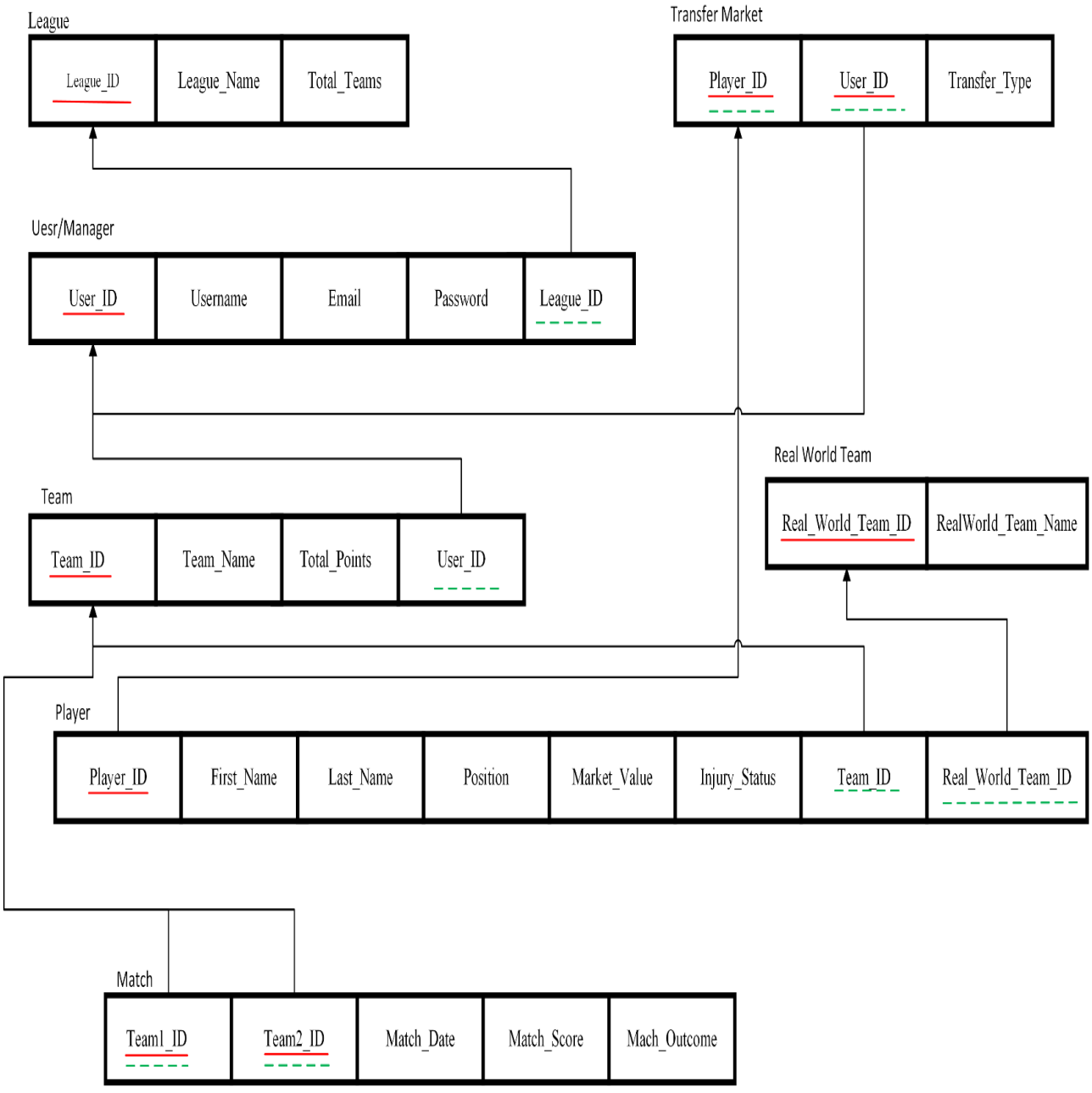
Figure 2.0 Conceptual Design



4.2 Logical Design

Figure 3.0 Details the logical design of our teams Fantasy Manager Database. This diagram translates the conceptual design or ER model into relations. This model shows the relation definitions as well as primary, foreign, and composite keys.

Figure 3.0 Relational Model



5.0 SQL Code

The following sections will dive into the SQL code generated for our team's Fantasy Manager Database. The first section will detail the creation of the database, including the establishment of entities and the development of relationships among them. Lastly, the second section will showcase data manipulation techniques, like inputting actual data into our database. This will demonstrate a variety of data, including details about Players, Leagues, Teams, Matches, and more.

5.1 Data Definition

The following figures 4.0 – 4.6 demonstrate the SQL code behind each entity and their related attributes. This shows how each entity was created and shows the primary and foreign keys and how they reference with each other. This is the foundation of our teams database.

Figure 4.0 League SQL

```
-- Create a table for representing Leagues
CREATE TABLE League (
    League_ID INT PRIMARY KEY,      -- Unique identifier for each league
    League_Name VARCHAR(30),        -- Name of the league
    Total_Teams INT                 -- Total number of teams in the league
);
```

Figure 4.1 User/Manager SQL

```
-- Create a table for representing Users
CREATE TABLE User (
    User_ID INT PRIMARY KEY,        -- Unique identifier for each user
    Username VARCHAR(255),          -- Username of the user
    Email VARCHAR(255),             -- Email address of the user
    Password VARCHAR(255),          -- Password of the user (Note: In practice, this should be hashed)
    League_ID INT,                  -- Foreign key referencing the League the user is associated with
    FOREIGN KEY (League_ID) REFERENCES League(League_ID)  -- Establish a relationship with the League table
);
```

Figure 4.2 Teams SQL

```
-- Create a table for representing Teams
CREATE TABLE Team (
    Team_ID INT PRIMARY KEY,       -- Unique identifier for each team
    Team_Name VARCHAR(255),        -- Name of the team
    Total_Points INT,              -- Total points earned by the team
    User_ID INT,                   -- Foreign key referencing the User who owns the team
    FOREIGN KEY (User_ID) REFERENCES User(User_ID)  -- Establish a relationship with the User table
);
```

Figure 4.3 Players SQL

```
-- Create a table for representing Players
CREATE TABLE Player (
    Player_ID INT PRIMARY KEY,             -- Unique identifier for each player
    First_Name VARCHAR(255),               -- First name of the player
    Last_Name VARCHAR(255),                -- Last name of the player
    Position VARCHAR(50),                  -- Position played by the player
    Market_Value DECIMAL(10, 2),           -- Market value of the player
    Injury_Status VARCHAR(50),             -- Injury status of the player
    Team_ID INT,                           -- Foreign key referencing the Team the player belongs to
    Real_World_Team_ID INT,                -- Foreign key referencing the real-world team associated with the player
    FOREIGN KEY (Team_ID) REFERENCES Team(Team_ID),                -- Establish a relationship with the Team table
    FOREIGN KEY (Real_World_Team_ID) REFERENCES Real_World_Team(Real_World_Team_ID)  -- Establish a relationship with the Real_World_Team table
);
```

Figure 4.4 Matches SQL

```
-- Create a table for representing Game Matches
CREATE TABLE GameMatch (
    Team1_ID INT,                          -- Foreign key referencing the first team in the match
    Team2_ID INT,                          -- Foreign key referencing the second team in the match
    Match_Start_Date DATETIME,             -- Start date and time of the match
    Match_End_Date DATETIME,               -- End date and time of the match
    Match_Score VARCHAR(20),               -- Score of the match
    Match_Outcome VARCHAR(20),             -- Outcome of the match
    PRIMARY KEY (Team1_ID, Team2_ID),      -- Composite primary key for match uniqueness
    FOREIGN KEY (Team1_ID) REFERENCES Team(Team_ID),   -- Relationship with the Team table for Team1
    FOREIGN KEY (Team2_ID) REFERENCES Team(Team_ID)    -- Relationship with the Team table for Team2
);
```

Figure 4.5 Real-World Teams SQL

```
-- Create a table for representing Real World Teams
CREATE TABLE Real_World_Team (
    Real_World_Team_ID INT PRIMARY KEY,       -- Unique identifier for each real-world team
    Real_World_Team_Name VARCHAR(50)          -- Name of the real-world team
);
```

Figure 4.6 Transfer Market SQL

```
-- Create a table for representing Player Transfers in the Market
CREATE TABLE TransferMarket (
    Player_ID INT,                     -- Foreign key referencing the player involved in the transfer
    User_ID INT,                       -- Foreign key referencing the user involved in the transfer
    Transfer_Type VARCHAR(50),         -- Type of transfer (e.g., Buy, Sell, Loan)
    PRIMARY KEY (Player_ID, User_ID),  -- Composite primary key for transfer uniqueness
    FOREIGN KEY (Player_ID) REFERENCES Player(Player_ID),   -- Establish a relationship with the Player table
    FOREIGN KEY (User_ID) REFERENCES User(User_ID)         -- Establish a relationship with the User table
);
```

5.2 Data Manipulation

The following figures 5.0 – 5.6 demonstrates the data manipulation code that added data to our team's database. This includes data such as Player information, Leagues, Matches, Users/Managers, and more. Some of the data manipulation is too long to attach to the following figures. Some of the data is over 100 lines, because of this no more than 10 lines will be shown. This is to get a general understanding of the SQL logic behind it.

Figure 5.0 Inserting League

```
INSERT INTO League (League_ID, League_Name, Total_Teams)
VALUES (1, 'Premier League', 8);
```

Figure 5.1 Inserting Users

```
INSERT INTO User (User_ID, Username, Email, Password, League_ID)
VALUES
    (1, 'SoccerWizardJD', 'john.doe@gmail.com', 'A#b4C6d8Ef0Gh2I', 1),
    (2, 'AliceKicks', 'alice.smith@yahoo.com', 'Jk1LmN&Op3Qr$St', 1),
    (3, 'SoccerBob123', 'bob.jenkins@gmail.com', 'U5vXwYz@7A8B9C', 1),
    (4, 'GoalGetterEmily', 'emily.wang@yahoo.com', 'D1E2Fg3Hi$4JkL', 1),
    (5, 'KickMasterMike', 'mike.jackson@hotmail.com', 'M5N6oPq7R8s#Tt', 1),
    (6, 'SoccerStarSara', 'sara.miller@hotmail.com', 'Uv9W0XyZ!1@2#3', 1),
    (7, 'GoalChaserDave', 'david.white@aol.com', '4aBcD5eFg6H7I$', 1),
    (8, 'SoccerPassionLiv', 'olivia.jones@gmail.com', 'Jk8LmNoP9Qr$St', 1);
```

Figure 5.2 Inserting Teams

```
INSERT INTO Team (Team_ID, Team_Name, Total_Points, User_ID)
VALUES
    (1, 'Dragons FC', 0, 1),
    (2, 'Titans United', 0, 2),
    (3, 'Shadow Strikers', 0, 3),
    (4, 'Phoenix Kings', 0, 4),
    (5, 'Galactic Stars', 0, 5),
    (6, 'Nightmare Legends', 0, 6),
    (7, 'Eternal Warriors', 0, 7),
    (8, 'Mystic Rangers', 0, 8);
```

Figure 5.3 Inserting Real World Teams

```
INSERT INTO Real_World_Team (Real_World_Team_ID, Real_World_Team_Name)
VALUES
    (1, 'Manchester United'),
    (2, 'Manchester City'),
    (3, 'Liverpool FC'),
    (4, 'Chelsea FC'),
    (5, 'Arsenal FC'),
    (6, 'Tottenham Hotspur'),
    (7, 'Leicester City'),
    (8, 'Everton FC'), ...
```

Figure 5.4 Inserting Players

```
INSERT INTO Player (Player_ID, First_Name, Last_Name, Position, Market_Value, Injury_Status, Team_ID, Real_World_Team_ID)
VALUES
  (1, 'Tom', 'Heaton', 'Goalkeeper', 65, 'Healthy', NULL, 1),
  (2, 'André', 'Onana', 'Goalkeeper', 75, 'Healthy', 8, 1),
  (3, 'Victor', 'Lindelöf', 'Defender', 70, 'Healthy', 5, 1),
  (4, 'Harry', 'Maguire', 'Defender', 72, 'Healthy', NULL, 1),
  (5, 'Bruno', 'Fernandes', 'Midfielder', 90, 'Healthy', 2, 1),
  (6, 'Christian', 'Eriksen', 'Midfielder', 85, 'Healthy', 8, 1),
  (7, 'Anthony', 'Martial', 'Forward', 80, 'Injured', 4, 1),
  (8, 'Marcus', 'Rashford', 'Forward', 88, 'Healthy', 5, 1), ...
```

Figure 5.5 Inserting Transfer Market

```
INSERT INTO TransferMarket (Player_ID, User_ID, Transfer_Type)
VALUES
  (1, NULL, 'Available'),
  (2, 8, 'Unavailable'),
  (3, 5, 'Unavailable'),
  (4, NULL, 'Available'),
  (5, 2, 'Unavailable'),
  (6, 8, 'Unavailable'),
  (7, 4, 'Unavailable'),
  (8, 5, 'Unavailable'), ...
```

Figure 5.6 Inserting Matches

```
-- Insert data into Matches
INSERT INTO GameMatch (Team1_ID, Team2_ID, Match_Start_Date, Match_End_Date, Match_Score, Match_Outcome)
VALUES
-- Week 1 Matches
(1, 2, '2023-01-02', '2023-01-08', '5-1', 'HOME WINS'),
(3, 4, '2023-01-02', '2023-01-08', '1-3', 'AWAY WINS'),
(5, 6, '2023-01-02', '2023-01-08', '2-4', 'AWAY WINS'),
(7, 8, '2023-01-02', '2023-01-08', '1-5', 'AWAY WINS'),

-- Week 2 Matches
(3, 1, '2023-01-09', '2023-01-15', '1-4', 'AWAY WINS'),
(4, 2, '2023-01-09', '2023-01-15', '5-2', 'HOME WINS'),
(5, 7, '2023-01-09', '2023-01-15', '3-5', 'AWAY WINS'),
(6, 8, '2023-01-09', '2023-01-15', '4-1', 'HOME WINS'),

-- Week 3 Matches
(1, 4, '2023-01-16', '2023-01-22', '5-0', 'HOME WINS'),
(2, 3, '2023-01-16', '2023-01-22', '4-3', 'HOME WINS'),
(5, 8, '2023-01-16', '2023-01-22', '2-5', 'AWAY WINS'),
(6, 7, '2023-01-16', '2023-01-22', '1-4', 'AWAY WINS'),
```

6.0 Database Queries

This section will cover the SQL queries used to display information. These queries, selected based on what our team deemed important to show, consist of ten in total. Six of these queries utilize single-table data, such as displaying lists of Players, Teams, Leagues, and more. The remaining four queries will display data from multiple tables, such as Players and Teams, or Leagues and Users, among others. The following figures will show each query generated; however, the output of these queries will not be shown in this document. These are also our team's main queries; the code document provided has extra queries as well.

Figure 6.0 Player Information Query

```
-- Select columns from Team and Player tables
SELECT T.Team_ID,
       T.Team_Name,
       P.Player_ID,
       P.First_Name,
       P.Last_Name,
       P.Position

FROM Team AS T       -- From the Team table, aliased as T

-- Perform an inner join with the Player table, aliased as P
INNER JOIN Player AS P

-- Join condition: Match Team_ID in both tables
ON T.Team_ID = P.Team_ID  -- The join is made where Team IDs match between Team and Player tables

ORDER BY T.Team_ID ASC,  -- Orders the results by Team ID in ascending order
         P.Position ASC; -- Then orders by Player Position in ascending order
```

Figure 6.1 Injury Status of All Drafted Players Query

```
-- Select player information along with team names, injury status, and positions
SELECT T.Team_Name,
       P.First_Name,
       P.Last_Name,
       P.Injury_Status,
       P.Position

FROM Player AS P           -- From the Player table, aliased as P

INNER JOIN Team AS T       -- Inner join with the Team table, aliased as T
ON P.Team_ID = T.Team_ID   -- Join is made where Team IDs match between Player and Team tables

WHERE P.Team_ID IS NOT NULL  -- Filter to include only players with an actual team

ORDER BY T.Team_Name ASC,    -- Order the results by team name in ascending order
         P.Position ASC;     -- Then, order by player position in ascending order
```

Figure 6.2 Each Positions Best Player Query

```
-- Select the position, first name, and last name
SELECT P.Position,
       MAX(P.Market_Value) as Max_Value, -- Select the maximum market value for players in each position
       MAX(P.First_Name) as First_Name,
       MAX(P.Last_Name) as Last_Name

FROM Player AS P                        -- From the Player table, aliased as P

GROUP BY P.Position;                -- Group the results by the player's position
```

Figure 6.3 Average Real World Team Market Value Query

```
-- Select the real world team name and calculate the average market value of players
SELECT RWT.Real_World_Team_Name,
       AVG(P.Market_Value) as Avg_Market_Value

FROM Real_World_Team RWT                        -- From the Real_World_Team table, aliased as RWT

JOIN Player P                                   -- Join with the Player table, aliased as P
ON RWT.Real_World_Team_ID = P.Real_World_Team_ID  -- Join condition: match the real-world team ID from both tables

GROUP BY RWT.Real_World_Team_Name;              -- Group the results by the real-world team name
```

Figure 6.4 Player Info for Players Greater Than or Equal To 75.00 Query

```
-- Select first name, last name, position, market value, and availability status of players with market value >= 75
SELECT First_Name, Last_Name, Position, Market_Value,
       CASE
           WHEN Team_ID IS NULL THEN 'Unavailable'  -- If the player doesn't have a team, their availability is "Unavailable"
           ELSE 'Available'                         -- If the player has a team, their availability is "Available"
       END AS Availability
FROM Player
WHERE Market_Value >= 75  -- Filter for players with a market value greater than or equal to 75
ORDER BY Market_Value DESC; -- Order the results by market value in descending order
```

Figure 6.5 User and Team Info Query

```
-- Retrieve usernames and fantasy team names for all users
SELECT User.Username, Team.Team_Name
FROM User
JOIN Team ON User.User_ID = Team.User_ID;
```

Figure 6.6 Real World Teams and Players Query

```
-- Fetch real-world team names and players associated with each team
SELECT Real_World_Team.Real_World_Team_Name, Player.First_Name, Player.Last_Name, Player.Position
FROM Real_World_Team
JOIN Player ON Real_World_Team.Real_World_Team_ID = Player.Real_World_Team_ID
ORDER BY Real_World_Team_Name;
```

Figure 6.7 Undrafted Team Information Query

```
SELECT Player.Player_ID, Player.First_Name, Player.Last_Name, Player.Position, Player.Market_Value, TransferMarket.Transfer_Type, Real_World_Team.Real_World_Team_Name
FROM Player AS Player
LEFT JOIN TransferMarket AS TransferMarket ON Player.Player_ID = TransferMarket.Player_ID
LEFT JOIN Real_World_Team AS Real_World_Team ON Player.Real_World_Team_ID = Real_World_Team.Real_World_Team_ID
WHERE Player.Team_ID IS NULL;
```

Figure 6.8 Schedule of Matches and Outcome Query

```
-- Display upcoming game matches and participating fantasy teams with only dates
SELECT DATE(Match_Start_Date) AS Match_Start_Date,
       DATE(Match_End_Date) AS Match_End_Date,
       Team1.Team_Name AS Home_Team,
       Team2.Team_Name AS Away_Team,
       Match_Outcome
FROM GameMatch
INNER JOIN Team AS Team1 ON GameMatch.Team1_ID = Team1.Team_ID
INNER JOIN Team AS Team2 ON GameMatch.Team2_ID = Team2.Team_ID
ORDER BY Match_Start_Date ASC;
```

Figure 6.9 Players associated with Users Query

```
-- Fetch users and players transferred by each user
SELECT User.Username, Player.First_Name, Player.Last_Name
FROM User
JOIN TransferMarket ON User.User_ID = TransferMarket.User_ID
JOIN Player ON TransferMarket.Player_ID = Player.Player_ID;
```