**Entities**

# Users

**Attributes**: UserId (PK), UserName, Email

**Assumptions:** Represents users such as fans or administrators. Each user has a unique ID which can uniquely identify a user.

**Reason for Entity Modeling:** This entity is modeled to manage user personal data. This can be extended for user authentication, preferences, and other functionalities that might be needed in the future.  
**Cardinality:** Users can like multiple teams and players, and each team and player can be liked by multiple users, indicating a many-to-many relationship with both teams and players through likes.

# Teams

**Attributes:** TeamName (PK), HomeGoals, AwayGoals, WinRecordHome, WinRecordAway  
**Assumptions**: Each TeamName can uniquely represent a team, ensuring no two teams share the same name. This is used to represent various football teams along with their performance metrics and to establish a link with players.  
**Reason for Entity Modeling**: This entity captures comprehensive details about teams, which are essential for analyzing team performance and statistics. Modeling teams as a separate entity allows for storing specific performance metrics and linking with players and stadiums.

**Cardinality**: Each team can have multiple players, establishing a one-to-many relationship with players. Additionally, each team has one dedicated home stadium, ensuring a one-to-one relationship with stadiums.

# Players

**Attributes**: PlayerId (PK), PlayerName, TeamName (FK to Teams.TeamName), Nationality, Age, Position  
**Assumptions**: Represents individual players and their team affiliations with their background. Each Player can be uniquely identified by a PlayerId that cannot be shared.  
**Reason for Entity Modeling**: This entity is modeled to manage player details such as their personal information and their affiliation with teams. It also links with game history for performance tracking, enabling detailed analysis of each player’s contributions.  
**Cardinality**: Each player has one game history record, establishing a one-to-one relationship with player game history. Players are part of only one team at a time, ensuring a many-to-one relationship with teams as a team is composed of multiple players.

# Stadiums

**Attributes**: StadiumName (PK), HomeTeam (FK to Teams.TeamName), Capacity  
**Assumptions**: Represents stadiums, each associated with a home team that can be uniquely identified by the StadiumName.  
**Reason for Entity Modeling**: This entity stores stadium-related information, which is crucial for the venue details. Modeling stadiums as separate entities allows for storing capacity and linking each stadium with a specific team.

**Cardinality**: One-to-one with teams as each team has a unique home stadium allocated.

# Player Game History

**Attributes**: PlayerId (PK, FK to Players.PlayerId), MinutesPlayed, Assists, GoalsScored, Appearances, CleanSheets, Cards, Rank  
**Assumptions**: Historical game performance metrics for players. The dataset we plan on using is in aggregate, so this has a one-to-one relationship with players that can be uniquely identified by a PlayerId.  
**Reason for Entity Modeling**: This entity is modeled to track detailed game statistics, facilitating performance analysis over time. Modeling this as a separate entity ensures that each player’s performance data is encapsulated and can be easily linked to the respective player. This approach supports comprehensive performance analysis and historical tracking given it is available as a separate dataset.  
**Cardinality**: One-to-One with players because each player has his own unique data record for the games he has played.

# Normalization

The database schema adheres to BCNF which avoids redundant data and preserves the data integrity by imposing stricter rules than 3NF:

**Users**(UserId,Username,Email) has FD:

**UserId** →Username, Email

**Teams**(TeamName, HomeGoals, AwayGoals, WinRecordAtHome, WinRecordAway) has FD:

**TeamName** →HomeGoals, AwayGoals, WinRecordAtHome, WinRecordAway

**Players**(PlayerId, PlayerName, Team, Nationality, Age, Position) has FD:

**PlayerId** → PlayerName, Team, Nationality, Age, Position

**Stadiums**(StadiumName, HomeTeam, Capacity) has FD:

**StadiumName** → HomeTeam, Capacity

**PlayerGameHistory**(PlayerId, MinutesPlayed, Assists, GoalsScored, Appearances, CleanSheets, Cards, Rank) has FD:

**PlayerId** →MinutesPlayed, Assists, GoalsScored, Appearances, CleanSheets, Cards, Rank

All above functional dependencies have left side as super key(primary key) and are non trivial proving that we follow BCNF.

# Logical Relational Schema

**Users**  
Users(UserId: INT [PK], UserName: VARCHAR(255), Email: VARCHAR(255))  
  
**Teams**  
Teams(TeamName: VARCHAR(255) [PK], HomeGoals: INT, AwayGoals: INT, WinRecordHome: INT, WinRecordAway: INT)

**Players**  
Players(PlayerId: INT [PK], PlayerName: VARCHAR(255), Team: VARCHAR(255) [FK to Teams.TeamName], Nationality: VARCHAR(255), Age: INT, Position: VARCHAR(255))  
  
**Stadiums**  
Stadiums(StadiumName: VARCHAR(255) [PK], HomeTeam: VARCHAR(255) [FK to Teams.TeamName], Capacity: INT)  
  
**Player Game History**  
PlayerGameHistory(PlayerId: INT [PK, FK to Players.PlayerId], MinutesPlayed: INT, Assists: INT, GoalsScored: INT, Appearances: INT, CleanSheets: INT, Cards: VARCHAR(255), Rank: INT)

**User’s Liked Players**

UserLikedPlayers(UserId: INT [FK to Users.UserId], PlayerId: INT [FK to Players.PlayerId])

**User’s Liked Teams**

UserLikedTeams(UserId: INT [FK to Users.UserId], TeamId: INT [FK to Teams.TeamId])