These are our assumptions for entities and relationships in our model.

- User
    - This is an entity since each user should have their own profile, where they can have their unique login info and set preferences. The user entity has two relationships.
    - Relationship 1: Users are related to Players because they can choose a favorite player, whose stats are put on the homepage for easy access. This is a many-to-one relationship, since a user can only have one favorite player, but a player can be a favorite of many users.
    - Relationship 2: In a similar fashion, Users are related to Teams because they can choose a favorite team, whose stats are put on the homepage for easy access. This is a many-to-one relationship, since a user can only have one favorite team, but a team can be a favorite of many users.
        - For these teams, a user will choose a team name, but not a specific year. We will focus on the most recent information to provide the user from the current season, with additional information from past seasons.
- Player
    - This is an entity because we need to keep track of certain attributes for each player that cannot be represented in the other tables.
    - Relationship 1: Team_Name to Team_Name: Many to one as multiple players can map to the same team. Each player can only have one team, since the way we are designing our database will have it list the player's team as the team they most recently played on
    - Relationship 2: Start_Season to Start_Year: Many to one as multiple players can start on the same season
    - Relationship 3: End_Season to Start_Year: Many to one as multiple players can retire on the same season
- Team
    - Team is an entity because a team can change from year to year; players are being drafted, retiring, and being traded, so teams can look very different every year.
    - Relationship 1: Team is related to Season since teams change every season. We need to have seasonal entries for each team to keep track of how their roster changes. This is a many-to-one relationship since each team only corresponds to a specific season, but every season will have every team corresponding to it. This is because each team and season is a unique row. To accommodate a team's roster changing every season, a new row is made.
- Game

    - We made Game its own entity so it would be easy to retrieve the scores of any games played which will likely be something that the user will want to see. The gameID is a PK that is unique for each game, which will naturally uniquely identify the teams playing, date, season, and final-score.
    - Relationship 1: Season to Season: Many to one as many games can be part of at most one season, but each individual game only belongs to one season (the one it took place in)
    - Relationship 2: Home_Team to Team_Name: many to one as the different games in each row can connect to the same Team_Name but each Home_Team (LHS) will only at max connect one Team_Name(RHS)
    - Relationship 3: Away_Team to Team_Name: many to one as the different games in each row can connect to the same Team_Name but each Away_Team (LHS) will only at max connect one Team_Name(RHS)
- Shot
    - Relationship 1: GameID to GameID: many to one as there are multiple shots attempted per game
    - Relationship 2: PlayerID to PlayerID: many to one as a player takes multiple shots
- Season
    - This is an entity because we need to keep track of seasonal awards, champions, and team rosters, which change on a per season basis.
    - Relationship 1: Champion is related to team by team name and season because only one team can win a specific season. This is a one-to-one relation as there is only one champion per season, and there are no other columns that intake a Team.
    - Relationship 2: Player awards are related to Players because each player award is won by one player once a year. This is a many-to-one relationship, since a player can win multiple awards per season but each each award can only be won by one player

These are our functional dependencies and their decompositions.

- User
    - Username -> Password
    - Username -> Favorite_Player
    - Username -> Favorite_Team
    - Since all of the dependencies involve the primary key on the left side, this does not need to be normalized any further.
- Player
    - PlayerID -> Player_Name

    - PlayerID -> Team_Name
    - PlayerID -> Other Statistics
    - Looking at all the dependencies, all the left hand sides of the dependencies involve primary keys, every right hand side can be uniquely determined. This means that normalization and further decomposition is not needed.
- Team
    - Team_Name, Season -> PlayerIDs
        - To conserve space, we are taking the PlayerID of each player on the team and combining them into one 60-char VARCHAR (15 players on a team)
    - You need both the team name and the season to determine its players, since the sets of players can change for each team year over year. Team name and season combine to create a primary key, which also uniquely determines every right hand side. This means that normalization and decomposition is not needed.
- Game: Primary key is GameID
    - GameID -> Home_Team
    - GameID -> Away_Team
    - GameID -> Final_Score
    - GameID -> Season
    - The natural composition of this entity is one table with GameID as the primary key and everything else (Home_Team, Away_Team, Final_Score, Season) as dependent on the primary key as everything else is all uniquely determined by that specific game, the gameID. That is the only functional dependency in this table so no further decomposition is needed.
- Shot
    - Game_ID, Shot_Time -> Player_ID
    - Game_ID, Shot_Time -> X
    - Game_ID, Shot_Time -> Y
    - Game_ID, Shot_Time -> Result
    - Looking at all the dependencies, all the left hand sides of the dependencies involve primary keys, every right hand side can be uniquely determined. This means that normalization and further decomposition is not needed.
- Season
    - Start_Year -> Champion
    - Start_Year -> Individual Awards(Most_Valuable_Player, Defensive_Player_Of_The_Year, …, Finals_MVP)

        - These are all the awards and the players who won them for the season. Each award is given to one player, and these will be the PlayerID of the player that won the award
    - Looking at all the dependencies, all the left hand sides of the dependencies involve primary keys, every right hand side can be uniquely determined. This means that normalization and further decomposition is not needed.

This is the logical design of our conceptual database:

- **User**(Username:VARCHAR(256)[PK], Password:VARCHAR(256), Favorite_Player:VARCHAR(256)[FK to Players.PlayerID], Favorite_Team:VARCHAR(256)[FK t o Team.Team_Name])
- **Player**(PlayerID:INT [PK], Player_Name: Varchar(256), Team_Name:VARCHAR(256)[FK to Team.Team_Name], Start_Season: VARCHAR(4) [FK to Season.Start_Year], End_Season: VARCHAR(4) [FK to Season.Start_Year], Position: VARCHAR(2), Games_Played: INT, Games_Started: INT, Minutes_Played: INT, Field_Goals_Made: INT, Field_Goals_Attempted: INT, 3_Pointers_Attempted: INT, 3_Pointers_Made: INT, 2_Pointers_Attempted: INT, 2_Pointers_Made: INT, Free_Throws_Made: INT, Free_Throws_Attempted: INT, Offensive_Rebounds: INT, Defensive_Rebounds: INT, Assists: INT, Steals: INT, Blocks: INT, Turnovers: INT, Personal_Fouls: INT, Points: INT
- **Team**(Team_Name:VARCHAR(256)[PK], Season:VARCHAR(4) [PK, FK to Season.Start_Year], PlayerIDs:VARCHAR(60)
- **Game**(GameID:INT[PK], date:DATE, Season: VARCHAR(4) [FK to Season.Start_Year], Home_Team: Varchar(256) [FK to Team.Team_Name], Away_Team: Varchar(256) [FK to Team.Team_Name], Final_Score: Varchar(256))
- **Shot**(GameID:INT[PK, FK to Game.GameID],Shot_Time:INT[PK], PlayerID:INT[FK to Player.PlayerID], X:INT, Y:INT, Result:Boolean)
- **Season**(Start_Year: VARCHAR(4) [PK], Champion: VARCHAR(256) [FK to Team.Team_Name], Most_Valuable_Player: INT [FK to Player.PlayerID], Defensive_Player_Of_The_Year: INT [FK to Player.PlayerID], Rookie_Of_The_Year: INT [FK to Player.PlayerID], Sixth_Man_Of_The_Year: INT [FK to Player.PlayerID], Most_Improved_Player: INT [FK to Player.PlayerID], Clutch_Player_Of_The_Year: INT [FK to Player.PlayerID], Finals_MVP: INT [FK to Player.PlayerID])