1. UserInfo

- Assumptions:
    - Each user must have a unique ID (UserID) to distinguish them from others.
    - Username, email, and password are essential for user login and authentication.
- Why Entity:
    - The UserInfo table is an entity because it represents a core actor in the system, and its attributes are user-specific data.
    - It's more efficient to keep this information in a separate table rather than embedding it in other tables like UserBets, as it would violate normalization rules and cause redundancy.
- Cardinality:
    - A single user can have multiple bets however, each bet record corresponds to one user - (1 to N relationship between UserInfo and UserBets).

2. UserBets

- Assumptions:
    - Users can place multiple bets, and each bet will have a unique BetID.
    - BetTypeID identifies the type of bet (e.g., win/loss, over/under) placed by the user.
    - GameID links the bet to a specific game between two teams.
    - Status tracks the outcome of the bet (e.g., won, lost, pending).
    - Amount represents the wager placed by the user.
- Why Entity:
    - Bets need to be tracked separately from users because a user can place numerous bets, each with unique characteristics like the amount and bet type.
    - It wouldn't be efficient to store betting information as attributes within the UserInfo table, as it would repeat user information for every bet.
- Cardinality:
    - Each bet relates to a single user (N to 1 relationship between UserBets and UserInfo).
    - Each bet is related to a specific game but one game can have many bets placed on it (N to 1 relationship between UserBets and Games).
    - Each bet has a specific bet type but multiple bets can relate to the same bet type (N to 1 with BetTypes through BetTypeID).

3. Games Entity

- Assumptions:
    - A GameID uniquely identifies each game.
    - HomeTeamID and AwayTeamID reference the teams playing, and WinTeamID captures the result.
    - GameDate records when the game took place.
- Why Entity:

    - The Games table is a separate entity because it stores unique games' information and allows easy tracking of historical data.
    - Storing game information as an attribute within UserBets would lead to redundancy, as multiple users may place bets on the same game.
- Cardinality:
    - A game can have multiple bets associated with it (1 to N relationship between Games and UserBets).
    - A team can be involved in many games(1 to N relationships between Teams and Games).

4. HistoricalOdds Entity

- Assumptions:
    - OddsID references different odds types (e.g., moneyline, spread).
    - OddsValue stores the numerical value of odds for a particular game.
- Why Entity:
    - HistoricalOdds is modeled as an entity to store specific betting odds for each game and bet type. This allows easy updates and queries for the odds related to past games.
    - Storing odds data directly within the Games or UserBets table would cause data duplication and inefficiencies.
- Cardinality:
    - Each odds record references a single game and a single bet type (N to 1 relationships with Games and BetTypes).

5. BetTypes Entity

- Assumptions:
    - Contains distinct types of bets that users can place (e.g., win/loss, over/under).
    - BetTypeName provides a description of the type.
- Why Entity:
    - Bet types are abstracted into a separate entity to avoid data duplication and ensure consistency. It allows the program to add or update bet types without altering the bet records.
- Cardinality:
    - A bet type can be applied to multiple bets (1 to N relationship between BetTypes and UserBets).

6. Teams Entity

- Assumptions:
    - Each team is uniquely identified by TeamID.
    - TeamName stores the name of the team.
- Why Entity:
    - Teams are abstracted as separate entity because each team can participate in many games, and each game involves two teams (home and away).

    - Storing team information directly within Games would lead to redundancy and would not allow efficient updates.
- Cardinality:
    - A single team can participate in many games (1 to N relationship between Teams and Games).

Normalization:

We used BCNF to normalize to eliminate more redundancy and provide better data integrity. For instance, in our initial design, if UserInfo and UserBets were combined into a single table, it would cause issues because UserID alone could not uniquely identify the other attributes in UserBets (such as GameID, Amount, or BetTypeID). This would lead to data redundancy and potential update anomalies.

For example, if a user placed multiple bets and then updated their email address, the email information would need to be updated in multiple bet records. This would not only be inefficient but could also result in inconsistent data if some records were missed during the update.

To resolve these issues, we separated UserBets into its own table and introduced UserID as a foreign key. Each bet is uniquely identified by BetID and references a specific user. This adjustment ensures that user attributes like email and password are only stored in the UserInfo table, and all bet-related information remains in the UserBets table. This separation eliminates partial dependencies.

Similarly, we created a separate Games table to store information about each game, with a GameID as the primary key. This allows multiple bets to reference a single game without duplicating game data, ensuring that any updates to game details do not have to be made across multiple records in UserBets.

The BetTypes entity was also separated to manage different types of bets. This way, each BetTypeID uniquely identifies a bet type, and multiple bets can refer to the same bet type. This design adjustment ensures consistency in the classification of bets.

Relational Schema

UserInfo Table
UserID: INT [PK]
Username: VARCHAR
Email: VARCHAR
Password: VARCHAR

Games Table
GameID: INT [PK]
HomeTeamID: INT [FK to Teams.TeamID]
AwayTeamID: INT [FK to Teams.TeamID]
WinTeamID: INT [FK to Teams.TeamID]
GameDate: DATE
WinTeamScore: INT
LoseTeamScore: INT

Teams Table
TeamID: INT [PK]
TeamName: VARCHAR(100)

UserBets Table
BetID: INT [PK]
UserID: INT [FK to UserInfo.UserID]
GameID: INT [FK to Games.GameID]
BetTypeID: INT [FK to BetTypes.BetTypeID]
Amount: DECIMAL(10, 2)
Status: VARCHAR

BetTypes Table
BetTypeID: INT [PK]
BetTypeName: VARCHAR(50)

HistoricalOdds Table
BetID: INT [PK]
GameID: INT [FK to Games.GameID]
OddsID: INT [FK to BetTypes.BetTypeID]
OddsValue: INT