**Normalization**

A table is required to store the list of all the teams and their managers with their respective details.

List\_of\_teams (TeamName, Stadium, ManagerName, ManagerAge, ManagerStyle)

The above table is clearly satisfying 1NF since all the column names are unique and all the columns would contain atomic values and all the data present in a column are of the same type.

But a deletion anomaly occurs when we try to delete all the records from the table.

When we delete all the team details (or records) from the table, we accidentally delete all the manager information as well. So, to avoid this kind of anomaly we split the table into two different tables i.e.

ManagerDetails (M\_id, ManagerName, Age, Style)

List\_of\_Teams (TeamName, M\_id, Stadium)

Here in the above tables, M\_id is a primary key in the ManagerDetails table and M\_id is a foreign key in the List\_of\_Teams table. When the table is split into two tables and now when we delete all the team details (or records) from the List\_of\_Teams table, the manager details remain unaltered. So, we can say that our tables now satisfy the 2NF form since it is devoid of deletion anomalies that we previously encountered and there are no partial dependencies.

Our Tables (List\_of\_teams & ManagerDetails) are also satisfying the 3NF form since there are no transitive dependencies.

Another table is required to store the details of all the players of each team.

List\_of\_Players (Player\_id {TeamName, Jersey No}, Name, Pos, DOB, AGE)

Here TeamName and Jersey No of the player together form a composite key which uniquely identifies all the records of our List\_of\_Players table.

The above table (List\_of\_Players) clearly satisfies 1NF since all the column names are unique and all the columns would contain atomic values and all the data present in a column are of the same type.

Here in this table TeamName is a foreign key which refers to the TeamName in the List\_of\_Teams table.

The List\_of\_Players table is also satisfying the 2NF and 3NF since there are no partial and transitive dependencies. And the redundant data is minimized and it is devoid of all the anomalies.

A table is required to store all the matches that took place between different teams and all the players who played in a particular match.

We could create a single table that contains all the details of the players who played in different matches but doing so would lead to a lot of redundant data in our table and can cause insertion, deletion and update anomalies.

So, we split the table we intended to create into 3 different tables namely

Fixtures (F\_id, Date, time, HomeTeam, AwayTeam, score)

Homeplayers (F\_id, HomeTeam, Jersey\_No, Playing11)

Awayplayers (F\_id, Awayteam, Jersey\_No, Playing11)

The fixtures table would contain all the previous match information like the date, time, score etc. whereas the Homeplayers and the Awayplayers table would contain whether a particular player was in the match as one of the playing 11 players or if he was in the substitutes or reserves.

The F\_id is the primary key in the fixtures table and is a foreign key in the tables Homeplayers and Awayplayers.

The fixtures table satisfies the 1NF, 2NF and the 3NF since all the column names are unique and the values are atomic and it doesn’t have any partial or transitive dependencies.

In the Homeplayers table, we can identify each record of the table uniquely by considering the F\_id, HomeTeam and Jersey\_no together as a primary key. So therefore (F\_id, HomeTeam, Jersey\_no) together form the composite key which can uniquely identify each record of the table.

The Homeplayers table satisfies the 1NF form since all the column names are unique and the values are atomic. It also satisfies the 2NF form since it doesn’t have any partial dependencies because of the functional dependency

{F\_id, HomeTeam, Jersey\_no} → Playing11 i.e. all the 3 columns are required to identify if a player was in the Playing11 or not.

The Homeplayers table also satisfies the 3NF form since there aren’t any transitive dependencies because no non-prime attribute depends on any other non-prime attribute.

Similarly, we can say the same for the Awayplayers table too.

Therefore, all the tables in the database are normalized to remove all the anomalies and to reduce redundant data.