Summary of Dataset

For our project, we have chosen to use data from the 2024-2025 NBA season. The dataset includes game-by-game stats on every player, team data, player data, draft data, combine data, schedule data, coach data and arena data The complete dataset is drawn from nine different files with a total of 36122 rows and 151 columns. The size of the dataset is a great fit for this project as it is large and diverse enough to be broken down into 10 or more tables, while still being manageable and complex enough to make some interesting queries. In this stage we will trim attributes that aren't relevant to our needs.

As mentioned, there are 8 different files that this database will be taking information from. The `common_player_info.csv` file contains 4,171 rows with 33 attributes describing player details, including `person_id`, `height`, `weight`, `birthdate`, `school`, and `position`. The `draft_combine_stats.csv` file has 1,202 rows and 47 attributes of player measurements and combine stats from the NBA Draft Combine, such as `player_id`, `season`, `standing_vertical_leap`, `max_vertical_leap`, and `bench_press`. The `draft_history.csv` file has draft outcomes for players from 1947–2023, with 7,990 rows and 14 attributes, including `person_id`, `season`, `round_number`, `round_pick`, and `overall_pick`. The `player.csv` file holds a simpler set of player records from the 1947-2023, that contains 4,831 rows with 5 attributes, including `id`, `first_name`, `last_name`, `full_name`, and `is_active`. The team.csv file contains 30 rows representing all NBA teams, with 7 attributes such as `id`, `full_name`, `abbreviation`, `city`, `state`, and `year_founded`. NOTE that this is for ONE year. These csv files were taken from https://www.kaggle.com/datasets/wyattowalsh/basketball.

The `play in.csv` contains detailed game-by-game stats for every player in the 2024-2025 season, with 16512 rows, and 25 attributes. Some attributes include, `Player`, `Tm` (team), `Opp` (oppponent), `Res` (result), `MP` (minutes played), `FG` field goals made, `FGA` field goal attemps, `ORB` offensive rebounds, and many other performance metrics. This csv file was taken from https://www.kaggle.com/datasets/eduardopalmieri/nba-player-stats-season-2425.

The `coaches_24_25.csv` contains 35 rows and 22 attributes that summarize the 2024-2025 coaches. Attributes include `Coach` (name), `Tm` (team), `Seasons_with_franchise`, `Seasons_overall`, and attributes about coaching performance in the regular season and playoffs, including games coached, games won, and games lost. This csv file was taken from https://www.basketball-reference.com/leagues/NBA_2025_coaches.html.

The `Arena.csv` contains 5 columns 30 rows of all arenas that are played in at the NBA. The data inside is `Arena name`,`seating capacity`,`Opening year`,`Team name` and `Arena location`. The csv file was taken from https://geojango.com/pages/list-of-nba-teams?srsltid=AfmBOooHCjFL0n6ZRB-rIDjEjJ8x_ZAeYeU2I4F2A7WTUGfL7jPp9f40

Lastly, the `Game.csv` file records the season schedule with 1321 rows and 12 attributes, such as `date`, `Start` (time start ET), `Vistor`, `Home`, `Away_PTS`, `Home_PTS`, an other attributes such as attendance and arena name. This csv file was taken from https://www.basketball-reference.com/leagues/NBA_2025_games.html. The pandas library was used to read each months schedule into one csv file.

Accomplished cleaning plan

First and foremost we have cleaned every attribute we did not use in our ER/EER diagram. There was some redundancy in some of the tables and atrributes we did not want to use since our ER diagram was large enough. We have made sure that the columns represent their names as in the ER diagram as well

- Changed column names in `teams.csv` to match all attributes in my ER diagram no columns were removed
- Removed column `full name` in `player.csv` since we have first name and last name already in place. Column `id` got changed to `PlayerID`.

- Made table `Game.csv` and removed `NBA_24_25_Schedule.csv` Since we wanted to be more clear when designing the ER diagram. Added column `GameID` in `Game.csv` to make the games easily identifiable. There were a few empty columns, we dropped and renamed the columns appropriate to match the ER diagram. We added a result for home if they win or lose.
- In `Draft_History.csv` is our `drafts` relationship. We removed column `Player_profile_flag` and we will keep some of the redundancy for now for the normalization steps. For example having `team_city` and `team_abbreviation` in here when its already in `Team.csv`
- In `draft_combine_stats.csv` we decided to remove a lot of columns to match what attributes we have for the ER model. There is some redundancy we will keep for the normalization steps
- For `database_24_25.csv`, it will now be renamed `play in.csv` since there is all the stats in a player's game. There were additional attributes that we removed that were not used in the ER model. There is also redundancy that we will keep in here until the normalization step.
- `Common_Player_info.csv` is known as our `Player information` entity type we removed every column to do with name and only kept the attributes we have in the ER model. We have also had to fix the format of height since it was recognized as a date.
- `Coaches_24_25.csv` Added coachID in this since it was not added from before I will leave some redundancy in there for the normalization process.
- `Arenas.csv` some redundancy is in there till the normalization part but the rest of the columns are being left in there.

We scraped the 2024-2025 NBA coach statistics from Basketball Reference. We then removed unnecessary columns, and renamed columns.

We collected the full 2024-2025 NBA schedule by scraping each monthly schedule from Basketball Reference and concatenated them into one csv file.

We updated the player table to include rookies from the 2024–2025 season by comparing player names in the `plays_in.csv` file to those in the `player.csv` file. New players were assigned sequential `PlayerIDs` beginning after the current highest `PlayerID` in `player.csv`. The new player was then appended to the player table. If a rookie shared the same first and last name as an existing player, they were not re-added to avoid duplicate entries. This is a mild limitation, as it will omit a few players in `plays_in.csv` who share a name with someone already in `player.csv`, but the overall impact on the database is negligible given that only a small number of such cases exist. I performed sanity checks to ensure that all rookies had the correct `PlayerID` mapped to them.

The `plays_in.csv` table was then updated to replace `Player` with `PlayerID`. We also performed several sanity checks here, to confirm that all `Player` values were replaced with the correct `PlayerID`.

Normalization

For this part all entities and their relationships will follow a series of steps on how we execute an EER model to a BCNF merged relational model

Translate ER model into relational model

The first thing that we did was make sure that the dataset was cleaned.

For each table that we have we will normalize the table in each the highest normal form that the table can go

Arena

1. Translation to 1NF This table is in 0NF since there is a composite attribute that is not atomic. That is arena location so we will break this down into Arena City Name and Arena state/province.
2. Translation to 2NF Now we know that this table is in 1NF we want to make sure there is no partial key dependency in the table

- Arena name -> seating capacity,opening year, Arena city name, arena state/province
- TeamID -> Arena name We will shift our arena name to be under teams.csv and remove the teamID from arena.csv. We can then remove the duplicate arena names

3. Translation to 3NF Now that this table is in 2NF we want to make sure there is no transitive dependencies
   - There is one from Arena name -> city, city-> state in this dataset it is true that this occurs. We have decided to remove state instead of splitting city into another table. There is no other transitive dependencies now so we have this table in 3NF.
4. Translation to BCNF For the final form translation we have our 3NF table and we will make sure that non-key attributes do not determine anything
   - We see that both Arena name and City name are both candidate keys now. We will keep arena name as our primary key. Thus `Arena.csv` is now in BCNF

   We have now finalized the table to being Arena(Arena name, Seating capacity, opening year,arena city name)

Coaches

Before we start the normalization process we have mentioned in our ER diagram that the coach stats has 2 different coach stats, regular and playoff so we will split this off beforehand appropriately.

1. This table we see starts off in 1NF because for coach name we will keep together. There are no other MVA lists or composite attributes so this table is in 1NF.
2. Translation to 2NF
   - We have our table in 1NF and there is no partial key dependancy observed. This table is now in 2NF
3. Translation to 3NF
   - We will look to see if there is any transitive dependencies which there is not. Coach name and CoachID can determine all, teamID determines nothing and so does the stats
4. Translation to BCNF
   - As we have our table to 3NF we see that all determinants are superkeys which are CoachID and Coach Name

   The final relational model for these tables is Coach(CoachID,Coach_name,Team_id,Seasons_Franchise,Seasons_Overal

   RegularGameCoachStats(CoachID,Reg_Current_G,Reg_Current_W, Reg_Current_L,Reg_Franchise_G,Reg_Franc
   Reg_Overall_G,Reg_Overall_W,Reg_Overall_L)

   PlayoffGameCoachStats(CoachID, Playoffs_Current_G,Playoffs_Current_W,Playoffs_Current_L,Playoffs_Franchise
   Playoffs_Franchise_L,Playoffs_Overall_G,Playoffs_Overall_W,Playoffs_Overall_L)

Team

1. The teams table will start off in 0NF since there is a city attribute that does not make much sense to have in teams so that will be removed. That will put us now in 1NF
2. The teams table is in 1NF and now we would like to get to 2NF

- There is no partial key dependancy in this table so this table moves onto 2NF

3. The teams table is now in 2NF we want it in 3NF
   - Our primary key TeamID is able to identify all attributes alone so this table is in 3NF
4. Teams is now in 3NF we want it now in BCNF
   - There are no determinants that are not superkeys. So `Teams.csv` is now in BCNF

   The final relational model for this table
   Team(Team_id,Team name,Team Abbrv,nickname,state, year_founded, Arena Name)

Game

1. This table was in BCNF to begin with no changes were needed

Normalizing the plays_in table

To normalize the `plays_in` table, I first replaced `Tm` (the player's team) with `TeamID`, since `TeamID` is the primary key of the `team` table and provides a foreign key reference to the `team` table. I then removed `Opp` (the opponent team), as it is a derived attribute that can be determined by matching the `GameID` in the `plays_in` table to the `GameID` in the `Game` table, then if the `TeamID` matches the `HomeTeamID` in `Game`, then the opponent is the visitor team, otherwise it's the home team. Next, I removed several other derived attributes, including `PTS`, which can be calculated as `PTS = 2 × (FG - 3P) + 3 × (3P) + FT`, and the percentage columns `FT%`, `FG%`, and `3P%`, which can be computed from their corresponding makes divided by attempts (ex. `FG% = FG/FGA`). Finally, I removed `Res`, since a player's win or loss result can also be derived from the `Game` table by matching the `GameID` in the `plays_in` table to the `GameID` in the `Game` table, then if the `TeamID` in the `plays_in` table matches the `HomeTeamID` in the `Game` table, and the home team won, then the player's result is a win, otherwise it's a loss. The same logic can be used for when the `TeamID` in the `plays_in` table matches the `VisitorTeamID` in the `Game` table. After these changes, all non-key attributes in the `plays_in` table depend solely on the primary key (`PlayerID`, `GameID`), and all determinants are superkeys. Therefore, the table is fully normalized to BCNF.

Normalizing the player table

The `player` table is already in BCNF. Since all determinants are superkeys, as the only determinant is the primary key, which is `PlayerID` and no non-key attributes determines another, the table satisfies BCNF.

Normalizing the player_info table

To normalize the `player_info` table, I removed the `roster status` attribute, since it duplicates the `is_active` column in the `player` table. After this, all non-key attributes in the `player_info` table depend solely on the primary key which is `PlayerID`, and all determinants are superkeys. Therefore, the table is fully normalized to BCNF.

Normalizing the combine_stats table

To normalize the `draft_combine` table, I first removed the redundant-name related columns including `PlayerName`, `FirstName`, and `LastName`, since this information is already in the `player` table. I also removed the `Position` column, as the player positions are already stored in the `player_info` table. Lastly, I removed the `height_wo_shoes` attribute. I did this because, even though the measurement is interesting the `player_info` table already includes player height, and it was unclear whether that value represented height with or without shoes. Without knowing this, comparing the two heights would not provide any meaningful information and would only introduce redundancy. After these changes, all non-key attributes in the `combine_stats` table solely depend on the primary key (`Season`, `PlayerID`), and all determinants are superkeys.Therefore, the table is fully normalized to BCNF.

Normalizing the draft_history table

To normalize the `draft_history` table, I first removed redundant columns including `PlayerName`, `TeamCity`, `TeamName`, and `TeamAbbreviation`, since this information already exists in the `player` and `team` tables. Next, I noticed that `organization` determines `organization type`, creating a transitive dependency because `(PlayerID, Season)` determines `organization`, and `organization` determines `organization type`. To eliminate this dependency, I created a separate `organization` table containing `OrganizationID`, `OrganizationName`, and `OrganizationType`. I replaced the `organization` and `organization type` columns in `draft_history` with a single foreign key reference to `OrganizationID`. Finally, I introduced a surrogate key (`DraftID`) to simplify joins and improve clarity. Because `PlayerID` and `Season` together can determine `DraftID`, the table is in 3NF.

Merging process

All the tables we have viewed are not going to be merged since merging the tables does not make any sense.

The only table that would have made the most sense to merge into another is the `common_player_info.csv` and `Player.csv`. The reason we are keeping the tables seperate is because once a player is added into the database players.csv is not touched again. When we need to update information about a player that can change this will all be kept in player information.