NBA Database Project

Team Members -

- ★ Jay Gala
- ★ Mitali Mishra
- ★ Saurabh Raut
- ★ Vandana Miglani

Team Mentor -

Hayden Hyunjoo Ji

Abstract

The National Basketball Association (NBA), one of the most popular sports in America, is a professional basketball league consisting of 30 teams.

Basketball Datasets have a wide variety of information - ranging from detailed information about players, games played and statistics about the teams. In order to elaborate about the scale of the dataset, it is helpful to consider the kind of information recorded per game. Each NBA Team plays a regular 82-game season schedule. As there are 30 teams, the total number of scheduled regular season games is approximately 1230. As may be seen in our chosen dataset (elaborated in the coming sections), there are statistics associated with each game played (in every season). This contains general information like the number of points scored by the Home/Visitor Team, number of rebounds, assists, free throws, etc.. In addition to this, the dataset also comprises the further breakdown of each game - in terms of player-wise performance of both the Home and Visitor Team players, who participated in a particular game. This gives us detailed information about the performance of players, and aggregation queries can also help us understand the performance of the teams per season (or across all seasons).

The motivation behind the selection of this dataset was the huge amount of data available and the kind of interesting insights that we can derive from it. With this dataset, we can dive deeper to understand 3 components - Games, Teams and Players. Aggregation queries can help us in understanding trends and deriving insights.

Our web application 'NBA Game Time' aims to bring about interesting information and analysis from the NBA dataset (dated from 2000-2018) and present the statistics in a user-friendly and easy-to-navigate application.

Introduction and Project Goals

For the purpose of this project, our team wanted to choose a dataset which was relevant and captured our interest, in addition to being robust and consisting of records and tables that would allow us to derive interesting information.

The NBA Games dataset, as available on Kaggle (originally sourced from the official website nba.com) satisfied our requirements. Like in any other sport, injuries are a relatively common occurrence and so, in addition to information about games, teams and players, we were also interested in understanding the Injuries that occurred for a particular player. For this, we used another dataset available - NBA Injuries, also available on Kaggle (originally sourced from prosportstransactions.com).

With this project, we aim to present this huge amount of NBA related data, in a format which makes it easy for Basketball-lovers to navigate and gain information about their favorite team/player and understand their performance in a particular game/season. The application also provides additional information about Players (such as Player Height, Player Weight, Home Country, etc..) and Teams (Team Owner, Manager, Arena, etc..) - which are useful and interesting data points as well.

Further, having data at such a scale, also opens the realm of possibilities to derive interesting insights. For instance, analytics about Team and Player Performance can really help guide Players, Coaches and Teams in understanding their current level performance, while allowing them to compare it against their average/most recent performance.

Our web application captures several interesting insights from the data - Star Players for a particular team, performance of a particular player against any particular team, average performance of the players in a team for a particular season, etc.. Such analysis can be very useful in understanding the trends for teams and players across seasons and games.

Another important aspect of our project goal was ensuring that the web application should be easy to navigate, visually appealing and an easy-go-to-reference for Basketball lovers.

Data Sources and Technologies used:

We have collected data from the following websites:

1. https://www.kaggle.com/nathanlauga/nba-games

2. https://www.kaggle.com/justinas/nba-players-data
3. https://www.kaggle.com/ghopkins/nba-injuries-2010-2018
4. https://www.kaggle.com/mharvnek/nba-team-stats-00-to-18

Using these datasets, we have created 7 tables as described below.

Data:

(Note: The final tables are created after extensive preprocessing of the csv files)

1. https://www.kaggle.com/nathanlauga/nba-games:
    a. teams.csv - Contains information about all the teams and their Team_ID. This was used to create table – Team
    b. players.csv - Contains information about all the players and their Player_ID. This was used to create table – Player
    c. games.csv - Contains information about all the games played in the NBA and their Game_ID. This was used to create table - Game
    d. games_details.csv - Contains detailed information about the performance of each player in each game. This was used to create table - GameStats
2. https://www.kaggle.com/justinas/nba-players-data:
    a. all_seasons.csv - Contains information about a player's cumulative performance statistics in every season. This was used to create table - PlayerStats
3. https://www.kaggle.com/ghopkins/nba-injuries-2010-2018:
    a. injuries_2010-2020.csv - Contains information about every injuries that any player has made with the date and details. This was used to create table - Injuries
4. https://www.kaggle.com/mharvnek/nba-team-stats-00-to-18:
    a. nba_team_stats_00_to_18.csv - Contains information about a team's performance and statistics after every season. This was used to create table - TeamStats

Technologies Used :

For creating Database :

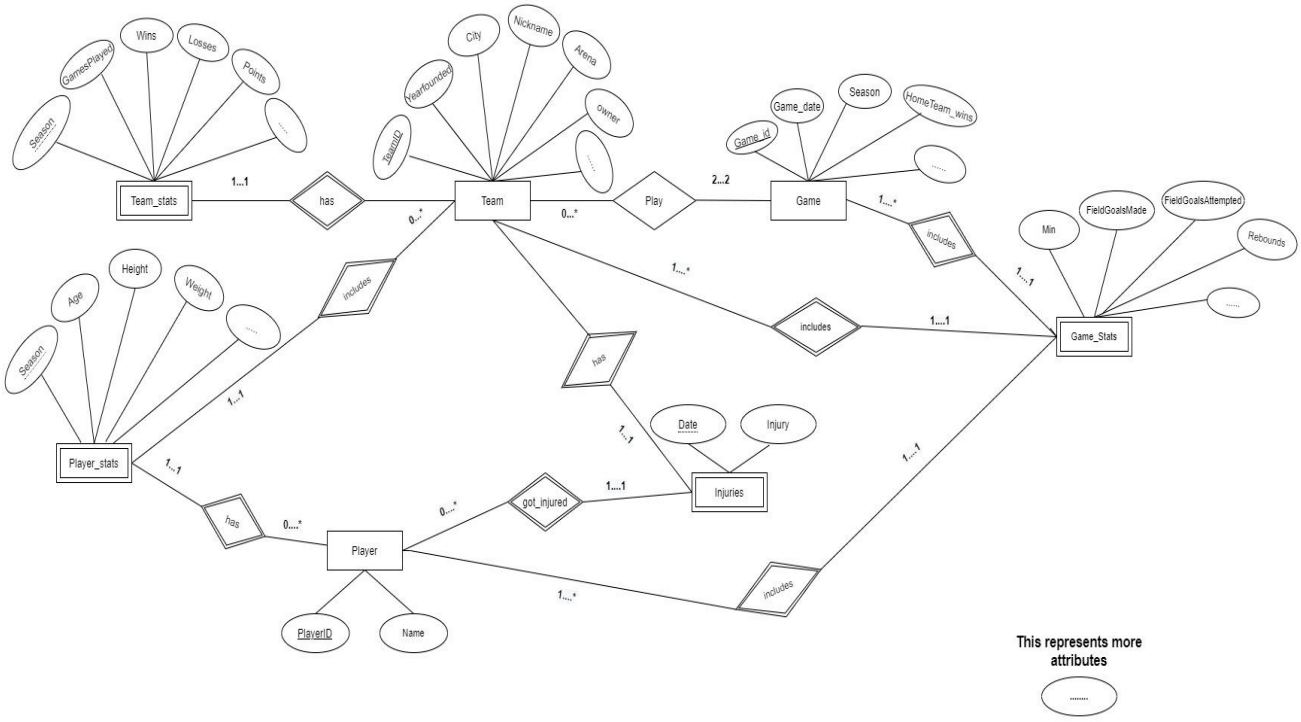
- We store our data in a MySQL database (using MySQL - Connector hosted on AWS).
- To access our database, we have extensively used DataGrip.

For creating the application :

- For creating the web application Javascript was used. In particular Node.js and React.js for backend and frontend respectively were used.
- HTML/ Bootstrap and CSS to make the website more aesthetic and pleasing to look at.
- IDE used was Visual Studio Code editor.

Relational Schema

ER Diagram :



Our database consists of 7 relational tables which are as follows:

- Teams Table
  Schema : (**Team_ID**, Abbreviation, Year_Founded, Team_Name, Arena, Arena_Capacity, Owner, General_Manager, HeadCoach, DLeague_Affiliation)
  No. of instances : 30

- TeamStats Table
  Schema : (***Team_ID*, Season**, GP,W, L, Win_Percentage, Min, Pts,FGM, FGA, FG_Percentage, 3PM, 3PA, 3P_Percentage, FTM, FTA, FT_Percentage, OREB, DREB, REB, AST, TOV, STL, BLK, BLKA, PF, PFD, Plus_Minus,**Team_ID FOREIGN KEY REFERENCES Team (Team_ID)**)

No. of instances: 520

- Players Table
  Schema : (**Player_ID**, Player_Name)
  No. of instances :1,769

- PlayerStats Table
  Schema : (**Player_ID, Season**, Team_ID, Age, Player_Height, Player_weight, College, Country, Draft_Year, Draft_Round, Draft_Number, Gp, Pts, Reb, Ast, Net_Rating, Oreb_Pct, Dreb_Pct, Usg_Pct, Ts_Pct, Ast_Pct, **Player_ID FOREIGN KEY REFERENCES Player(Player_ID), Team_ID FOREIGN KEY REFERENCES Team(Team_ID)**)

  No. of instances : 7,958

- Game Table
  Schema **: (Game_ID**,Game_Date_EST, Home_Team_ID, Visitor_Team_ID, Season, PTS_Home, FG_PCT_Home, FT_PCT_Home, FG3_PCT_Home, AST_Home, REB_Home, PTS_Away, FG_PCT_Away, FT_PCT_Away, FG3_PCT_Away, AST_Away, REB_Away, Home_Team_Wins, **Home_Team_ID FOREIGN KEY REFERENCES Team(Team_ID), Visitor_Team_ID FOREIGN KEY REFERENCES Team(Team_ID)**)

  No of instances : 24,523

- GameStats Table
  Schema : (**Game_ID,Team_ID, Player_ID**, Min, FGM, FGA, FG_PCT, FG3M, FG3A, FG3_PCT, FTM, FTA, FT_PCT, OREB, DREB, REB, AST, STL, BLK, TO, PF, PTS, Plus_Minus,Game_ID **FOREIGN KEY REFERENCES Game(Game_ID), Team_ID FOREIGN KEY REFERENCES Team(Team_ID), Player_ID FOREIGN KEY REFERENCES Player(Player_ID)**)

  No. of instances : 447,224

- Injuries Table
  Schema: (**Player_ID**, Date, Team_ID, Injury_Info, **Team_ID FOREIGN KEY REFERENCES Team (Team_ID)**)

  No of instances : 11,187

Normalization of Database

As can be seen from the description of datasets, the data for this project was sourced from different data sources. So, we reorganized the dataset in a manner that would allow us to easily link relations from different datasets.

Example, While the Players table had PlayerID as an attribute already present in the table, the Player Injuries dataset did not. To tackle this problem, PlayerID is included in another column in both the PlayerStats table and the Player Injuries table. And, the Player Name column was dropped to avoid redundancies.

This same approach was also followed for the Teams Table. Here, we have the Team ID in the Teams table along with other attributes such as Team Name, etc.. In all other tables referencing Teams, such as Team Stats, Game Stats, Injuries, etc.. we only included the Team ID and dropped the Name columns to prevent redundancy.

Some of the Functional Dependencies -
Player_ID -> Player_Name
Team_ID -> Team_Name, Arena, General Manager, Affiliation, ...
Player_ID, Season -> Player_Height, Player_Weight, Gp, Pts, Reb, ... {This allows us to see season wise performance of the player}

BCNF Checking:

Conditions for BCNF are -
For every relation schema and for every functional dependency X->A, X is a superkey of R.

As can be seen from the reorganized schema, the tables are in BCNF. Example, in the Players Table with dependency Player_ID -> Player_Name, Player_ID is the superkey of the table. Similarly, in this functional dependency Team_ID -> Team_Name, Arena, General Manager, Affiliation, ..., Team_ID is the superkey. So, the tables are in BCNF.

System Architecture

Below are descriptions of our application portal service web pages with information about what the respective pages accomplished -

Welcome Page :

This is the starter page of our web application which displays a welcome message,random basketball and NBA related images and option to navigate to other pages.

Home Page:
This is the main home/dashboard page displaying three cards - Teams,Players,Games. From here we can go towards more informative pages in our application. By clicking on Teams we land on to the Team's page having list of Teams,by clicking on players we land on to players page having list of players,and by clicking on Games we land on to games page having a list of all games.

Teams Page:
This page will display the names of all the teams on the Teams page.When the user clicks on a particular Team Name it will direct them to that Team's TeamInfo page.

TeamInfo Page:
This page has 5 tabs,each tab displaying different information about the team

Team Info -> Overview Tab :
This tab will display the information about a particular team such as its logo,arena,manager,coach,year founded etc. This is the default tab which gets opened when routed to the TeamInfo page from the Teams page.

TeamInfo -> Team Statistics Tab:
This tab would display the average performance statistics of a particular team. It will have an option to check the performance of the team for every season it has played in. The season can be selected from a dropdown menu.

TeamInfo -> Team Performance Tab :
This tab displays the information about a team's performance against a particular team and a season.It will have an option to select the opponent team,the season in which we want to see the performance and also,the outcome of the game - whether the team won or lost. The results will be displayed in a table format having various columns such as

TeamInfo -> Team Players Tab:
The tab will display names of all the players that have played in the team. This tab will have a search bar to search the names of players and a season dropdown to display results of names of players who played in the team for that season.

TeamInfo -> Star Players Tab:
This tab displays the information about the best season of the team ,along with its star players and max points scored by the team.

Players Page:
When a user clicks on the Players tile from the home page, this page opens up. This would display a list of all the (distinct) players in our database. We will also have the ability to filter the results by the PlayerName, TeamName and Season. If a user does not select any values for the filter, it would display all the players. When a user clicks on a particular player, they can see more detailed information about the player (PlayerInfo Page).

PlayerInfo Page:
This page opens up when the user clicks on the name of a particular player. Here, there would be 4 tabs on the top - Overview, Injuries, Performances and Player Stats. Default would open up to the Overview Tab.

PlayerInfo -> Overview Tab :
Here, information like Age, Weight, Height, Team, College, Country, etc.. about the player is displayed.

PlayerInfo -> Injuries Tab :
Here, information about the injuries of the player will be displayed. This would include the date of the injury and a brief description about the injury that took place.

PlayerInfo -> Performances Tab :
This tab shows the performance statistics of a player. A drop down to choose a season would be displayed based on which the statistics (for seasons) can be displayed. By default, the value of dropdown would be 'Aggregate performance', which would display aggregates of the performance of the player across all seasons that he/she played in. The user can change the value of the dropdown to select a particular season, which would display the performance of the player in that particular season.

PlayerInfo -> Player Vs Team Performance Tab :
This tab shows how the player has performed against an opponent team. This tab displays two tables and a dropdown menu to select the opponent team. First table shows the average performance of the player against an opponent team whereas the second table

shows each instance of the games where the player played against that team.By default,the page displays these statistics for all the teams the players have played against. From the dropdown menu, a particular team can be selected and the relevant statistics will be displayed accordingly.

PlayerInfo -> Player's Team Performance Tab :
This tab will display the points contribution by the player for the teams whom he/she was a part of for more than 2 seasons and during that time while the player was a part of the team, the team won the games it played.

Games Page:
When a user clicks on the Games tile from the home page, this page opens up. This would display a list of all the games played by every team. This page also provides a search bar for searching for home teams and away teams for the game. By clicking on the game it will navigate to the Games Info page.

GameInfo Page :
This page will display the detailed information about the individual games and teams who played the game. It is has 5 different tabs displaying different aspects of the game :

GameInfo -> Game overview Tab:
This tab displays the points scored by the home and away team of the game along with the results of the game.

GameInfo -> HomeTeamStats Tab :
This tab displays the points scored by each player of the home team in the game and their respective contribution.

GameInfo -> AwayTeamStats Tab :
This tab displays the points scored by each player of the away team in the game and their respective contribution.

GameInfo -> HomeTeam Season Stats Tab :
This tab displays the points scored by each player of the home team in the entire season when this game was played. This will help us to compare and assess the performance of the player in the season and that particular game.

GameInfo -> AwayTeam Season Stats Tab :
This tab displays the points scored by each player of the away team in the entire season when this game was played. This will help us to compare and assess the performance of the player in the season and that particular game.

Queries:

For a detailed look at the complex queries, please refer to:
https://docs.google.com/document/d/1tnl-ZbQ8GCkmaYuVHc4cyS5ifkTi9gt_D9FPOw5Bulg/edit?usp=sharing

How we addressed required features:

We combined several datasets from different sources and did some preprocessing. The data was displayed in a user-friendly format through our application.

We generated complex queries to provide interesting functionalities by combining different tables and optimized the running times by indexing and carefully choosing join orders and selections.

We built a complex architecture incorporating many relevant technologies:
React, Node and SQL

Performance evaluation:

After getting the dataset we observed that the dataset had a lot of redundancy. We carried out a lot of preprocessing to ensure that each table only had data that was required.

The size of our data was very huge and hence we had to carry out certain optimizations to bring down the running time of our complex queries.

Some Complex Query Descriptions	Original	Optimised
Player Performance vs a particular opponent	2s 62ms	676ms
Player Performance in a team for more than 2 seasons	3s 362ms	633ms
Finding the star performers of a particular team across different seasons, alongwith their best performance (in points scored)	2s 499ms	749ms
Average performance of the Game players in that Particular season	3s 67ms	813ms

Optimizations:

1. Used Inner Joins instead of Cartesian Products
2. Projecting only the required attributes and moving it as inside as possible
3. Created 2 indexes on GameStats - one on Game_ID and the other on Team_ID:
4. Extracting only the seasons for particular team very early on
5. Index on Game table to find seasons faster

The combined effect of all these steps lead to a significant drop in running times as visible from the table above.

<u>Technical Challenges and how they were overcome:</u>

The first challenge we faced was because of diverse data sources. Since we picked our database from four different sources, we had to deal with different formats for each source and accordingly had to incorporate additional features in some tables, while also removing redundant features in some other tables. We also had to perform a decent amount of data cleaning and pre-processing for which we used elite libraries like numpy and pandas. Our resultant dataset had a few large tables, for example one of the final tables had around 400,000 rows, and insertion of this table seemed problematic at first. After persevering through all errors and multiple tries, the data was finally uploaded. We also faced an issue while thinking about the complex queries since most of our tables were self-sufficient and contained all the relevant information. We still created multiple complex queries that show

interesting results for the web app users. Lastly, all of us were beginners at web development, hence, we faced quite a few issues during the frontend development phase due to our lack of experience. To solve this, we devoted a lot of time and effort to it and made sure we had a presentable and aesthetically pleasing web application in the end.

Appendix

Appendix A : *Data Repositories*

1. For tables - Games, GameStats, Players, Teams
   https://www.kaggle.com/nathanlauga/nba-games
2. For table - PlayerStats
   https://www.kaggle.com/justinas/nba-players-data
3. For table - Injuries
   https://www.kaggle.com/ghopkins/nba-injuries-2010-2018
4. For table - TeamStats
   https://www.kaggle.com/mharvnek/nba-team-stats-00-to-18