CIS 550: Database and Information Systems

Final Project: NBA Central

Team Member:
Weichen (Eric) Song
Ziyang (Sid) Zhang
Chenkang (Stephen) Zhang
Runqi (Vickie) Liu
Group Number: 45

GitHub Link: https://github.com/sidzzzzz3/cis550
Demo Video Link:
https://drive.google.com/file/d/1wOUZyHVqwhJOvw81cpCFXWYDVI3-2DdA/view?usp=sharing

---

Part 1: Introduction

1.1 Project Goal & Motivation

The motivation behind the project is to create a dynamic and user-friendly website application serving as a centralized platform for accessing NBA data from session 2003 to session 2021. The goal is to provide basketball enthusiasts with an interactive platform where they can explore comprehensive player profiles, team information, and detailed game statistics.

1.2 Application Functionality

Through interactive interfaces, users will be able to explore comprehensive player profiles, including per game statistics and highlights. Team information will provide insights championship wins and performance metrics. Detailed game statistics will be available for every NBA game played during this timeframe, allowing users to analyze player performances, team statistics, and game summaries. By incorporating interactive elements such as filters and visualization tools, the website aims to provide an engaging and informative experience for basketball enthusiasts to approach NBA with numerical support.

Part 2: Architecture

2.1 Technology Used

- Exploratory Data Analysis & Data Processing: Python: Pandas, Matplotlib
- Back End
    - JavaScript: Node.js, Express.js

    - MySQL
    - AWS RDS
- Front End
    - JavaScript: React, Material UI, Recharts

2.2 Description of system architecture/application

The architecture of our NBA data-centric web application leverages a combination of backend, frontend, and data processing technologies to deliver a dynamic and user-friendly platform for basketball enthusiasts. Below is an overview of the technologies used in each layer:

Exploratory Data Analysis & Data Processing

- Python: Used for data manipulation, exploratory data analysis (EDA), and feature engineering.
- Pandas: Pandas is employed for efficient data processing, manipulation, and entity resolution
- Matplotlib: Matplotlib facilitates data visualization to explore the characteristics of the dataset.

Backend

- JavaScript: JavaScript serves as the primary programming language for the backend, enabling us to build robust and scalable server-side applications.
- Node.js: Node.js provides the runtime environment for executing JavaScript code on the server side, offering non-blocking, event-driven architecture for high performance and it is the key for implementing our routes
- Express.js: Express.js is utilized as the web application framework for Node.js, simplifying the development of RESTful APIs and handling HTTP requests/responses.
- MySQL: MySQL is employed as the main data retrieval engine, where we write queries to retrieve data from the stored database. This helps to establish routes and return corresponding data used by the front end.
- AWS RDS: Amazon Web Services Relational Database Service is utilized for hosting MySQL queries and our NBA data in the cloud, offering scalability, reliability, and security features.

Frontend

- JavaScript: JavaScript is used for building interactive and dynamic user interfaces on the client side.
- React: React.js is employed as the frontend JavaScript library for building reusable UI components, enabling us to create a responsive and modular frontend application.
- Material UI: Material UI provides pre-designed React components and styles based on Google's Material Design, enhancing the visual appeal and user experience of the application. Lots of our page features used this, for example the search, the filter, etc.
- Recharts: Recharts is utilized for creating interactive and customizable charts and graphs, allowing users to visualize NBA data trends effectively.

Part 3: Data

3.1 Dataset Link

In total, we are using 5 datasets from:
https://www.kaggle.com/datasets/nathanlauga/nba-games?select=games.csv.

3.2 Data Description & Raw Data Statistics

Games: 26651 rows, 21 columns
This dataset contains information about all NBA games ranging from season 2003 to season 2021. Each row in the dataset exposes a game's statistics, such as two opposing teams' overall final points, rebounds, assists, field goal percentage, etc.

Ranking: 210342 rows, 13 columns
This dataset contains information about ranking statistics for each team for any given day in season 2003 to season 2021. Each row in the dataset exposes a team's ranking information, such as number of winning games, loss games, standing in either West or East conferences.

Game_Details: 668628 rows, 29 columns
This dataset contains information about all NBA games' details ranging from season 2003 to season 2021. Each row in the dataset exposes each player's (by name and by team) statistics, such as the points this player made, his rebounds statistics, and his free throw percentage, etc.

Players: 7228 rows, 4 columns
This dataset contains information about all NBA players' information ranging from season 2003 to season 2021. Each row in the dataset exposes each player (by player ID)'s name, team for a specific season.

Teams: 30 rows, 14 cols
This dataset contains information about all NBA team's detail, specifically, there are 30 teams from both West and East conferences. Each row in this dataset exposes the information about a team's found year, team's home court, and last year it entered the champions.

The column details and data statistics for each of these raw tables can be found in Appendix A.

3.4 How Data is used

We first inspected the data, performed preliminary EDA to understand our data, and cleaned the data by removing incorrect and unnecessary null values, and performed necessary feature engineering. In particular, we removed information that is incomplete (i.e., 2002 season-related information), and added a feature to distinguish between in-season and postseason games, which is helpful for our future queries. Certain NULL values are intentionally left as suggested by the 'Project Tips pdf' and will be addressed in the queries. Following that, we processed the datasets to ensure that all of the relations are in 3NF (details in Part 4). The code used to do data-processing and 3nf decomposition can be found in our GitHub with

name: Data Pre-processing & 3NF Conversion.ipynb. Lastly, we uploaded the data to MySQL (with MySQL-Connector hosted on AWS), which became readily available to use.

3.5 Datasets used relation

Games Dataset:

- Relationship with Ranking Dataset: Linked through team identifiers, enabling correlation of game outcomes with team rankings.
- Relationship with Game_Details Dataset: Connected via game identifiers, allowing retrieval of detailed player statistics for each game.

Ranking Dataset:

- Relationship with Games Dataset: Associated through team identifiers, facilitating analysis of team performance relative to rankings.

Game_Details Dataset:

- Relationship with Games Dataset: Connected via game identifiers, enabling aggregation of player-level statistics for each game.
- Relationship with Players Dataset: Linked through player identifiers, allowing retrieval of player information associated with game statistics.

Players Dataset:

- Relationship with Game_Details Dataset: Associated via player identifiers, enabling correlation of player profiles with game statistics.
- Relationship with Teams Dataset: Linked through team affiliations, facilitating identification of players associated with specific teams.

Teams Dataset:

- Relationship with Players Dataset: Linked through team affiliations, enabling retrieval of players associated with each team.
- Relationship with Games Dataset: Associated via team identifiers, facilitating analysis of team performance across games.

These relationships form the foundation for querying and analyzing NBA data within the application, allowing users to explore player performances, team dynamics, and game outcomes comprehensively.

Part 4: Database

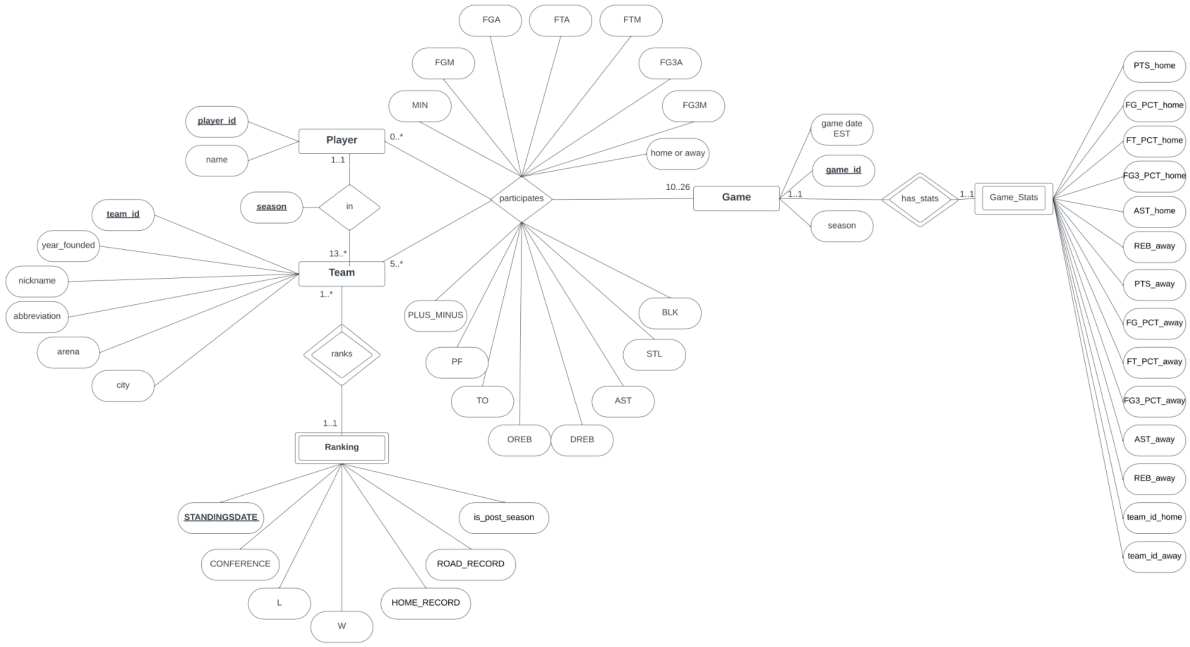
4.1 Explanation of data ingestion procedure and entity resolution efforts

With the data imported and cleaned, we identified that some of our relations have transitive / partial dependency problems. For example, the original Player table has PLAYER_ID -> PLAYER_NAME and which is a partial dependency because PLAYER_ID is part of the candidate key of (PLAYER_ID, TEAM_ID, SEASON). Therefore we normalized the data to become 3NF by identifying the minimum set of functional dependencies within each relation and removing any transitive dependencies by breaking them into smaller relations when applicable. The final relational schemas are:

4.2 Normalized Schema & Table Size

Player(PLAYER_ID, PLAYER_NAME) [2678 * 2]
Team(TEAM_ID, ABBREVIATION, NICKNAME, YEARFOUNDED, CITY, ARENA)[30 * 6]
Roster(TEAM_ID, PLAYER_ID, SEASON) [13129 * 3]
    TEAM_ID foreign key referencing Teams(TEAM_ID)
    PLAYER_ID foreign key referencing Player(PLAYER_ID)
Game(GAME_ID, GAME_DATE_EST, SEASON) [26622 * 3]
Game_Stats_Team(GAME_ID, TEAM_ID_home, PTS_home, FG_PCT_home, FT_PCT_home, FG3_PCT_home, AST_home, REB_home, TEAM_ID_away, PTS_away, FG_PCT_away, FT_PCT_away, FG3_PCT_away, AST_away, REB_away) [26622 * 15]
    GAME_ID foreign key referencing Games(GAME_ID)
Game_Stats_Player(GAME_ID, TEAM_ID, PLAYER_ID, MIN, FGM, FGA, FG3M, FG3A, FTM, FTA, OREB, DREB, AST, STL, BLK, TO, PF, PLUS_MINUS, home_or_away)[668338 * 19]
    GAME_ID foreign key referencing Games(GAME_ID)
    TEAM_ID foreign key referencing Teams(TEAM_ID)
    PLAYER_ID foreign key referencing Player(PLAYER_ID)
Ranking(TEAM_ID, STANDINGSDATE, CONFERENCE, W, L, HOME_RECORD, ROAD_RECORD, is_post_reason) [195370 * 8]
    TEAM_ID foreign key referencing Teams(TEAM_ID)

4.3 ER diagram



4.4 3NF Proof

1NF is not proved because all attributes are atomic. For each relation below, 3NF will be shown by proving there is no partial or transitive dependency.

Player(PLAYER_ID, PLAYER_NAME)
FD: PLAYER_ID -> PLAYER_NAME

CK is {PLAYER_ID}

No partial dependency because LHS is not a proper subset of the CK

No transitive dependency because LHS the superkey.

Team(TEAM_ID, ABBREVIATION, NICKNAME, YEARFOUNDED, CITY, ARENA)
FD: TEAM_ID -> ABBREVIATION, NICKNAME, YEARFOUNDED, CITY, ARENA
ABBREVIATION -> TEAM_ID, NICKNAME, YEARFOUNDED, CITY, ARENA
NICKNAME -> TEAM_ID, ABBREVIATION, YEARFOUNDED, CITY, ARENA

There are three CK: {TEAM_ID}, {ABBREVIATION}, {NICKNAME} (each is a superkey)

No partial dependency because none of the LHS is a proper subset of the CK.

No transitive dependency because all LHS are superkeys.

Roster(TEAM_ID, PLAYER_ID, SEASON)

Since all attributes are part of the primary key, it is guaranteed to be in 3NF, there are no non-prime attributes.

Game(GAME_ID, GAME_DATE_EST, SEASON)
FD: GAME_ID -> GAME_DATE_EST, SEASON

CK is {GAME_ID}

No partial dependency because LHS is not a proper subset of the CK

No transitive dependency because LHS the superkey

Game_Stats_Team(GAME_ID, TEAM_ID_home, PTS_home, FG_PCT_home, FT_PCT_home, FG3_PCT_home, AST_home, REB_home, TEAM_ID_away, PTS_away, FG_PCT_away, FT_PCT_away, FG3_PCT_away, AST_away, REB_away)
FD: GAME_ID -> TEAM_ID_home, PTS_home, FG_PCT_home, FT_PCT_home, FG3_PCT_home, AST_home, REB_home, TEAM_ID_away, PTS_away, FG_PCT_away, FT_PCT_away, FG3_PCT_away, AST_away, REB_away

CK is {GAME_ID}

No partial dependency because LHS is not a proper subset of the CK

No transitive dependency because LHS the superkey

Game_Stats_Player(GAME_ID, TEAM_ID, PLAYER_ID, MIN, FGM, FGA, FG3M, FG3A, FTM, FTA, OREB, DREB, AST, STL, BLK, TO, PF, PLUS_MINUS, home_or_away)
FD: GAME_ID, TEAM_ID, PLAYER_ID -> MIN, FGM, FGA, FG3M, FG3A, FTM, FTA, OREB, DREB, AST, STL, BLK, TO, PF, PLUS_MINUS, home_or_away

CK is {GAME_ID, TEAM_ID, PLAYER_ID}

No partial dependency because LHS is not a proper subset of the CK

No transitive dependency because LHS the superkey

Ranking(TEAM_ID, STANDINGSDATE, CONFERENCE, W, L, HOME_RECORD, ROAD_RECORD, is_post_reason)
FD: TEAM_ID, STANDINGSDATE -> CONFERENCE, W, L, HOME_RECORD, ROAD_RECORD, is_post_reason

CK is {TEAM_ID, STANDINGDATE}

No partial dependency because LHS is not a proper subset of the CK

No transitive dependency because LHS the superkey

Part 5: Web App Description

HomePage
The page displays the overview of the application. On the top, there are different sections to select: Player, Team, Championship, Trend, Games, and Login. Then there will be three sections. The first section is the acknowledgement, which displays the author, motivation, and description of the project. The second section displays a total of 30 NBA teams, with corresponding information such as the team name, founding year, city, and logo. Users can click on the team name to see the detailed information about the team. Another section will be the search. This section will display the players'/teams' detailed information that meet the search criteria, such as name, points, etc.

Championship Page
The page displays the champion team for each season, from 2003 to 2021. The website displays the championship information on the year 2021 by default. If a user would like to learn about other year's information, they can select the corresponding year in the search bar.

Player Page
The page lists the details of the specific player. The player name will be on the top of the page, with his corresponding details such as his current team, his average game statistics in the given season.

Team Page
The page lists the details of the specific team. The team name and team logo will be on the top of the page, with its essential details such as the team founded year, team's arena etc. Below, there will be a section showcasing the current roster of players, categorized by season . Moreover, there will be a section displaying the detailed statistics from past games in which the team participated, also organized by season.

Games Page
The Games Page enables users to view all games played on a specific date. Users can enter a specific date to retrieve a list of all games that occurred on that day.

Trend Page
The page features various line graphs illustrating the statistical trends of players. It is divided into distinct sections, allowing users to explore player-level trends with options to view a player's latest 5 games' metrics (both player name and metrics can be specified by user), ranking compared to other players in the same game, and the game over game metrics percentage change. In addition, it provides a count of the teams and games count in his entire career.

Part 6: API Specification

We have attached our API specifications as submitted in milestone 4 in Appendix D.

Part 7: Queries

The table below shows 14 queries (>= 6 simple and >= 4 complex) and their corresponding page and short description. **The SQL code can be found in Appendix B.** Some techniques we have used for but are not shown in the table include extensive use of case-when clause, and windows functions (rank, lag).

No	*Simple / Complex*	*Page*	*Description*	*Multiple Joins*	*Subqueries / Views*	*Aggregate*	*Universal / Existential Check**	*>15s runtime*
1	Complex	Championship	Get the championship information of the year	✔	✔	✔	✔	
2	Simple	Games	Get the games information on the date	✔	✔			
3	Simple	Home	Get all NBA teams with their key information like name, abbreviation, arena, Logo,, etc.	✔	✔			
4	Complex	Home	Search & Retrieve Players by name, season, dates, results, points, assists, stoles, etc.	✔	✔			✔
5	Complex	Home	Search & Retrieve Teams by name, season, dates, results, points, assists, stoles, etc.	✔	✔	✔		
6	Simple	Home	Retrieve all teams name so that enable the feature of searching team by selecting name options instead of entering name manually (no blur searching)					
7	Simple	Team	Obtain all the game information of a selected team in a selected season	✔	✔	✔		
8	Simple	Player	Obtain all the games and statistics a selected player played in a selected season	✔		✔		
9	Simple	Player	Retrieve the performance of two players in a range of seasons	✔	✔	✔		
10	Simple	Trend	Count of the teams the player played in			✔		
11	Simple	Trend	Count the number of games a player played			✔		
12	Simple	Trend	Get the names of players that have played in at least one game; used to generate dropdown list			✔		
13	Complex	Trend	Return the player's metric's score ranking relative to other players in the same game for the latest 5 games played; shown as line plot	✔	✔		✔	✔
14	Complex	Trend	Return the player's metric's score in the latest 5 games played and the metric's game over game percentage change; shown as line plot	✔	✔			✔

Table 1. Queries and descriptions,
**the use of IN and NOT IN*

Complex 14 is a complex query that is used in the trend page to return a player's metric score in the latest 5 games and the game over game metric % change, where the player and the metric are query parameters. It is a complex query as it involves multiple joins, subqueries, case when clause, windows function (lag), and has a >15s pre-optimized runtime.

Part 8: Performance evaluation

Screenshots showing the pre-optimization query (SQL code) and timings can be found in Appendix C.

Query No.	***Pre Time***	***Post Time***	***Optimization Technique Used & Explanation***
1	~1s	~0.5s	**Restructuring Query**: Select the necessary season before join; filter the table to reduce the intermediate size.
4	~25s	~1s	**Restructuring Query**: change the order of order by GAME_DATE and filtering clauses such that order by comes later and ordering is done on fewer rows & faster **Caching:** created table (because views are not supported by mySQL as suggested by ed #1495) for the part of the query that is unaffected by the query (named as Wrapper), avoid joining of Game_Stats_Player, Game_Update_Matches, Player, and Team upon every iteration and thus saves time
5	~1	~0.5s	**Caching:** created table (because views are not supported by mySQL as suggested by ed #1495) for the part of the query that is unaffected by the query (named as Wrapper_Team), avoid joining of Game_Stats_Team, Team, Game upon every iteration and thus saves time
12 (simple query)	~5s	~0.5s	**Caching:** Since the query doesn't depend on user query input and output size is small, we create a table to store the result (simulate materialized view in MySQL) in Player_gtone_game, this greatly boosts the performance because all the joining, group by, and having clauses are done only once (when the table was created) and no longer needed to be processed at runtime.
13	~16s	~1s	**Restructuring Query**: push down name selection to reduce the number of games selected, greatly reduced intermediate CTE size, effectively reduced time **Indexing:** Adding Player(name) and Game_Stats_Player(player_id) indexes, same function and reason as that in Query 12 below.
14	~75s	~1s	**Restructuring Query**: 1. Push down selection (player name and metrics needed) to reduce the size of the intermediate tables 2. Avoid recalculating windows function (lag function result) because windows function involves sorting and ranking which is time-consuming; with restructuring, we were able to reduce the time from ~70s to ~5s **Caching:** we tried to create a table to simulate materialized view but because the static component (Game_Stats_Player_with_Total) only involved simple projection and selection, it did not help with performance and therefore is excluded in the final submission; all the other subqueries are unsuitable for caching as they involve dynamic query parameters. **Indexing:** Improve performance from ~5s to ~1s 1. Create index for Player(name) helpful for carrying out where filtering (`WHERE name =`) via index range scan. 2. Create index for Game_Stats_Player(player_id), helpful for JOIN (`ON g_s.player_id = p.PLAYER_ID`) via index unique scan.

Table 2. Optimization for complex queries

**Console times are recorded, >=3 runs are done to avoid anomalous data.*

Difficulty: It is challenging to choose the right combination of strategies for each query to achieve the best result as not all strategies work on all queries; in addition, thinking about what indexes to add to maximize the optimization, and potentially bring the advantages to multiple queries at the same time.

Part 9: Technical Challenges

There were some major technical challenges that we faced during the project. First, there was some incorrect information in the dataset. The Ranking table contains the game records for each team on many dates. We have previous knowledge that the regular NBA season usually starts around the end of October and ends around the beginning of April. However, we have a lot of data with dates between May and September, which is redundant. We had to come up with a formula to filter out these useless records.

Second, since we all had little experience with front end applications, the front end was a difficult part of this project. I spent a lot of time on the creation of the link on the team names on the home page. At the beginning, we used lazy tables, similar to what we did in the homework. Nonetheless, the lazy table has limited functionalities so I had to do some research to find a way to implement the link. In the end, I managed to use a rendercell function with a NavLink to make it work.

In addition, on the home page, we had a problem of displaying player data. Initially, the search always returned the same data for some reason which we could not figure out. We stuck on this for a long time and eventually we learned that for a DataGrid, we have to have a unique identifier for each row in order to show the proper table. We fixed this by assigning the gameID as the unique identifier.

Part 10: Extra Credit

We implemented the login feature.

Appendix A

- 1. games
  - Link: https://www.kaggle.com/datasets/nathanlauga/nba-games?select=games.csv
  - Description
    - This dataset contains information about all NBA games ranging from season 2003 to season 2021. Each row in the dataset exposes a game's statistics, such as two opposing teams' overall final points, rebounds, assists, field goal percentage, etc.
  - 26651 rows, 21 columns
  - Data Statistics:

```
In [18]: game.describe()
Out[18]:
             GAME_ID  HOME_TEAM_ID  VISITOR_TEAM_ID        SEASON  \
count  2.665100e+04  2.665100e+04     2.665100e+04  26651.000000
mean   2.175487e+07  1.610613e+09     1.610613e+09   2012.113879
std    5.570189e+06  8.638670e+00     8.659299e+00      5.587031
min    1.030000e+07  1.610613e+09     1.610613e+09   2003.000000
25%    2.070001e+07  1.610613e+09     1.610613e+09   2007.000000
50%    2.120076e+07  1.610613e+09     1.610613e+09   2012.000000
75%    2.180005e+07  1.610613e+09     1.610613e+09   2017.000000
max    5.210021e+07  1.610613e+09     1.610613e+09   2022.000000

       TEAM_ID_home      PTS_home   FG_PCT_home   FT_PCT_home  FG3_PCT_home  \
count  2.665100e+04  26552.000000  26552.000000  26552.000000  26552.000000
mean   1.610613e+09    103.455898      0.460735      0.760377      0.356023
std    8.638670e+00     13.283370      0.056676      0.100677      0.111164
min    1.610613e+09     36.000000      0.250000      0.143000      0.000000
25%    1.610613e+09     94.000000      0.422000      0.697000      0.286000
50%    1.610613e+09    103.000000      0.460000      0.765000      0.357000
75%    1.610613e+09    112.000000      0.500000      0.833000      0.429000
max    1.610613e+09    168.000000      0.684000      1.000000      1.000000

           AST_home      REB_home  TEAM_ID_away      PTS_away   FG_PCT_away  \
count  26552.000000  26552.000000  2.665100e+04  26552.000000  26552.000000
mean      22.823441     43.374284  1.610613e+09    100.639876      0.449732
std        5.193308      6.625769  8.659299e+00     13.435868      0.055551
min        6.000000     15.000000  1.610613e+09     33.000000      0.244000
25%       19.000000     39.000000  1.610613e+09     91.000000      0.412000
50%       23.000000     43.000000  1.610613e+09    100.000000      0.449000
75%       26.000000     48.000000  1.610613e+09    110.000000      0.487000
max       50.000000     72.000000  1.610613e+09    168.000000      0.687000

        FT_PCT_away  FG3_PCT_away      AST_away      REB_away  HOME_TEAM_WINS
count  26552.000000  26552.000000  26552.000000  26552.000000    26651.000000
mean       0.758816      0.349489     21.496271     42.113249        0.587032
std        0.103429      0.109441      5.160596      6.533039        0.492376
min        0.143000      0.000000      4.000000     19.000000        0.000000
25%        0.692000      0.278000     18.000000     38.000000        0.000000
50%        0.765000      0.350000     21.000000     42.000000        1.000000
75%        0.833000      0.419000     25.000000     46.000000        1.000000
max        1.000000      1.000000     46.000000     81.000000        1.000000
```

- Data Info:

```
In [19]: game.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 26651 entries, 0 to 26650
Data columns (total 21 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   GAME_DATE_EST     26651 non-null  object
 1   GAME_ID           26651 non-null  int64
 2   GAME_STATUS_TEXT  26651 non-null  object
 3   HOME_TEAM_ID      26651 non-null  int64
 4   VISITOR_TEAM_ID   26651 non-null  int64
 5   SEASON            26651 non-null  int64
 6   TEAM_ID_home      26651 non-null  int64
 7   PTS_home          26552 non-null  float64
 8   FG_PCT_home       26552 non-null  float64
 9   FT_PCT_home       26552 non-null  float64
 10  FG3_PCT_home      26552 non-null  float64
 11  AST_home          26552 non-null  float64
 12  REB_home          26552 non-null  float64
 13  TEAM_ID_away      26651 non-null  int64
 14  PTS_away          26552 non-null  float64
 15  FG_PCT_away       26552 non-null  float64
 16  FT_PCT_away       26552 non-null  float64
 17  FG3_PCT_away      26552 non-null  float64
 18  AST_away          26552 non-null  float64
 19  REB_away          26552 non-null  float64
 20  HOME_TEAM_WINS    26651 non-null  int64
dtypes: float64(12), int64(7), object(2)
memory usage: 4.3+ MB
```

- 2. gamesDetails
    - Link: https://www.kaggle.com/datasets/nathanlauga/nba-games?select=games_details.csv
    - Description
        - This dataset contains information about all NBA games' details ranging from season 2003 to season 2021. Each row in the dataset exposes each player's (by name and by team) statistics, such as the points this player made, his rebounds statistics, and his free throw percentage, etc.
    - 668628 rows, 29 columns
    - Data Statistics:

```
In [16]: game.describe()
Out[16]:
            GAME_ID       TEAM_ID     PLAYER_ID            FGM            FGA  \
count  6.686280e+05  6.686280e+05  6.686280e+05  558938.000000  558938.000000
mean   2.171771e+07  1.610613e+09  4.013434e+05       3.588446       7.896652
std    5.656289e+06  8.652260e+00  7.225618e+06       3.030466       5.677002
min    1.030000e+07  1.610613e+09  1.500000e+01       0.000000       0.000000
25%    2.070003e+07  1.610613e+09  2.466000e+03       1.000000       3.000000
50%    2.120096e+07  1.610613e+09  2.011810e+05       3.000000       7.000000
75%    2.180014e+07  1.610613e+09  2.034710e+05       5.000000      11.000000
max    5.210021e+07  1.610613e+09  1.962938e+09      28.000000      50.000000

              FG_PCT           FG3M           FG3A        FG3_PCT  \
count  558938.000000  558938.000000  558938.000000  558938.000000
mean        0.416842       0.778117       2.186019       0.201032
std         0.251913       1.227615       2.569913       0.289685
min         0.000000       0.000000       0.000000       0.000000
25%         0.267000       0.000000       0.000000       0.000000
50%         0.429000       0.000000       1.000000       0.000000
75%         0.571000       1.000000       4.000000       0.400000
max         1.000000      14.000000      24.000000       1.000000

                 FTM            FTA         FT_PCT           OREB  \
count  558938.000000  558938.000000  558938.000000  558938.000000
mean        1.733217       2.284212       0.435949       1.024212
std         2.353981       2.886583       0.428166       1.397830
min         0.000000       0.000000       0.000000       0.000000
25%         0.000000       0.000000       0.000000       0.000000
50%         1.000000       2.000000       0.500000       1.000000
75%         3.000000       4.000000       0.909000       2.000000
max        26.000000      39.000000       1.000000      18.000000

                DREB           REB            AST            STL  \
count  558938.000000  558938.00000  558938.000000  558938.000000
mean        3.033798       4.05801       2.103958       0.721436
std         2.687384       3.48250       2.475476       0.972231
min         0.000000       0.00000       0.000000       0.000000
25%         1.000000       1.00000       0.000000       0.000000
50%         2.000000       3.00000       1.000000       0.000000
75%         4.000000       6.00000       3.000000       1.000000
max        25.000000      31.00000      25.000000      10.000000
```

```
                 BLK             TO             PF            PTS  \
count  558938.000000  558938.000000  558938.000000  558938.000000
mean        0.460339       1.320297       1.999538       9.688218
std         0.860962       1.402329       1.502963       8.082152
min         0.000000       0.000000       0.000000       0.000000
25%         0.000000       0.000000       1.000000       3.000000
50%         0.000000       1.000000       2.000000       8.000000
75%         1.000000       2.000000       3.000000      14.000000
max        12.000000      12.000000      15.000000      81.000000

          PLUS_MINUS
count  535277.000000
mean       -0.000488
std        10.665573
min       -57.000000
25%        -7.000000
50%         0.000000
75%         6.000000
max        57.000000
```

- Data Info:

```
In [14]: game.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 668628 entries, 0 to 668627
Data columns (total 29 columns):
 #   Column             Non-Null Count   Dtype
---  ------             --------------   -----
 0   GAME_ID            668628 non-null  int64
 1   TEAM_ID            668628 non-null  int64
 2   TEAM_ABBREVIATION  668628 non-null  object
 3   TEAM_CITY          668628 non-null  object
 4   PLAYER_ID          668628 non-null  int64
 5   PLAYER_NAME        668628 non-null  object
 6   NICKNAME           53037 non-null   object
 7   START_POSITION     255765 non-null  object
 8   COMMENT            109689 non-null  object
 9   MIN                558938 non-null  object
 10  FGM                558938 non-null  float64
 11  FGA                558938 non-null  float64
 12  FG_PCT             558938 non-null  float64
 13  FG3M               558938 non-null  float64
 14  FG3A               558938 non-null  float64
 15  FG3_PCT            558938 non-null  float64
 16  FTM                558938 non-null  float64
 17  FTA                558938 non-null  float64
 18  FT_PCT             558938 non-null  float64
 19  OREB               558938 non-null  float64
 20  DREB               558938 non-null  float64
 21  REB                558938 non-null  float64
 22  AST                558938 non-null  float64
 23  STL                558938 non-null  float64
 24  BLK                558938 non-null  float64
 25  TO                 558938 non-null  float64
 26  PF                 558938 non-null  float64
 27  PTS                558938 non-null  float64
 28  PLUS_MINUS         535277 non-null  float64
dtypes: float64(19), int64(3), object(7)
memory usage: 147.9+ MB
```

- 3. players
    - Link: https://www.kaggle.com/datasets/nathanlauga/nba-games?select=players.csv
    - Description
        - This dataset contains information about all NBA players' information ranging from season 2003 to season 2021. Each row in the dataset exposes each player (by player ID)'s name, team for a specific season.
    - 7228 rows, 4 columns
    - Data Statistics:

```
In [23]: player.describe()
Out[23]:
            TEAM_ID     PLAYER_ID       SEASON
count  7.228000e+03  7.228000e+03  7228.000000
mean   1.610613e+09  2.355862e+06  2014.159934
std    8.723521e+00  6.106688e+07     3.126216
min    1.610613e+09  2.440000e+02  2009.000000
25%    1.610613e+09  2.007680e+05  2012.000000
50%    1.610613e+09  2.023465e+05  2014.000000
75%    1.610613e+09  2.039100e+05  2017.000000
max    1.610613e+09  1.962938e+09  2019.000000
```

    - Data Info:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7228 entries, 0 to 7227
Data columns (total 4 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PLAYER_NAME  7228 non-null   object
 1   TEAM_ID      7228 non-null   int64
 2   PLAYER_ID    7228 non-null   int64
 3   SEASON       7228 non-null   int64
dtypes: int64(3), object(1)
memory usage: 226.0+ KB
```

- 4. ranking
    - Link: https://www.kaggle.com/datasets/nathanlauga/nba-games?select=ranking.csv
    - Description
        - This dataset contains information about ranking statistics for each team for any given day in season 2003 to season 2021. Each row in the dataset exposes a team's ranking information, such as number of winning games, loss games, standing in either West or East conferences.
    - 210342 rows, 13 columns
    - Data Statistics:

	TEAM_ID	LEAGUE_ID	SEASON_ID	G	W	L	W_PCT	RETURNTOPLAY
count	2.103420e+05	210342.0	210342.000000	210342.000000	210342.000000	210342.000000	210342.000000	3990.000000
mean	1.610613e+09	0.0	21401.054773	56.659735	28.333357	28.326378	0.492833	0.600000
std	8.641501e+00	0.0	2395.250417	28.644294	17.268500	17.260557	0.187763	0.489959
min	1.610613e+09	0.0	12003.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	1.610613e+09	0.0	22006.000000	32.000000	14.000000	14.000000	0.372000	0.000000
50%	1.610613e+09	0.0	22011.000000	67.000000	28.000000	29.000000	0.500000	1.000000
75%	1.610613e+09	0.0	22017.000000	82.000000	42.000000	41.000000	0.621000	1.000000
max	1.610613e+09	0.0	22022.000000	82.000000	73.000000	72.000000	1.000000	1.000000

    - Data Info:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 210342 entries, 0 to 210341
Data columns (total 13 columns):
 #   Column         Non-Null Count   Dtype
---  ------         --------------   -----
 0   TEAM_ID        210342 non-null  int64
 1   LEAGUE_ID      210342 non-null  int64
 2   SEASON_ID      210342 non-null  int64
 3   STANDINGSDATE  210342 non-null  object
 4   CONFERENCE     210342 non-null  object
 5   TEAM           210342 non-null  object
 6   G              210342 non-null  int64
 7   W              210342 non-null  int64
 8   L              210342 non-null  int64
 9   W_PCT          210342 non-null  float64
 10  HOME_RECORD    210342 non-null  object
 11  ROAD_RECORD    210342 non-null  object
 12  RETURNTOPLAY   3990 non-null    float64
dtypes: float64(2), int64(6), object(5)
memory usage: 20.9+ MB
```

- 5. teams
    - Link: https://www.kaggle.com/datasets/nathanlauga/nba-games?select=teams.csv
    - Description
        - This dataset contains information about all NBA team's detail, specifically, there are 30 teams from both West and East conferences. Each row in this dataset exposes the information about a team's found year, team's home court, and last year it entered the champions.
    - 30 rows, 14 cols
    - Data Statistics:

	LEAGUE_ID	TEAM_ID	MIN_YEAR	MAX_YEAR	YEARFOUNDED	ARENACAPACITY
count	30.0	3.000000e+01	30.000000	30.0	30.000000	26.000000
mean	0.0	1.610613e+09	1969.700000	2019.0	1969.700000	18553.307692
std	0.0	8.803408e+00	16.698441	0.0	16.698441	3916.923362
min	0.0	1.610613e+09	1946.000000	2019.0	1946.000000	0.000000
25%	0.0	1.610613e+09	1952.000000	2019.0	1952.000000	18641.500000
50%	0.0	1.610613e+09	1970.000000	2019.0	1970.000000	19131.000000
75%	0.0	1.610613e+09	1979.000000	2019.0	1979.000000	19790.750000
max	0.0	1.610613e+09	2002.000000	2019.0	2002.000000	21711.000000

    - Data Info:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30 entries, 0 to 29
Data columns (total 14 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   LEAGUE_ID           30 non-null     int64
 1   TEAM_ID             30 non-null     int64
 2   MIN_YEAR            30 non-null     int64
 3   MAX_YEAR            30 non-null     int64
 4   ABBREVIATION        30 non-null     object
 5   NICKNAME            30 non-null     object
 6   YEARFOUNDED         30 non-null     int64
 7   CITY                30 non-null     object
 8   ARENA               30 non-null     object
 9   ARENACAPACITY       26 non-null     float64
 10  OWNER               30 non-null     object
 11  GENERALMANAGER      30 non-null     object
 12  HEADCOACH           30 non-null     object
 13  DLEAGUEAFFILIATION  30 non-null     object
dtypes: float64(1), int64(5), object(8)
memory usage: 3.4+ KB
```

Appendix B: Final Queries

Query 1 [Complex]

```
with cte1 AS (
   Select STR_TO_DATE(GAME_DATE_EST,'%m/%d/%Y') as date,
          Game.SEASON
   FROM Game
),
   cte2 as(
       Select TEAM_ID,
              STR_TO_DATE(Ranking.STANDINGSDATE,'%Y-%m-%d') as date,
              Ranking.CONFERENCE, W,L,
              Ranking.HOME_RECORD,Ranking.ROAD_RECORD
              From Ranking
   ),
   west as(
       Select TEAM_ID, cte1.SEASON, cte2.date, CONFERENCE,W,L,
              HOME_RECORD,ROAD_RECORD
       From cte2 join cte1 on cte2.date=cte1.date
       where SEASON='${req.params.season}' and CONFERENCE='West'
       order by W desc limit 1
   ),
   east as(
       Select TEAM_ID, cte1.SEASON, cte2.date, CONFERENCE,W,L,
              HOME_RECORD,ROAD_RECORD
       From cte2 join cte1 on cte2.date=cte1.date
       where SEASON ='${req.params.season}' and CONFERENCE='East'
       order by W desc limit 1
   ),
   westaway as (
       SELECT Game_Stats_Team.TEAM_ID_away,
              Round(avg(PTS_away),2) as AVG_Away_Score
       From Game_Stats_Team
       Join Game
       on Game_Stats_Team.GAME_ID = Game.GAME_ID
       WHERE TEAM_ID_away IN (SELECT west.TEAM_ID from west)
             and SEASON='${req.params.season}'
       GROUP BY TEAM_ID_away
   ),
   westhome as (
       SELECT Game_Stats_Team.TEAM_ID_home,
              Round(avg(PTS_home),2) as AVG_Home_Score
       From Game_Stats_Team
       Join Game
       on Game_Stats_Team.GAME_ID = Game.GAME_ID
       WHERE TEAM_ID_home IN (SELECT west.TEAM_ID from west)
             and SEASON='${req.params.season}'
       GROUP BY TEAM_ID_home
   ),
   eastaway as (
       SELECT Game_Stats_Team.TEAM_ID_away,
              Round(avg(PTS_away),2) as AVG_Away_Score
       From Game_Stats_Team
       Join Game
```

```
        on Game_Stats_Team.GAME_ID = Game.GAME_ID
        WHERE TEAM_ID_away IN (SELECT east.TEAM_ID from east)
              and SEASON='${req.params.season}'
        GROUP BY TEAM_ID_away
    ),
    easthome as (
        SELECT Game_Stats_Team.TEAM_ID_home,
               Round(avg(PTS_home),2) as AVG_Home_Score
        From Game_Stats_Team
        Join Game
        on Game_Stats_Team.GAME_ID = Game.GAME_ID
        WHERE TEAM_ID_home IN (SELECT east.TEAM_ID from east)
              and SEASON='${req.params.season}'
        GROUP BY TEAM_ID_home
    )

SELECT CONCAT(Team.city, ' ', Team.nickname) as abbreviation, Season,
CONFERENCE,W,L,HOME_RECORD,ROAD_RECORD,
      AVG_Away_Score,AVG_Home_Score
From west join westaway on west.TEAM_ID = westaway.TEAM_ID_away
         join westhome on west.TEAM_ID = westhome.TEAM_ID_home
         join Team on west.TEAM_ID = Team.TEAM_ID
union
SELECT CONCAT(Team.city, ' ', Team.nickname) as abbreviation, Season,
CONFERENCE,W,L,HOME_RECORD,ROAD_RECORD,
      AVG_Away_Score,AVG_Home_Score
From east join eastaway on east.TEAM_ID = eastaway.TEAM_ID_away
         join easthome on east.TEAM_ID = easthome.TEAM_ID_home
         join Team on east.TEAM_ID = Team.TEAM_ID
```

Query 2 [Simple]

```
With cte1 AS(
   SELECT *, CASE
             WHEN Game_Stats_Team.PTS_home>Game_Stats_Team.PTS_away THEN 'HomeWin'
             ELSE 'AwayWin' END AS Result
  FROM Game_Stats_Team
  ),
     cte2 as(
         select Game_Stats_Team.GAME_ID, TEAM_ID_home, TEAM_ID_away,
         CONCAT(Team.city, ' ', Team.nickname) as Home
         FROM Game_Stats_Team JOIN Team on Game_Stats_Team.TEAM_ID_home = Team.TEAM_ID
     ),
     teamnames as(
         select cte2.GAME_ID,cte2.Home, CONCAT(Team.city, ' ', Team.nickname) as Away
         FROM cte2 JOIN Team on cte2.TEAM_ID_away=Team.TEAM_ID
         order by GAME_ID
     ),
     gamesummary as(
         Select teamnames.Home, teamnames.Away,cte1.GAME_ID,
                cte1.Result, cte1.PTS_home,cte1.PTS_away,
                cte1.FG3_PCT_home,cte1.FG3_PCT_away,
                cte1.AST_home,cte1.AST_away,
                cte1.FG_PCT_home,cte1.FG_PCT_away,
```

```
                cte1.FT_PCT_home,cte1.FT_PCT_away,
                cte1.REB_home,cte1.REB_away
                FROM cte1
                left join teamnames
                on cte1.Game_ID = teamnames.GAME_ID
     )

SELECT LEFT(str_to_date(GAME_DATE_EST,'%m/%d/%Y'),10) as date,
       SEASON, Home, Away,Result,PTS_home,PTS_away,
       FG_PCT_home,FG3_PCT_away,AST_home,AST_away,FG_PCT_home,FG_PCT_away,
       FT_PCT_home,FT_PCT_away,REB_home,REB_away
FROM gamesummary JOIN Game on gamesummary.GAME_ID = Game.GAME_ID
WHERE str_to_date(GAME_DATE_EST,'%m/%d/%Y')='${req.params.date}'
```

Query 3 [Simple]

```
WITH TeamName AS (
       SELECT TEAM_ID, CONCAT(Team.city, ' ', Team.nickname) AS name
       FROM Team
     ),

     Conference AS (
         SELECT DISTINCT Ranking.team_id AS TEAM_ID, Ranking.conference AS conference
         FROM Ranking
         WHERE NOT (Ranking.team_id = 1610612740 AND Ranking.conference = 'East')
     )

     SELECT
         TN.name AS Name,
         T.abbreviation AS Abbreviation,
         T.year_founded AS FoundedYear,
         C.conference AS Conference,
         T.arena AS Arena,
         T.thumbnail_url AS TeamLogo
     FROM Team T
     JOIN TeamName TN ON T.TEAM_ID = TN.TEAM_ID
     JOIN Conference C ON T.TEAM_ID = C.TEAM_ID
     ORDER BY Conference, Name;
```

Query 4 [Complex]

```
CREATE TABLE Wrapper AS //Caching
   SELECT
       GSP.game_id AS GAME_ID,
       PU.name AS Player_Name,
       T.nickname AS Player_Team,
       GUM.Game_season,
       STR_TO_DATE(GUM.Game_Date,'%m/%d/%Y') AS Game_Date,
       GUM.MatchUp,
       GSP.home_or_away AS Home_Away,
       GUM.Result AS WinningTeam,
       CASE
           WHEN GSP.home_or_away = 'home' AND GUM.Result = 'HomeWin' THEN 'Win'
           WHEN GSP.home_or_away = 'away' AND GUM.Result = 'AwayWin' THEN 'Win'
           WHEN GSP.home_or_away = 'home' AND GUM.Result = 'AwayWin' THEN 'Loss'
```

```
            WHEN GSP.home_or_away = 'away' AND GUM.Result = 'HomeWin' THEN 'Loss'
        END AS Result,
        CASE
            WHEN INSTR(MIN, ':') > 0 THEN MIN
            ELSE CONCAT(MIN, ':00')
        END AS MIN,
        (3*FG3M + 2*(FGM - FG3M) + 1*FTM) AS PTS,
        FGM,
        FGA,
        CASE WHEN FGA = 0 THEN 0.000 ELSE (FGM / FGA) END AS FG_PCT,
        FG3M,
        FG3A,
        CASE WHEN FG3A = 0 THEN 0.000 ELSE (FG3M / FG3A) END AS FG3_PCT,
        FTM,
        FTA,
        CASE WHEN FTA = 0 THEN 0.000 ELSE (FTM / FTA) END AS FT_PCT,
        OREB,
        DREB,
        (OREB + DREB) AS REB,
        AST,
        STL,
        BLK,
        TOO AS TOV,
        PF
    FROM Game_Stats_Player GSP
    JOIN (SELECT
          GU.GAME_ID,
          CONCAT(T1.nickname, ' vs. ', T2.nickname) AS MatchUp,
          GU.Game_season,
          GU.Game_Date,
          GU.Result
          FROM (SELECT
                GST.GAME_ID AS GAME_ID,
                GST.TEAM_ID_home AS Home_Team,
                GST.TEAM_ID_away AS Away_Team,
                G.SEASON AS Game_season,
                G.GAME_DATE_EST AS Game_Date,
                CASE WHEN GST.PTS_home > GST.PTS_away THEN 'HomeWin'
                      WHEN GST.PTS_home < GST.PTS_away THEN 'AwayWin'
                      WHEN GST.PTS_home = GST.PTS_away THEN 'Tie'
                      ELSE 'missing' END AS Result
                FROM Game_Stats_Team GST
                      JOIN Game G ON GST.GAME_ID = G.GAME_ID) GU
                      JOIN Team T1 ON GU.Home_Team = T1.TEAM_ID
                      JOIN Team T2 ON GU.Away_Team = T2.TEAM_ID
                WHERE GU.Result <> 'missing') GUM ON GUM.GAME_ID = GSP.game_id
        JOIN Player PU ON GSP.player_id = PU.PLAYER_ID
        JOIN Team T ON GSP.team_id = T.TEAM_ID;

SELECT *
    FROM Wrapper
    WHERE Game_season = ${game_season} AND
          Result = '${result}' AND PTS >= ${pts} AND
          FG_PCT >= ${fg_pct} AND FG3_PCT >= ${fg3_pct} AND
          FT_PCT >= ${ft_pct} AND REB >= ${reb} AND
```

```
        AST >= ${ast} AND STL >= ${stl} AND
        BLK >= ${blk} AND TOV >= ${tov} AND
        PF >= ${pf}
  AND Player_Name LIKE '%${player_name}%'
  AND Game_Date BETWEEN '${game_date_start}' AND '${game_date_end}
  ORDER BY Game_Date ASC
```

Query 5 [Complex]

```
CREATE TABLE Wrapper_Team AS //Caching
   SELECT
       GST.GAME_ID AS GAME_ID,
       GM.Game_Season,
       GM.Game_Date,
       GM.MatchUp,
       GM.Home_Team,
       GM.Away_Team,
       GM.Result,
       GST.PTS_home AS HomePTS,
       GST.PTS_away AS AwayPTS,
       GST.FG_PCT_home AS HomeFG_PCT,
       GST.FG_PCT_away AS AwayFG_PCT,
       GST.FT_PCT_home AS HomeFT_PCT,
       GST.FT_PCT_away AS AwayFT_PCT,
       GST.FG3_PCT_home AS HomeFG3_PCT,
       GST.FG3_PCT_away AS AwayFG3_PCT,
       GST.AST_home AS Home_AST,
       GST.AST_away AS Away_AST,
       GST.REB_home AS Home_REB,
       GST.REB_away AS Away_REB
   FROM Game_Stats_Team GST
   JOIN (SELECT
       GST.GAME_ID,
       SEASON AS Game_Season,
       STR_TO_DATE(GAME_DATE_EST,'%m/%d/%Y') AS Game_Date,
       CONCAT(T1.nickname, ' vs. ', T2.nickname) AS MatchUp,
       CONCAT(T1.city, ' ', T1.nickname) AS Home_Team,
       CONCAT(T2.city, ' ', T2.nickname) AS Away_Team,
       CASE
           WHEN GST.PTS_home > GST.PTS_away THEN 'HomeWin'
           WHEN GST.PTS_home < GST.PTS_away THEN 'AwayWin'
           WHEN GST.PTS_home = GST.PTS_away THEN 'Tie'
           ELSE 'missing'
       END AS Result
   FROM Game_Stats_Team GST
   JOIN Team T1 ON GST.TEAM_ID_home = T1.TEAM_ID
   JOIN Team T2 ON GST.TEAM_ID_away = T2.TEAM_ID
   JOIN Game G ON GST.GAME_ID = G.GAME_ID) AS GM ON GST.GAME_ID = GM.GAME_ID
   WHERE Result <> 'missing';

(SELECT
       GAME_ID,
```

```
       Home_Team AS Team,
       Game_Season, Game_Date, MatchUp, Home_Team, Away_Team, Result, HomePTS,
AwayPTS, HomeFG_PCT, AwayFG_PCT, HomeFT_PCT, AwayFT_PCT, HomeFG3_PCT, AwayFG3_PCT,
Home_AST, Away_AST, Home_REB, Away_REB
FROM Wrapper_Team
WHERE Home_Team = '${team}' AND HomePTS >= ${pts} AND
       HomeFG_PCT >= ${fg_pct} AND HomeFT_PCT >= ${ft_pct}
       AND HomeFG3_PCT >= ${fg3_pct}
       AND Home_AST >= ${ast} AND Home_REB >= ${reb} AND Result = 'HomeWin'
       AND Game_Season = ${game_season}
)
UNION
(SELECT
       GAME_ID,
       Away_Team AS Team,
       Game_Season, Game_Date, MatchUp, Home_Team, Away_Team, Result, HomePTS,
AwayPTS, HomeFG_PCT, AwayFG_PCT, HomeFT_PCT, AwayFT_PCT, HomeFG3_PCT, AwayFG3_PCT,
Home_AST, Away_AST, Home_REB, Away_REB
FROM Wrapper_Team
WHERE Away_Team = '${team}' AND AwayPTS >= ${pts} AND
       AwayFG_PCT >= ${fg_pct} AND AwayFT_PCT >= ${ft_pct}
       AND AwayFG3_PCT >= ${fg3_pct}
       ND Away_AST >= ${ast} AND Away_REB >= ${reb} AND Result = 'AwayWin'
       AND Game_Season = ${game_season})
)
```

Query 6 [Simple]

```
SELECT DISTINCT CONCAT(city, ' ', nickname) AS name
   FROM Team
   ORDER BY name;
```

Query 7 [Simple]

```
WITH season_games AS (
     SELECT * FROM Game g
     WHERE g.SEASON = ${season}
     AND STR_TO_DATE(GAME_DATE_EST, '%m/%d/%Y') <=
             (SELECT STR_TO_DATE(temp1.STANDINGSDATE, '%Y-%m-%d') AS Date1
             FROM (SELECT STANDINGSDATE
                   FROM Ranking
                   WHERE is_post_season = 1
                     AND YEAR(STR_TO_DATE(STANDINGSDATE, '%Y-%m-%d')) = ${season} + 1)
temp1
             ORDER BY Date1
             LIMIT 1)
     AND STR_TO_DATE(GAME_DATE_EST, '%m/%d/%Y') >=
             (SELECT STR_TO_DATE(temp2.STANDINGSDATE, '%Y-%m-%d') AS Date2
             FROM (SELECT STANDINGSDATE
                   FROM Ranking
```

```
                    WHERE is_post_season = 0
                      AND YEAR(STR_TO_DATE(STANDINGSDATE, '%Y-%m-%d')) = ${season})
temp2
             ORDER BY Date2
             LIMIT 1)
     ORDER BY STR_TO_DATE(GAME_DATE_EST, '%Y-%m-%d')
)
SELECT season_games.SEASON as SEASON,
       t1.nickname AS Home_team,
       t2.nickname AS Away_team,
       season_games.GAME_DATE_EST AS Date,
       PTS_home,
       FG_PCT_home,
       FT_PCT_home,
       FG3_PCT_home,
       AST_home,
       REB_home,
       PTS_away,
       FG_PCT_away,
       FT_PCT_away,
       FG3_PCT_away,
       AST_away,
       REB_away
FROM Game_Stats_Team gt
     JOIN Team t1 ON gt.TEAM_ID_home = t1.TEAM_ID
     JOIN Team t2 ON gt.TEAM_ID_away = t2.TEAM_ID
     JOIN season_games ON gt.GAME_ID = season_games.GAME_ID
WHERE t1.nickname LIKE '${team}' OR t2.nickname LIKE '${team}'
ORDER BY STR_TO_DATE(Date, '%Y-%m-%d')
```

Query 8 [Simple]

```
SELECT p.name, g.SEASON, t.nickname AS Team_Played_for, (SUM(FGM*2) + SUM(FTM) +
SUM(FG3M*3))/COUNT(*) AS Average_Points,
   SUM(FGM)/SUM(FGA) AS two_point_shot_Percentage,
   SUM(FG3M)/SUM(FG3A) AS three_point_shot_Percentage,
   SUM(FTM)/SUM(FTA) AS free_throw_Percentage,
   SUM(AST)/COUNT(*) AS Average_Assists,
   SUM(OREB + DREB)/COUNT(*) AS Average_Rebounds,
   SUM(STL)/COUNT(*) AS Average_Steals,
   SUM(BLK)/COUNT(*) AS Average_Blocks
FROM Game g
```

```
JOIN Game_Stats_Player gp ON g.GAME_ID = gp.game_id
 JOIN Player p ON gp.player_id = p.player_id
 JOIN Roster r ON p.player_id = r.player_id
 JOIN Team t ON r.team_id = t.team_id
 WHERE p.name LIKE '${req.params.player_name}'
 AND g.SEASON = ${season}
 AND r.SEASON = ${season}
 AND STR_TO_DATE(GAME_DATE_EST, '%m/%d/%Y') <=
       (SELECT STR_TO_DATE(temp1.STANDINGSDATE, '%Y-%m-%d') AS Date1
         FROM (SELECT STANDINGSDATE
               FROM Ranking
               WHERE is_post_season = 1
                 AND YEAR(STR_TO_DATE(STANDINGSDATE, '%Y-%m-%d')) = ${season} + 1)
temp1
         ORDER BY Date1
         LIMIT 1)
 AND STR_TO_DATE(GAME_DATE_EST, '%m/%d/%Y') >=
       (SELECT STR_TO_DATE(temp2.STANDINGSDATE, '%Y-%m-%d') AS Date2
         FROM (SELECT STANDINGSDATE
               FROM Ranking
               WHERE is_post_season = 0
                 AND YEAR(STR_TO_DATE(STANDINGSDATE, '%Y-%m-%d')) = ${season}) temp2
         ORDER BY Date2
         LIMIT 1)
 GROUP BY name, SEASON;
```

Query 9 [Simple]

```
 WITH Player1 AS (
     SELECT g.SEASON as season,
             p.name,
             AVG(gp.FGM * 2 + gp.FG3M * 3 + gp.FTM) as average_points
     FROM Game_Stats_Player gp
     JOIN Game g ON gp.GAME_ID = g.GAME_ID
     JOIN Player p ON gp.player_id = p.PLAYER_ID
     WHERE p.name LIKE '${req.params.player1_name}'
         AND g.SEASON BETWEEN ${season_start} AND ${season_end}
     GROUP BY g.SEASON
 ),
 Player2 AS (
     SELECT g.SEASON as season,
             p.name,
             AVG(gp.FGM * 2 + gp.FG3M * 3 + gp.FTM) as average_points
```

```
    FROM Game_Stats_Player gp
    JOIN Game g ON gp.GAME_ID = g.GAME_ID
    JOIN Player p ON gp.player_id = p.PLAYER_ID
    WHERE p.name LIKE '${req.params.player2_name}'
        AND g.SEASON BETWEEN ${season_start} AND ${season_end}
    GROUP BY g.SEASON
)
SELECT p1.season,
      p1.name AS Player1_Name,
      p2.name AS Player2_Name,
      p1.average_points AS Player1_average_points,
      p2.average_points AS Player2_average_points
FROM Player1 p1
JOIN Player2 p2
    ON p1.season = p2.season
ORDER By p1.season
```

Query 10 [Simple]

```
SELECT count(distinct team_id) as teams_count
   FROM Roster r JOIN
       (SELECT player_id FROM Player WHERE name = '${player_name}') p
       ON r.player_id = p.PLAYER_ID
```

Query 11 [Simple]

```
SELECT count(*) as games_count
   FROM Game_Stats_Player g_s JOIN
       (SELECT player_id FROM Player WHERE name = '${player_name}') p
       ON g_s.player_id = p.PLAYER_ID
   WHERE FGA IS NOT NULL
```

Query 12 [Simple]

```
SELECT name
FROM Player p JOIN Game_Stats_Player g_s
      ON p.player_id = g_s.player_id
WHERE FGM IS NOT NULL
GROUP BY name
HAVING count(*) > 1
```

Query 13 [Complex]

```
CREATE INDEX player_name ON Player(name);
CREATE INDEX g_s_player_id ON Game_Stats_Player(player_id);

WITH Game_Stats_Player_with_Total AS (
     SELECT *,( 3 * FG3M + 2 * (FGM - FG3M) + 1 * FTM) AS Total
     FROM Game_Stats_Player
   ), latest_5_games AS (
       SELECT g.GAME_ID
```

```
        FROM Game_Stats_Player g_s JOIN Player p
                ON g_s.player_id = p.PLAYER_ID
            JOIN Game g
                ON g_s.game_id = g.GAME_ID
        WHERE
            p.name = '${player_name}'
            AND g_s.FTA IS NOT NULL
        ORDER BY str_to_date(g.GAME_DATE_EST, '%m/%d/%y') DESC
        LIMIT 5
    ), latest_5_game_all_player AS (
        SELECT
            g.GAME_ID
            , g.GAME_DATE_EST
            , p.PLAYER_ID
            , p.name
            , ${metric_name} as metric

        FROM Game_Stats_Player_with_Total g_s JOIN Player p
                ON g_s.player_id = p.PLAYER_ID
            JOIN Game g
                ON g_s.game_id = g.GAME_ID
        WHERE g_s.FTA IS NOT NULL
            AND g_s.game_id IN (SELECT GAME_ID FROM latest_5_games)
    ), metric_rank AS (
        SELECT
            game_id,
            GAME_DATE_EST,
            player_id,
            name,
            metric,
            rank() over(partition by game_id order by metric DESC) as rnk
        FROM latest_5_game_all_player
        ORDER BY game_id, rnk
        )

    SELECT game_date_est, name, metric, rnk
    FROM metric_rank
    WHERE name = '${player_name}';
```

Query 14 [Complex]

```
CREATE INDEX player_name ON Player(name);
CREATE INDEX g_s_player_id ON Game_Stats_Player(player_id);

WITH Game_Stats_Player_with_Total AS (
   SELECT *,( 3 * FG3M + 2 * (FGM - FG3M) + 1 * FTM) AS Total
   FROM Game_Stats_Player
   )

   SELECT *
     FROM (
       SELECT
         GAME_DATE_EST
         , metric
         , round((CASE WHEN prev_game_metric = 0 AND metric > 0 THEN 1
```

```
                WHEN prev_game_metric = 0 AND metric = 0 THEN 0
            ELSE ((metric - prev_game_metric) / prev_game_metric) END), 3) as
game_over_game_pct
       FROM
         (SELECT
             str_to_date(g.GAME_DATE_EST, '%m/%d/%y') as GAME_DATE_EST
               , metric
               , lag(metric, 1) over(order by str_to_date(g.GAME_DATE_EST,
'%m/%d/%y')) as prev_game_metric
         FROM
             (
                 SELECT
                     g_s.game_id
                     , ${metric_name} as metric
                 FROM
                     (SELECT * FROM
                            Game_Stats_Player_with_Total
                            WHERE FTA IS NOT NULL) g_s JOIN
                     (SELECT * FROM Player WHERE name = '${player_name}') p
                     ON g_s.player_id = p.PLAYER_ID
             ) a
             JOIN Game g
                 ON a.game_id = g.GAME_ID
         ) b
       ORDER BY GAME_DATE_EST DESC
       LIMIT 5
     ) c
 ORDER BY GAME_DATE_EST;
```

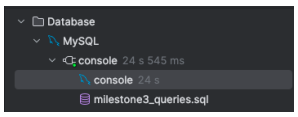
Appendix C: Optimization timing & screenshots

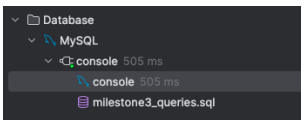
Query No.	*Pre-Op Time*	*Post-Op Time*	*Pre-Op Query*
1	console console_1 828 ms console_1 828 ms	console console_1 1 s 396 ms console_1 1 s 396 ms	

```
with cte1 AS (
    Select STR_TO_DATE(GAME_DATE_EST,'%m/%d/%Y') as date,Game.SEASON FROM Game
),
    cte2 as(
        Select TEAM_ID,STR_TO_DATE(Ranking.STANDINGSDATE,'%Y-%m-%d') as date,Ranking.CONFERENCE, W,L,Ranking.HOME_RECORD,Ranking.ROAD_RECORD
                From Ranking
    ),
    west as(Select TEAM_ID, cte1.SEASON, cte2.date, CONFERENCE,W,L,HOME_RECORD,ROAD_RECORD From cte2 join cte1 on cte2.date=cte1.date_
         CONFERENCE='West'
    ),
    east as(
        Select TEAM_ID, cte1.SEASON, cte2.date, CONFERENCE,W,L, HOME_RECORD,ROAD_RECORD From cte2 join cte1 on cte2.date=cte1.date
        CONFERENCE='East'
    ),
    westaway as (
        SELECT Game_Stats_Team.TEAM_ID_away,Round(avg(PTS_away),2) as AVG_Away_ScoreFrom Game_Stats_Team Join Game
        on Game_Stats_Team.GAME_ID = Game.GAME_ID WHERE TEAM_ID_away = (SELECT west.TEAM_ID from west) GROUP BY TEAM_ID_away
    ),
    westhome as (
        SELECT Game_Stats_Team.TEAM_ID_home,Round(avg(PTS_home),2) as AVG_Home_Score From Game_Stats_Team Join Game
        on Game_Stats_Team.GAME_ID = Game.GAME_IDWHERE TEAM_ID_home = (SELECT west.TEAM_ID from west)GROUP BY TEAM_ID_home
    ),
    eastaway as (
        SELECT Game_Stats_Team.TEAM_ID_away,
               Round(avg(PTS_away),2) as AVG_Away_Score
        From Game_Stats_Team
        Join Game
        on Game_Stats_Team.GAME_ID = Game.GAME_ID
        WHERE TEAM_ID_away = (SELECT east.TEAM_ID from east)
        GROUP BY TEAM_ID_away
    ),
```

```
    ),
    easthome as (
        SELECT Game_Stats_Team.TEAM_ID_home,
               Round(avg(PTS_home),2) as AVG_Home_Score
        From Game_Stats_Team
        Join Game
        on Game_Stats_Team.GAME_ID = Game.GAME_ID
        WHERE TEAM_ID_home = (SELECT east.TEAM_ID from east)
        GROUP BY TEAM_ID_home
    )

SELECT CONCAT(Team.city, ' ', Team.nickname) as abbreviation, Season, CONFERENCE,W,L,HOME_RECORD,ROAD_RECORD,
       AVG_Away_Score,AVG_Home_Score
From west join westaway on west.TEAM_ID = westaway.TEAM_ID_away
          join westhome on west.TEAM_ID = westhome.TEAM_ID_home
          join Team on west.TEAM_ID = Team.TEAM_ID
          WHERE SEASON=2021
           order by W desc limit 1
union
SELECT CONCAT(Team.city, ' ', Team.nickname) as abbreviation, Season, CONFERENCE,W,L,HOME_RECORD,ROAD_RECORD,
       AVG_Away_Score,AVG_Home_Score
From east join eastaway on east.TEAM_ID = eastaway.TEAM_ID_away
          join easthome on east.TEAM_ID = easthome.TEAM_ID_home
          join Team on east.TEAM_ID = Team.TEAM_ID
          WHERE SEASON=2021
           order by W desc limit 1
```

4



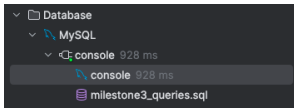


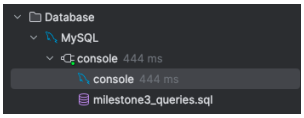

```
WITH Game_Update AS (
    SELECT
        GST.GAME_ID AS GAME_ID,
        GST.TEAM_ID_home AS Home_Team,
        GST.TEAM_ID_away AS Away_Team,
        G.SEASON AS Game_season,
        G.GAME_DATE_EST AS Game_Date,
        CASE
            WHEN GST.PTS_home > GST.PTS_away THEN 'HomeWin'
            WHEN GST.PTS_home < GST.PTS_away THEN 'AwayWin'
            WHEN GST.PTS_home = GST.PTS_away THEN 'Tie'
            ELSE 'missing'
        END AS Result
    FROM Game_Stats_Team GST
    JOIN Game G ON GST.GAME_ID = G.GAME_ID
),

Game_Update_Matches AS (
    SELECT
        GU.GAME_ID,
        CONCAT(T1.nickname, ' vs. ', T2.nickname) AS MatchUp,
        GU.Game_season,
        GU.Game_Date,
        GU.Result
    FROM Game_Update GU
    JOIN Team T1 ON GU.Home_Team = T1.TEAM_ID
    JOIN Team T2 ON GU.Away_Team = T2.TEAM_ID
    WHERE GU.Result <> 'missing'
),
```

```
Wrapper AS (
    SELECT
        PU.name AS Player_Name,
        T.nickname AS Player_Team,
        GUM.Game_season,
        STR_TO_DATE(GUM.Game_Date,'%m/%d/%Y') AS Game_Date,
        GUM.MatchUp,
        GSP.home_or_away AS Home_Away,
        GUM.Result AS WinningTeam,
        CASE
            WHEN GSP.home_or_away = 'home' AND GUM.Result = 'HomeWin' THEN 'Win'
            WHEN GSP.home_or_away = 'away' AND GUM.Result = 'AwayWin' THEN 'Win'
            WHEN GSP.home_or_away = 'home' AND GUM.Result = 'AwayWin' THEN 'Loss'
            WHEN GSP.home_or_away = 'away' AND GUM.Result = 'HomeWin' THEN 'Loss'
        END AS Result,
        CASE
            WHEN INSTR(MIN, ':') > 0 THEN MIN
            ELSE CONCAT(MIN, ':00')
        END AS MIN,
        (3*FG3M + 2*(FGM - FG3M) + 1*FTM) AS PTS,
        FGM,
        FGA,
        CASE WHEN FGA = 0 THEN 0.000 ELSE (FGM / FGA) END AS FG_PCT,
        FG3M,
        FG3A,
        CASE WHEN FG3A = 0 THEN 0.000 ELSE (FG3M / FG3A) END AS FG3_PCT,
        FTM,
        FTA,
        CASE WHEN FTA = 0 THEN 0.000 ELSE (FTM / FTA) END AS FT_PCT,
        OREB,
```

```
        DREB,
        (OREB + DREB) AS REB,
        AST,
        STL,
        BLK,
        TOO AS TOV,
        PF
    FROM Game_Stats_Player GSP
    JOIN Game_Update_Matches GUM ON GUM.GAME_ID = GSP.game_id
    JOIN Player PU ON GSP.player_id = PU.PLAYER_ID
    JOIN Team T ON GSP.team_id = T.TEAM_ID
)

SELECT *
FROM
    (SELECT *
    FROM Wrapper
    ORDER BY Game_Date DESC) a
WHERE Player_Name = 'LeBron James' AND
      Game_season = 2013 AND
      Result = 'Win' AND
      PTS >= 0 AND
      FG_PCT >= 0 AND FG3_PCT >= 0 AND FT_PCT >= 0 AND
      REB >= 0 AND AST >= 0 AND STL >=0 AND BLK >=0 AND
      TOV >=0 AND PF >= 0 AND
      Game_Date BETWEEN '2000-10-07' AND '2030-11-07';
```

5






```sql
WITH GameMatch AS (
    SELECT
        GST.GAME_ID,
        SEASON AS Game_Season,
        STR_TO_DATE(GAME_DATE_EST,'%m/%d/%Y') AS Game_Date,
        CONCAT(T1.nickname, ' vs. ', T2.nickname) AS MatchUp,
        CONCAT(T1.city, ' ', T1.nickname) AS Home_Team,
        CONCAT(T2.city, ' ', T2.nickname) AS Away_Team,
        CASE
            WHEN GST.PTS_home > GST.PTS_away THEN 'HomeWin'
            WHEN GST.PTS_home < GST.PTS_away THEN 'AwayWin'
            WHEN GST.PTS_home = GST.PTS_away THEN 'Tie'
            ELSE 'missing'
        END AS Result
    FROM Game_Stats_Team GST
    JOIN Team T1 ON GST.TEAM_ID_home = T1.TEAM_ID
    JOIN Team T2 ON GST.TEAM_ID_away = T2.TEAM_ID
    JOIN Game G ON GST.GAME_ID = G.GAME_ID
),

Wrapper AS (
    SELECT
        GM.Game_Season,
        GM.Game_Date,
        GM.MatchUp,
        GM.Home_Team,
```

```sql
        GM.Away_Team,
        GM.Result,
        GST.PTS_home AS HomePTS,
        GST.PTS_away AS AwayPTS,
        GST.FG_PCT_home AS HomeFG_PCT,
        GST.FG_PCT_away AS AwayFG_PCT,
        GST.FT_PCT_home AS HomeFT_PCT,
        GST.FT_PCT_away AS AwayFT_PCT,
        GST.FG3_PCT_home AS HomeFG3_PCT,
        GST.FG3_PCT_away AS AwayFG3_PCT,
        GST.AST_home AS Home_AST,
        GST.AST_away AS Away_AST,
        GST.REB_home AS Home_REB,
        GST.REB_away AS Away_REB
    FROM Game_Stats_Team GST
    JOIN GameMatch AS GM ON GST.GAME_ID = GM.GAME_ID
    WHERE Result <> 'missing'
)
```

```sql
(SELECT
    Home_Team AS Team,
    Game_Season, Game_Date, MatchUp, Home_Team, Away_Team, Result, HomePTS, AwayPTS, HomeFG_PCT, AwayFG_PCT, HomeFT_PCT, AwayFT_PCT, HomeFG3_PCT, AwayFG3_PCT
FROM Wrapper
WHERE Home_Team = 'Philadelphia 76ers' AND HomePTS >= 90 AND
      HomeFG_PCT >=0 AND HomeFT_PCT >= 0 AND HomeFG3_PCT >=0
      AND Home_AST >=0 AND Home_REB >= 0 AND Result = 'HomeWin'
      AND Game_Season = 2013 AND Game_Date BETWEEN '2013-10-07' AND '2013-12-07'
ORDER BY Game_Date DESC)
UNION
(SELECT
    Away_Team AS Team,
    Game_Season, Game_Date, MatchUp, Home_Team, Away_Team, Result, HomePTS, AwayPTS, HomeFG_PCT, AwayFG_PCT, HomeFT_PCT, AwayFT_PCT, HomeFG3_PCT, AwayFG3_PCT,
FROM Wrapper
WHERE Away_Team = 'Philadelphia 76ers' AND AwayPTS >= 90 AND
      AwayFG_PCT >=0 AND AwayFT_PCT >= 0 AND AwayFG3_PCT >=0
      AND Away_AST >=0 AND Away_REB >= 0 AND Result = 'AwayWin'
      AND Game_Season = 2013 AND Game_Date BETWEEN '2013-10-07' AND '2013-12-07'
ORDER BY Game_Date DESC)
```

13	Database MySQL console 16 s 823 ms console 16 s milestone3_queries.sql	Database MySQL console 447 ms console 447 ms milestone3_queries.sql	WITH Game_Stats_Player_with_Total AS (SELECT *,(3 * FG3M + 2 * (FGM - FG3M) + 1 * FTM) AS Total FROM Game_Stats_Player), games_ranked AS (SELECT g.GAME_ID, p.name , rank() over(partition by p.player_id ORDER BY str_to_date(g.GAME_DATE_EST, '%m/%d/%y') DESC) as rnk FROM Game_Stats_Player g_s JOIN Player p ON g_s.player_id = p.PLAYER_ID JOIN Game g ON g_s.game_id = g.GAME_ID WHERE g_s.FTA IS NOT NULL), latest_5_games AS (SELECT * FROM games_ranked WHERE rnk <= 5) , games_all AS (SELECT g.GAME_ID, g.GAME_DATE_EST, p.PLAYER_ID, p.name, Total as metric FROM Game_Stats_Player_with_Total g_s JOIN Player p ON g_s.player_id = p.PLAYER_ID JOIN Game g ON g_s.game_id = g.GAME_ID WHERE g_s.FTA IS NOT NULL), metric_rank AS (SELECT game_id, GAME_DATE_EST, player_id, name, metric, rank() over(partition by game_id order by metric DESC) as rnk FROM games_all ORDER BY game_id, rnk) SELECT game_date_est, name, metric, rnk FROM metric_rank WHERE name = 'LeBron James' AND game_id IN (SELECT game_id FROM latest_5_games WHERE name = 'LeBron James');
14	Database MySQL console 1 m 20 s 436 ms console	Database MySQL console 930 ms console 930 ms milestone3_queries.sql	SELECT 1 m 5 s GAME_DATE_EST , metric , game_over_game_pct FROM (SELECT GAME_DATE_EST , player_id , name , metric , (CASE WHEN lag(metric, 1) over(partition by player_id order by GAME_DATE_EST) = 0 AND metric > 0 THEN 1 WHEN lag(metric, 1) over(partition by player_id order by GAME_DATE_EST) = 0 AND metric = 0 THEN 0 ELSE ((metric - lag(metric, 1) over(partition by player_id order by GAME_DATE_EST)) / lag(metric, 1) over(partition by player_id order by GAME_DATE_EST)) END) as game_over_game_pct FROM (SELECT str_to_date(g.GAME_DATE_EST, '%m/%d/%y') as GAME_DATE_EST , g_s.player_id, p.name, g_s.FGM as metric # can change metrics FROM Game_Stats_Player g_s JOIN Player p ON g_s.player_id = p.PLAYER_ID JOIN Game g ON g_s.game_id = g.GAME_ID WHERE g_s.min IS NOT NULL) a ORDER BY player_id, GAME_DATE_EST) b WHERE name = 'LUKA DONCIC' # can change name ORDER BY GAME_DATE_EST DESC LIMIT 5;

Appendix D: API Specification

Home Page Routes

Route 1

Request Path: /acknowledge/:type
Request Method: GET
Route Description: return the overall description of the project, which includes the project motivation, project description, and project authors. The information will display under the Acknowledgements section on the Home Page.

Request Path Parameter(s):

- **:type**: corresponding string description of the options below
    - motivation (String type): text describing the project motivation
    - description (String type): text describing the project description
    - authors (String type): text describing the project authors

Request Query Parameter(s): None

Required / Optional Indicators:

- ":type" is required

Route Handler: app.get('/acknowledge/:type', routes.acknowledge);

Return Type: JSON Object

Response Parameters:

- {motivation (String: a project motivation)}
- {description (String: a project description)}
- {authors (String: the project authors)}

HTTP Status: If failed, return status of 404

Route 2

Request Path: /get_team_names

Request Method: GET

Route Description: return all the processed teams name for selection (used in search section)

Request Path Parameter(s): None

Request Query Parameter(s): None

Required / Optional Indicators: None

Route Handler: app.get('/get_team_names', routes.get_team_names);

Return Type: JSON Object

Response Parameters:

- {name (String: team name)}

HTTP Status: If failed, return status of 404

Route 3

Request Path: /all_teams

Request Method: GET

Route Description: return all NBA teams and the corresponding key information. This will display under the Team section on the Home Page.

Request Path Parameter(s): None

Request Query Parameter(s): None

Required / Optional Indicators: None

Route Handler: app.get('/all_teams', routes.all_teams);

Return Type: JSON Object

Response Parameters:

- {Name (String: a NBA team name), Abbreviation (String: the corresponding abbreviated name of the NBA team), FoundedYear (String: the founding year of the NBA team), Conference (String,

the conference the NBA team belongs to: either West or East), Arena (String, the name of the home arena of the NBA team)}

HTTP Status: If failed, return status of 404

Route 4

Request Path: /search_players
Request Method: GET
Route Description: return matched NBA players' games and their corresponding key information. This will display under the PlayerSearch section on the Home Page.
Request Path Parameter(s): None
Request Query Parameter(s):

- player_name (String type): searched player's name (default null)
- game_season (Integer type): searched game season (default 2021)
- result (String type): searched game results: either win or loss (default 'Win')
- pts (Integer type): searched players that meet the specified game total points (default 0)
- fg_pct (Float type): searched players that meet the specified field goal percentage (default 0)
- fg3_pct (Float type): searched players that meet the specified three pointers made percentage (default 0)
- ft_pct (Float type): searched players that meet the specified free throws percentage (default 0)
- reb (Integer type): searched players that meet the specified rebounds (default 0)
- ast (Integer type): searched players that meet the specified assists (default 0)
- stl (Integer type): searched players that meet the specified steals (default 0)
- blk (Integer type): searched players that meet the specified blocks (default 0)
- tov (Integer type): searched players that meet the specified turnovers (default 0)
- pf (Integer type): searched players that meet the specified personal fouls (default 0)
- game_date (Date type): searched players that meet the specified date periods (default null)

Required / Optional Indicators:

- No required indicators
- player_name, game_season, result, pts, fg_pct, fg3_pct, ft_pct, reb, ast, stl, blk, tov, pf, game_date are optional indicators

Route Handler: app.get('/search_players', routes.search_players);
Return Type: JSON Object
Response Parameters:

- {Player_Name (String: matched player's name), Player_Team (String: matched player's team), Game_Season (Integer: matched player's season), Game_Date (Date: matched player's game date), MatchUp (String: matched player's played game), Home_Away (String: whether matched player's team is home team or away team), WinningTeam (String: matched players' played game's winning team), Result (String: matched players' played game's result: win or loss), MIN (String: matched player's played time), pts (Integer: matched players' game total points), fgm (Integer: matched players' game field goals made), fga (Integer: matched players' game field goals attempted), fg_pct (Float: matched players' game field goal percentage), fg3m (Integer:

matched players' game three pointers made), fg3a (Integer: matched players' game three pointers attempted), fg3_pct (Float: matched players' game three pointers percentage), ftm (Integer: matched players' game free throw made), fta (Integer: matched players' game free throw attempted), ft_pct (Integer: matched players' game free throw percentage), oreb (Integer: matched players' game offensive rebounds), dreb (Integer: matched players' game defensive rebounds), reb (Integer: matched players' game rebounds), ast (Integer: matched players' game assists), stl (Integer: matched players' game steals), blk (Integer: matched players' game blocks), tov (Integer: matched players' game turnovers), pf (Integer: matched players' game personal fouls)}

HTTP Status: If failed, return status of 404

Route 5

Request Path: /search_teams
Request Method: GET
Route Description: return matched NBA teams' games and their corresponding key information. This will display under the TeamSearch section on the Home Page.
Request Path Parameter(s): None
Request Query Parameter(s):

- team (String type): searched team's name (default 'Philadelphia 76ers')
- pts (Integer type): searched team's game that meet the specified total points (default 0)
- fg_pct (Float type): searched team's game that meet the specified field goal percentage (default 0)
- ft_pct (Float type): searched team's game that that meet the specified free throws percentage (default 0)
- fg3_pct (Float type): searched team's game that meet the specified three pointers percentage (default 0)
- ast (Integer type): searched team's game that meet the specified assists (default 0)
- reb (Integer type): searched team's game that meet the specified rebounds (default 0)
- result (String type): searched team's game that meet the specified result (win or loss) (default 'Win')
- game_season (Integer type): searched team's game on specified season (default 2021)
- game_date (Date type): searched team's game date on specified date (default '')

Required / Optional Indicators:

- No required indicators
- team, pts, fg_pct, ft_pct, fg3_pct, ast, reb, result, game_season, game_date are optional indicators

Route Handler: app.get('/search_teams', routes.search_teams);
Return Type: JSON Object
Response Parameters:

- {Team (String: matched team's name), Game_Season (Integer: matched team's game season), Game_Date (matched team's game date), MatchUp (String: matched team's game matchup), Home_Team (String: matched team's game home team), Away_Team (String: matched team's game away team), Result (String: matched team's game result), HomePTS (Integer: matched

team's game home team total points), AwayPTS (Integer: matched team's game away team total points), HomeFG_PCT (Float: matched team's game home team field goal percentage), AwayFG_PCT (Float: matched team's game away team field goal percentage), HomeFT_PCT (Float: matched team's game home team free throws percentage), AwayFT_PCT (Float: matched team's game away team free throws percentage), HomeFG3_PCT (Float: matched team's game home team's three pointers percentage), AwayFG3_PCT (Float: matched team's game away team three pointers percentage), Home_AST (Integer: matched team's game home team assists), Away_AST (Integer: matched team's game away team assists), Home_REB (Integer: matched team's game home team rebounds), Away_REB (Integer: matched team's game away team rebounds)}

HTTP Status: If failed, return status of 404

Game Page Routes

Route 1

Request Path: /search_games_in_date
Request Method: GET
Route Description: return matched NBA games given a specific date. This will display under the Games section on the Home Page.
Request Path Parameter(s): game_date(Date type): a specific date that the user wants to query
Request Query Parameter(s):game_id(Integer type): a specific game id that the user wants to query
Required / Optional Indicators: None
Route Handler: app.get('/games', routes.games);
Return Type: JSON Object
Response Parameters:

- {date (Date: the date that the user wants to query), Season (Integer: the corresponding season), Home(String: Home Abbreviation), Away(String: Away Abbreviation), Result (String: who won), PTS_HOME(Integer: home team points), PTS_AWAY(Integer: away team points), FG3_PCT_HOME (Float: FG3_PCT of home team), FG3_PCT_AWAY (Float: FG3_PCT of away team), AST_HOME (Float: AST of home team), AST_AWAY (Float: AST of away team), FG_PCT_HOME (Float: FG_PCT of home team), FG_PCT_AWAY (Float: FG_PCT of away team), FT_PCT_HOME (Float: FT_PCT of home team), FT_PCT_AWAY (Float: FT_PCT of away team), REB_HOME (Float: REB of home team), REB_AWAY (Float: REB of away team) }

HTTP Status: If failed, return status of 404

Route 2

Request Path: /search_games_in_date

Request Method: GET
Route Description: return matched NBA games given a specific date. This will display under the Games section on the Home Page.
Request Path Parameter(s): game_date(Date type): a specific date that the user wants to query
Request Query Parameter(s):game_id(Integer type): a specific game id that the user wants to query
Required / Optional Indicators: None
Route Handler: app.get('/games', routes.games);
Return Type: JSON Object
Response Parameters:

- {date (Date: the date that the user wants to query), Season (Integer: the corresponding season), Home(String: Home Abbreviation), Away(String: Away Abbreviation), Result (String: who won), PTS_HOME(Integer: home team points), PTS_AWAY(Integer: away team points), FG3_PCT_HOME (Float: FG3_PCT of home team), FG3_PCT_AWAY (Float: FG3_PCT of away team), AST_HOME (Float: AST of home team), AST_AWAY (Float: AST of away team), FG_PCT_HOME (Float: FG_PCT of home team), FG_PCT_AWAY (Float: FG_PCT of away team), FT_PCT_HOME (Float: FT_PCT of home team), FT_PCT_AWAY (Float: FT_PCT of away team), REB_HOME (Float: REB of home team), REB_AWAY (Float: REB of away team) }

HTTP Status: If failed, return status of 404

Championship Page Routes

Route 1

Request Path: /championship
Request Method: GET
Route Description: return the championship information given the season
Request Path Parameter(s): season(Integer type): a specific season that the user wants to query
Request Query Parameter(s):

- game_id(Integer type): a specific team id that the user wants to know

Required / Optional Indicators: None
Route Handler: app.get('/champ', routes.champ);
Return Type: JSON Object
Response Parameters:

- {Abbreviation (String: NBA team Abbreviation), Season (Integer: the corresponding season), CONFERENCE (String: West or East), W (Integer: number of wins), L (Integer: number of losses), HOME_RECOED (String: the cumulative record as a home team), AWAY_RECOED (String: the cumulative record as a away team), AVG_Away_Score (Float: the average score as an away team), AVG_Home_Score (Float: the average score as an home team) }

HTTP Status: If failed, return status of 404

Trend Page Routes

Trend Route 1

Request Path: /trend/games_count
Request Method: GET
Description: return the number of games a player has participated in in his career
Route Parameter(s): None
Query Parameter(s):
- player_name (string; a player's name); default 'Lebron James'

Required / Optional Indicators:
- No required indicators

Route Handler: app.get('/trend/games_count', routes.games_count)
Return Type: JSON Object
Return Parameters: { games_count (int; the number of games a player participated in) }
HTTP Status: If failed, return status of 404

Trend Route 2

Request Path: /trend/teams_count
Request Method: GET
Description: return the number of teams a player has played at in in his career
Route Parameter(s): None
Query Parameter(s):
- player_name (string; a player's name); default 'Lebron James'

Required / Optional Indicators:
- No required indicators

Route Handler: app.get('/trend/teams_count', routes.teams_count')
Return Type: JSON Object
Return Parameters: { teams_count (int; the number of teams a player participated in) }
HTTP Status: If failed, return status of 404

Trend Route 3

Request Path: /trend/compare
Request Method: GET
Description: return the player's metric's score ranking relative to other players who participated in the same game (represented by a newly created column called rank) for the latest 5 *games* where the player participated
Route Parameter(s): None

Query Parameter(s):

- player_name (string; a player's name); default 'Lebron James'
- metrics (string; the metric user is interested in, including [Total, FGA, FGM, etc.]); default 'Total'

Required / Optional Indicators:

- No required indicators

Route Handler: app.get('/trend/compare', routes.compare)
Return Type: JSON Object
Return Parameters: { GAME_DATE_EST (date; date the game is played), name (string; name of the player), metrics (int; the absolute points for the metric of interest), rnk (int; the player's ranking based on the metric compared to other players in the game) }
HTTP Status: If failed, return status of 404

Trend Route 4

Request Path: /trend/game
Request Method: GET
Description: The query will return the player's metric's score in the latest 5 *games* where the player participated and the metric's game over game percentage change.
Route Parameter(s): None
Query Parameter(s):

- player_name (string; a player's name); default 'Lebron James'
- metrics (string; the metric user is interested in, including [Total, FGA, FGM, etc.]); default 'Total'

Required / Optional Indicators:

- No required indicators

Route Handler: app.get('/trend/game', routes.trend_game)
Return Type: JSON Object
Return Parameters: { GAME_DATE_EST (date; date the game is played), metric (int; the absolute points for the metric of interest), game_over_game_pct (float; game over game percentage change in the metric of interest) }
HTTP Status: If failed, return status of 404

Trend Route 5

Request Path: /trend/game
Request Method: GET
Description: The query will return the player's metric's score in the latest 5 *games* where the player participated and the metric's season over season percentage change.
Route Parameter(s): None
Query Parameter(s):

- player_name (string; a player's name); default 'Lebron James'
- metrics (string; the metric user is interested in, including [Total, FGA, FGM, etc.]); default 'Total'

Required / Optional Indicators:

- No required indicators

Route Handler: app.get('/trend/game, routes.trend_game);

Return Type: JSON Object
Return Parameters: { GAME_DATE_EST (date; the NBA game date),(int; the absolute points for the metric of interest), game_over_game_pct (float; season over season percentage change in the metric of interest) }
HTTP Status: If failed, return status of 404

Player Page Routes

Route 1

Route: /players/:player_name
Request Method: GET
Description: The query will return the player's average points along with his shot percentages in a given season.
Route Parameter(s): player_name (string: name of the wanted player)
Query Parameter(s): season (int: wanted season)
Required / Optional Indicators:

- ":player_name" is required
- "season" is optional (default 2022)

Route Handler: app.get('/players/:player_name', routes.players);
Return Type: JSON Object
Return Parameters: {name (string: player name), SEASON (int: wanted season), Average_Points (float: average points), 2_point_shot_percentage (float: 2 point shot percentage), 3_point_shot_percentage (float: 3 point shot percentage), Free_Throw_Percentage (float: free throw percentage)}
HTTP Status: If failed, return status of 404

Route 2

Route: /compare/:player1_name/:player2_name
Request Method: GET
Description: The query will compare the performance of two players over a range of seasons.
Route Parameter(s): player1_name (string: name of the first player), player2_name (string: name of the other player)
Query Parameter(s): season_begin (int, default oldest season), season_end (int, default latest season)
Required / Optional Indicators: ":player1_name" and ":player2_name" is required, "season_begin" and "season_end" is optional
Route Handler: app.get('/compare/:player1_name/:player2_name', routes.compare);
Return Type: JSON Object

Return Parameters: {season (int: wanted season), {player1 name}_average_points (float: average points of the first player), {player2 name}_average_points (float: average points of the other player)}
HTTP Status: If failed, return status of 404

Team Page Routes

Route 1

Route: /team/:team_name
Request Method: GET
Description: The query will return the overall performance of each team in a given season
Route Parameter(s): team_name (str: name of wanted team)
Query Parameter(s): season (int: wanted season)
Required / Optional Indicators: ":team_name" is required; "season" is optional
Route Handler: app.get('/team/:team_name, routes.team);
Return Type: JSON Object
Return Parameters: {Team (str: name of the team), PTS (float: average number of points per game), FG_PCT (float: average field goal percentage), FT_PCT (float: average free throw percentage), FG3_PCT (float: average 3 point shot percentage), AST (float: average number of assists), REB_home (float: average number of rebounds)}
HTTP Status: If failed, return status of 404

Our Routes API Specification can be found at:

- https://github.com/sidzzzzz3/cis550/blob/main/server/routes.js