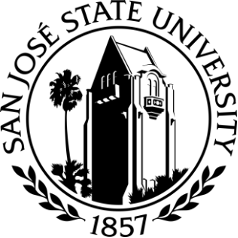
****

**San Jose State University**

***Department of Applied Science***

***One Washington Square, 95112***

****

**DATA 225: Sec-22 Database Systems for Analytics**

**Report - Team SNPS**

**An Application for Interactive Query and Exploration of an NBA Dataset**

|  |  |
| --- | --- |
| **Course Instructor:** | **Ronald Mak** |
| **Team Members:** | **Shresta** |
|  | **Naresh** |
|  | **Prudhvi Chirumamilla (016803410)** |
|  | **Shashank Reddy (016452501)** |

**INTRODUCTION**

The National Basketball Association (NBA) is a professional basketball league in North America composed of 30 teams (29 in the United States and 1 in Canada). It is one of the major professional sports leagues in the United States and Canada and is considered the premier professional basketball league in the world.

Key functionalities of the application include the following:

**Viewing**: The application allows users to view data within the database, generating reports on players, teams, games statistics & player statistics. This basic exploratory functionality tests the database's completeness and accuracy.

**Searching**: Search features utilize the database schema's attributes as filters, enabling targeted retrieval of records based on criteria such as player name, team, date range, and statistical thresholds.

**Filtering**: The application provides custom filtering of database tables to isolate subsets of records for focused analysis. This taps the full flexibility of the schema's multiple granularities.

**Updating**: Authorized application users can edit database fields to correct errors and append new information, testing the integrity constraints implemented within the schema design.

**Analyzing**: Visualizations and customized queries generated through the application interface leverage the full analytical power of the integrated, longitudinal NBA dataset. Emerging trends may then be identified.

**DATA SOURCE**

The dataset utilized in this project was sourced from the Kaggle website ([link here](https://www.kaggle.com/datasets/nathanlauga/nba-games)), where NBA stats website data was gathered, including game data, team stats, and player stats. The objective of this project is to create a user-friendly GUI application that allows individuals to access NBA data stored in MySQL.

The data source contains five files which are mentioned below:

* games\_details.csv: details of games dataset, all statistics of players for a given game.
* games.csv: all games from 2004 season to last update with the date, teams, and some details like number of points, etc.
* players.csv: players details (name).
* ranking.csv: ranking of NBA given a day (split into west and east on CONFERENCE column
* teams.csv: all teams of NBA.

**Operational Database Design**

The application is mainly divided into two parts, Operational and Analytical. Operational covers the first four functionalities mentioned above, namely Viewing, Searching, Filtering and Updating. Analytical part covers Analyzing the data and producing graphs.

The relationship between each entity can be described by ER diagram which is a relationship diagram that explains 1:1 (ONE - ONE), 1:M (ONE - MANY), M: M (MANY - MANY)

A picture containing screenshot, graphics, black, circle

Description automatically generatedFig. 1 ER Diagram of the Operational Database

The csv tables have been arranged in an effective manner into 6 entities like **standings**, **games**, **teams**, **team**\_**stats**, **players**, **player**\_**stats** and one junction table **roster** which is the result of teams and players entity having a M:M relation.

The entities and their attributes are mentioned below.

**players** - Stores information on individual players like name, height, weight, position, team, jersey number, and draft information. This allows you to query details about any current or former NBA player.

**games** - Contains information on each NBA game played like date, home team, away team, final score, venue, and season. This allows you to look up details about any game in NBA history.

**teams** - Stores information on each NBA team like name, city, arena, coach, general manager, and roster. This allows you to get details on any team, current or historic.

**team**\_**stats** - Tracks team statistics over time like points per game, field goal percentage, rebounds, assists, and more. This allows you to analyze team offensive and defensive trends.

**Player\_stats** - Contains statistics for each player in every game like points, rebounds, assists, steals, blocks, fouls, turnovers, and field goal percentage. This allows you to track player performances over their careers.

**standings** - Stores the standings, divisional rankings, wins, losses, and home/away records for each season. This allows you to see how teams fared and which teams made the playoffs for any given year.

**A screenshot of a computer

Description automatically generated with low confidence**

Fig 2. Relational Schema of the Operational Schema

**ETL to Operational Database**

There are few data inconsistency to be taken care of before loading the data into the database. One of which is missing data. The code below shows the missing players data for which there is games details but no player data.

df\_all = games\_details.merge(players.drop\_duplicates(), on=['PLAYER\_ID'], how='left', indicator=True)  
data = df\_all[df\_all['\_merge'] == 'left\_only']["PLAYER\_ID"]  
missingPlayers = pd.DataFrame(data = data)

To retrieve the missing data we have used the [nba\_api](https://github.com/swar/nba_api). We have particular used the CommonPlayerInfo method from nba\_api.stats.endpoints and created the getMissingPlayerInfo function which takes player\_id from the game\_details.csv and gets the players data.

from nba\_api.stats.endpoints import CommonPlayerInfo  
  
def getMissingPlayerInfo(player\_id):  
returnCommonPlayerInfo(player\_id).get\_data\_frames()[0][['FIRST\_NAME','LAST\_NAME', 'TEAM\_ID', 'HEIGHT', 'DRAFT\_YEAR']]

The missing data in games\_details.csv just means that the player has not played in that match. For the missing data in games.csv there is no data even in the official NBA site, as they are all concentrated in 2003 we will just drop 2003 data and start from 2004. The same will be done for ranking.csv file.

games = games[games.GAME\_DATE\_EST>'2004-01-01']  
ranking = ranking[ranking.STANDINGSDATE>'2004-01-01']

In players.csv we decided to add more features like height, weight, birthdate and draftyear we used CommonPlayerInfo again to get the details for all the players.

from nba\_api.stats.endpoints import commonplayerinfo  
  
def getPlayerInfo(player\_id):  
 return commonplayerinfo.CommonPlayerInfo(player\_id).get\_data\_frames()[0][['COUNTRY', 'HEIGHT','WEIGHT', 'DRAFT\_YEAR']]

Converting the height data which is an object data type to integer and converting foot data into cms.

def convertHeight(heightInFoot):  
 if len(str(heightInFoot).split("-")) == 2:  
 return(float(heightInFoot.split(" ")[0]+"."+player.HEIGHT.split("-")[1]))\*30.48  
 else:  
 return(float(heightInFoot)\*30.48

After dropping all the duplicates and dropping the columns where the missing values are very high, we start loading the data into the sql.

**Analytical Database Design**

A screenshot of a computer screen

Description automatically generated with medium confidenceIn the Analytical part we have focused on EDA of mainly two points, player stats and team stats. We built tools to see player performance over the seasons and team performance by calculating stats like win percentage, points per game, points allowed per game and so on. By focusing on EDA, organizations can gain a deeper understanding of the data, uncover valuable insights, and make informed decisions to optimize player and team performance in the NBA.

Fig 3. Star Schema of the Analytical Database

The user can analyze individual players, which players are most efficient, who improved in the game over time by seeing their scores in these visualization, and analyze by team , we can show the win percentage and scores per game defines the outcome of winning at home and away and with this user have the ability to slice and dice the stats by player and team and time reveals patterns, that how the coaches and general managers can use to maximize their odds of winning and we have taken this bunch of data turned it into an engine to improve strategy and performance.

**ETL to Analytical Database**

**Loading calendar Table**

sql = ( """  
 INSERT INTO snps\_wh.calendar(full\_date, day\_of\_week , mnth, qtr, yr)  
 SELECT DISTINCT(game\_date), DAYNAME(game\_date),   
 MONTH(game\_date), QUARTER(game\_date), YEAR(game\_date)  
 FROM games;  
 """  
 )  
  
cursor\_operational.execute(sql)  
conn\_operational.commit()

**Loading team Table**

sql = ( """  
 INSERT INTO snps\_wh.team(team\_key, team\_name, city , state, conference)  
 SELECT team\_id, concat(city, " ", name), city,   
 state, conference  
 FROM teams;  
 """  
 )  
  
cursor\_operational.execute(sql)  
conn\_operational.commit()

**Loading player Table**

sql = ( """  
 INSERT INTO snps\_wh.player(player\_key, player\_name, height, weight, draft\_year)  
 SELECT player\_id, player\_name, height, weight, draftyear  
 FROM players;  
 """  
 )  
  
cursor\_operational.execute(sql)  
conn\_operational.commit()

**Loading game Table**

sql = ( """  
 INSERT INTO snps\_wh.game(game\_key, date\_key, home\_team\_key , away\_team\_key, home\_score, away\_score, venue)  
 SELECT G.game\_id, C.date\_key, home\_team\_id, visitor\_team\_id, home.pts AS home\_pts, away.pts AS away\_pts, arena AS venue   
 FROM games G  
 INNER JOIN game\_stats home  
 ON home.game\_id = G.game\_id AND home.team\_id = home\_team\_id  
 INNER JOIN game\_stats away  
 ON away.game\_id = G.game\_id AND away.team\_id = visitor\_team\_id  
 INNER JOIN teams T  
 ON T.team\_id = home\_team\_id  
 INNER JOIN snps\_wh.calendar C  
 ON C.full\_date = G.game\_date;  
 """  
 )  
  
cursor\_operational.execute(sql)  
conn\_operational.commit()

**Loading player\_stats Table**

sql = ( """  
 INSERT INTO snps\_wh.player\_stats(points, rebounds, assists, steals, blocks, turnovers, minutes\_played, fgm, fga, ftm, fta, fg3m, fg3a, player\_key, game\_key, team\_key)  
 SELECT pts, reb, ast, stl, blk, turnover, seconds/60 as mintues\_played, fgm, fga, ftm, fta, fg3m, fg3a, player\_id, TPGS.game\_id, team\_id  
 FROM snps\_db.player\_game\_stats TPGS  
 INNER JOIN games G  
 ON G.game\_id = TPGS.game\_id;  
  
 """  
 )  
  
cursor\_operational.execute(sql)  
conn\_operational.commit()

**Loading team\_stats Table**

sql = ( """  
 INSERT INTO snps\_wh.team\_stats(points, rebounds, assists, fg\_pct,   
 ft\_pct, fg3\_pct, game\_key, team\_key)  
 SELECT pts, reb, ast, fg\_pct, ft\_pct, fg3\_pct, game\_id, team\_id   
 FROM game\_stats;  
 """  
 )  
  
cursor\_operational.execute(sql)  
conn\_operational.commit()

**Loading standings Table**

sql = ( """  
 INSERT INTO snps\_wh.standings(wins, losses, date\_key,   
 team\_key)  
 SELECT GW, GL, date\_key, team\_id  
 FROM snps\_db.ranking  
 INNER JOIN snps\_wh.calendar  
 ON full\_date = standings\_date;  
 """  
 )  
  
cursor\_operational.execute(sql)  
conn\_operational.commit()

**Use Cases**

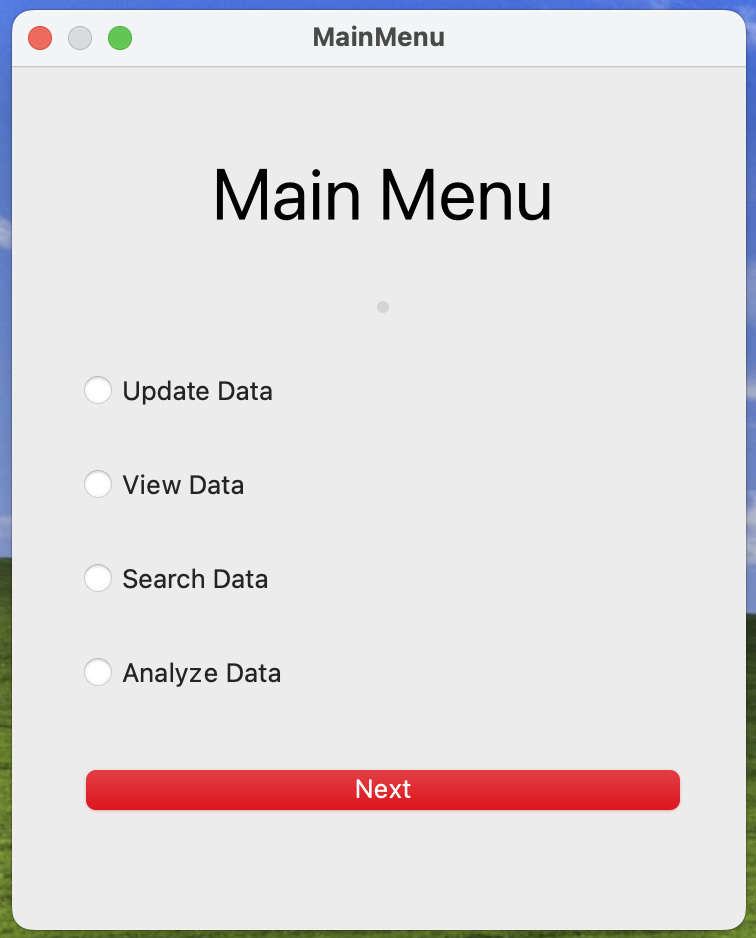
This is the Welcome screen of the NBA Operational and Analysis database GUI application.

Fig 4. The above User Interface/Application gives the user a chance to Update Data, View Data, Search Data, and Analyze the Data.

User can use this window to update players table or team table.

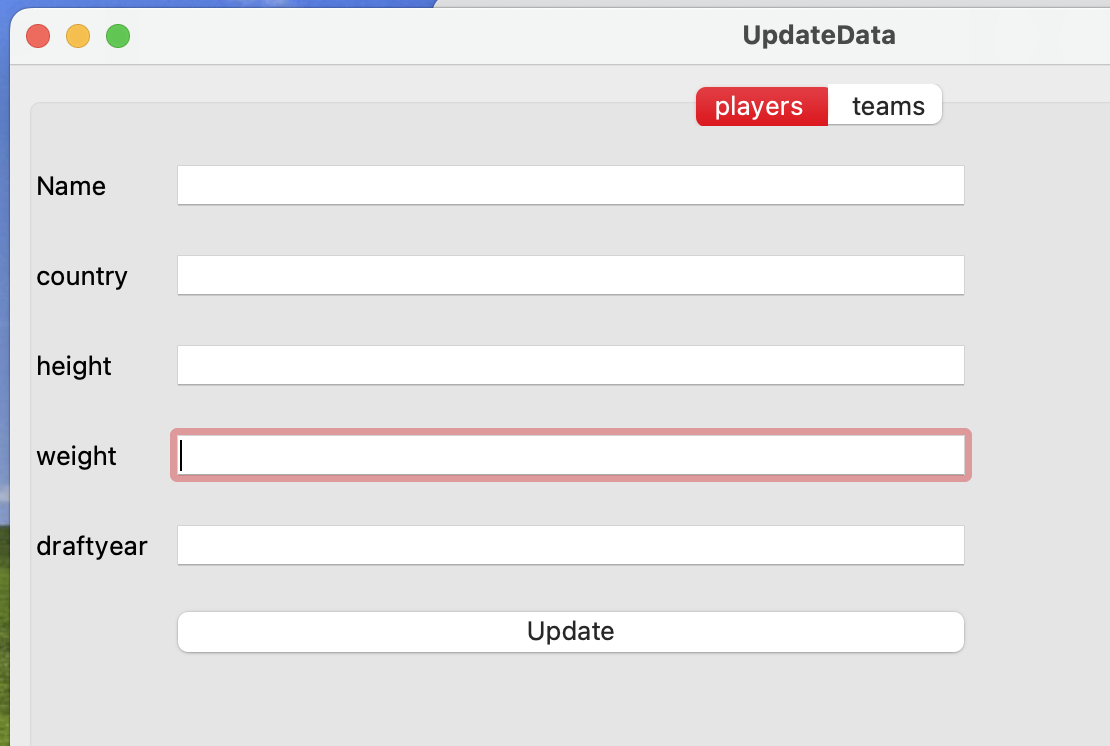
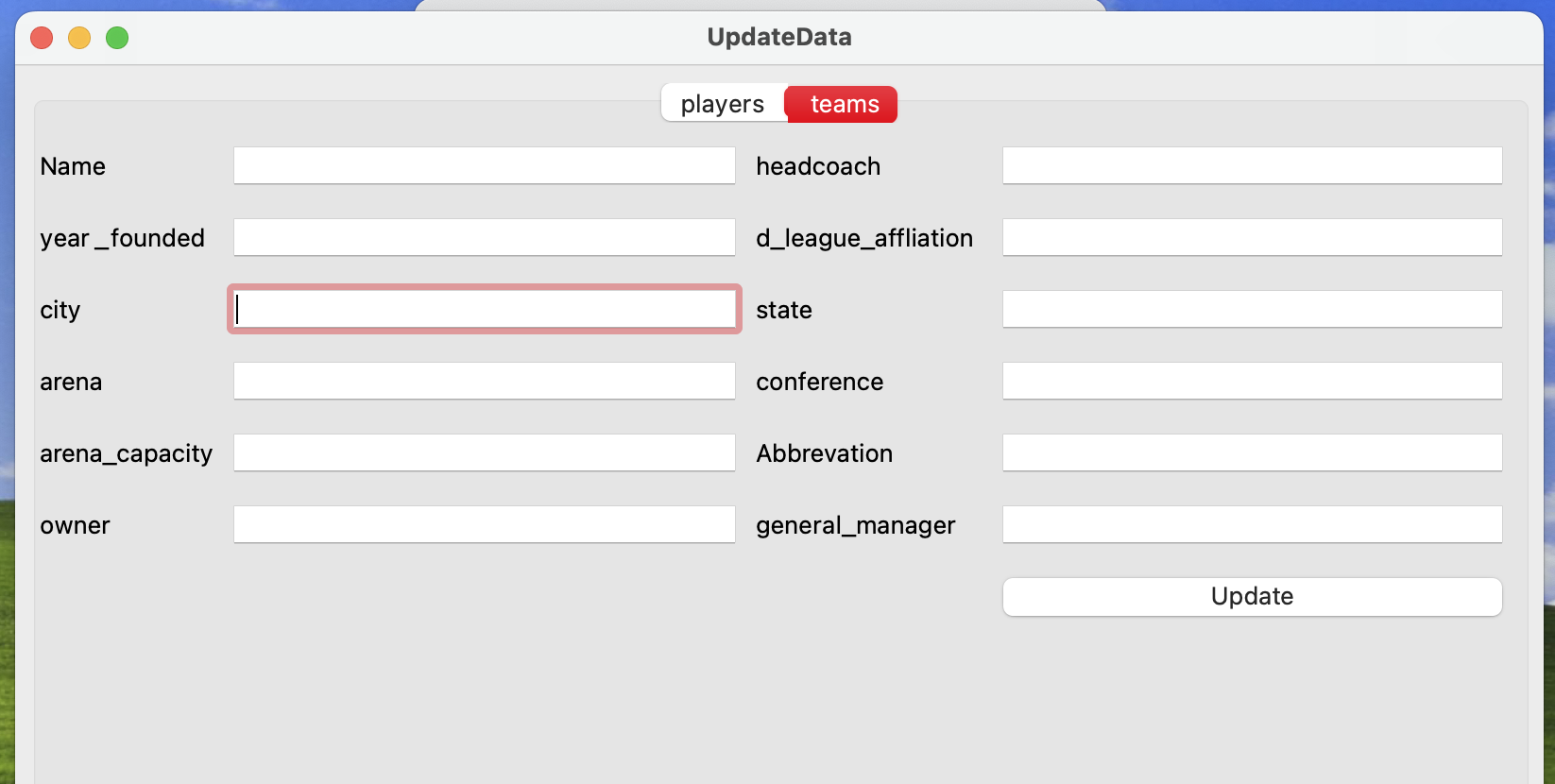
**Update Players**

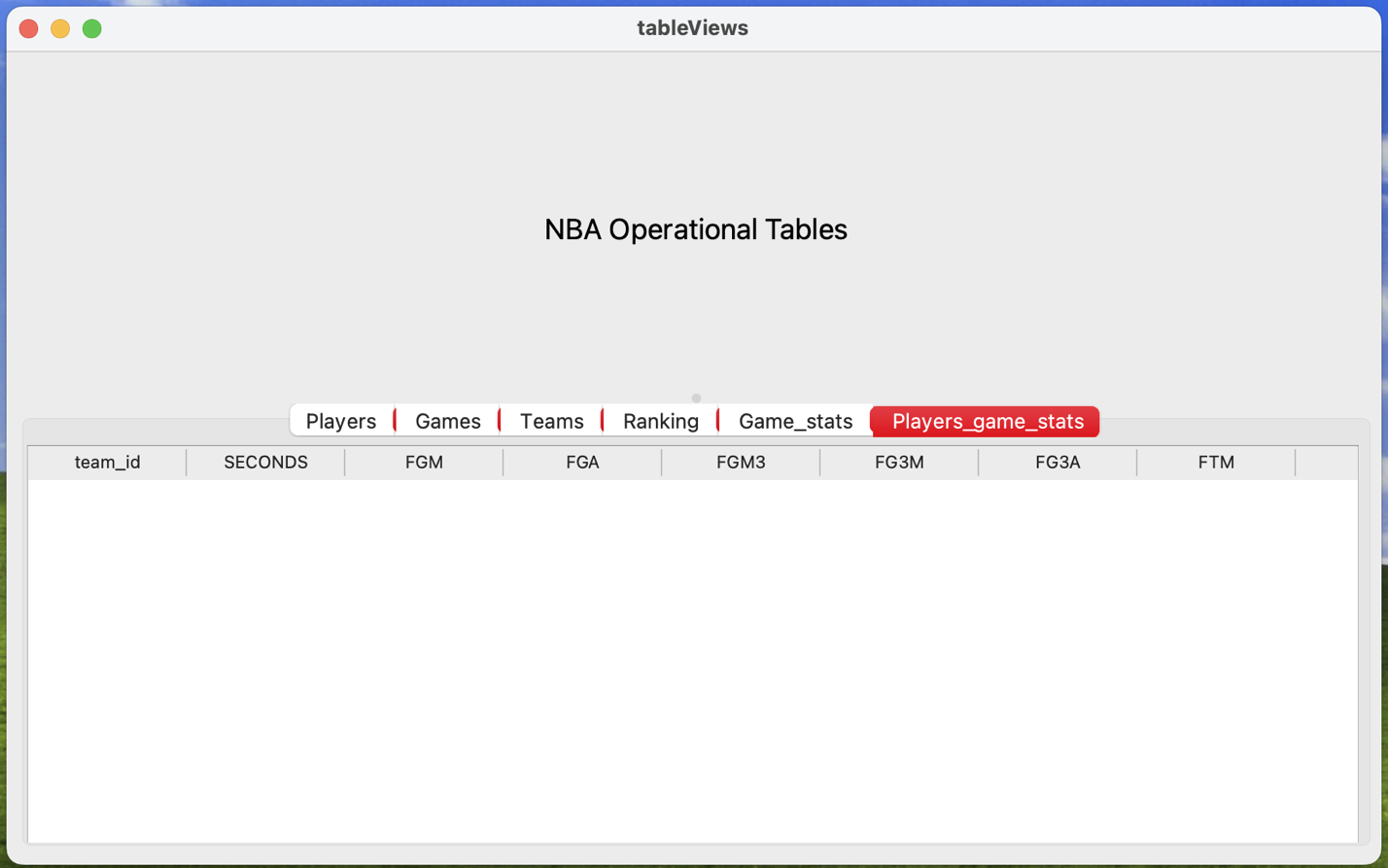
Fig 5. Adding new players who have entered the league.

**Update Teams**

Fig 6. Adding new teams who have entered the league.

**View DataA screenshot of a computer

Description automatically generated with low confidence**

Fig 7. Window showing all the tables and the data.

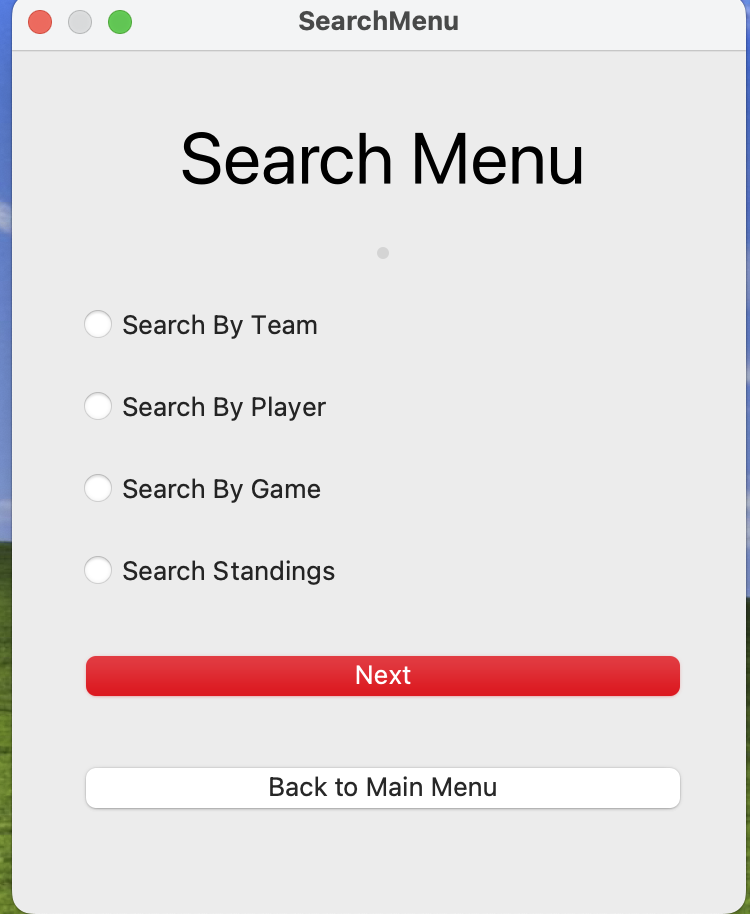
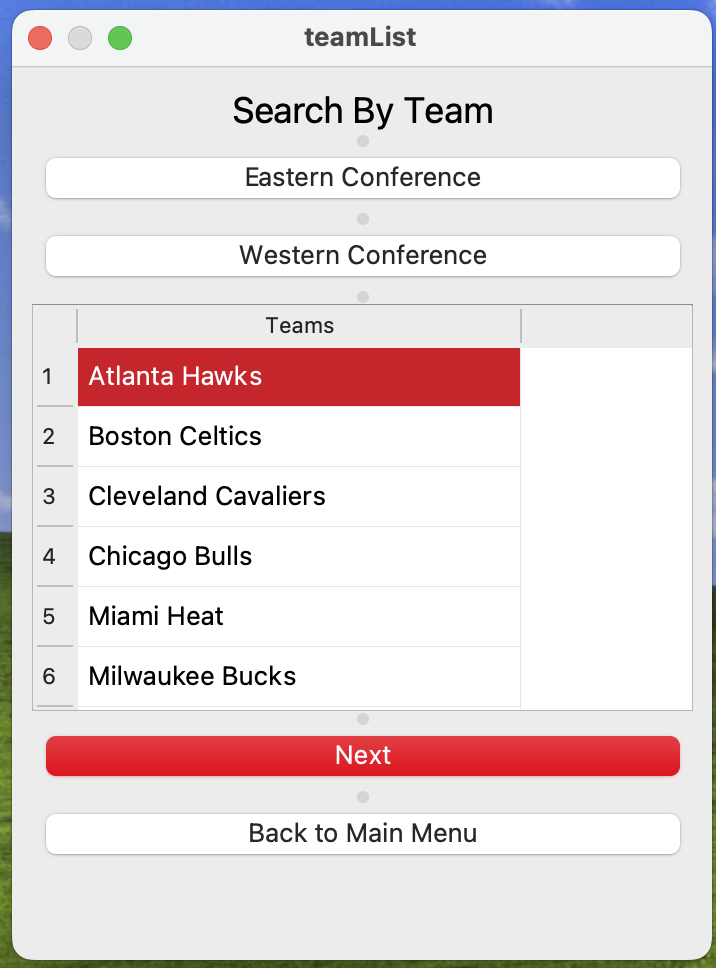
****Search Data**

Fig 8. Menu to search using different parameters.

**Search by Team**

This window helps the user to find a player through the team the player has played for, on selecting the desired team name, list of players will be displayed who have played for that team and finally on selecting a player the player statistics of that player will be displayed.

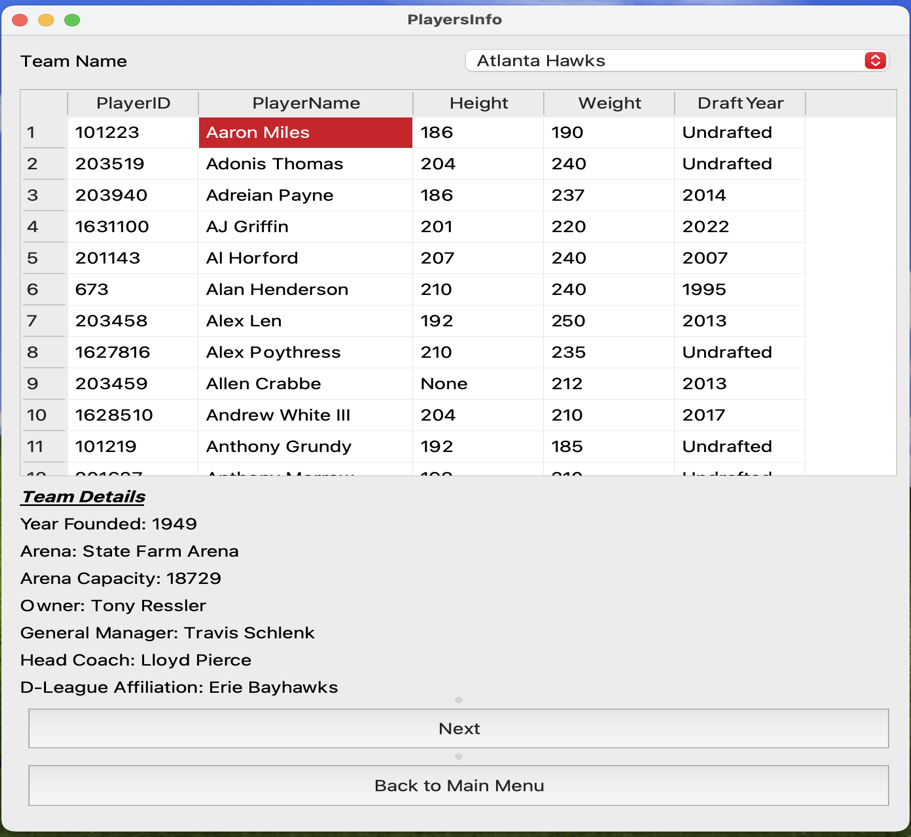
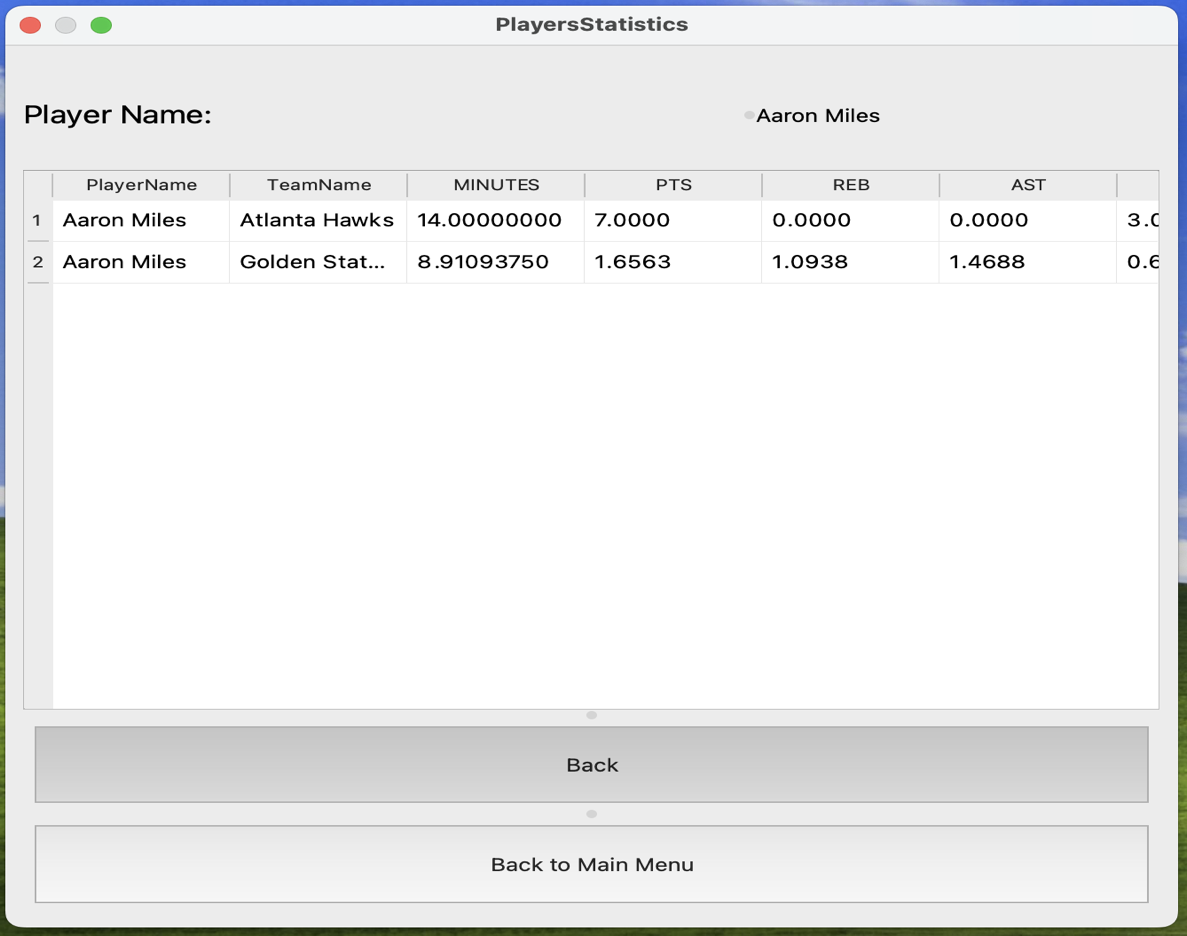
****Fig 9. List of teams in different conference

Fig 10. All team members of the team selected in fig 9.

Fig 11. Player Statistics of the player selected in Fig 10.

**Search by Player**

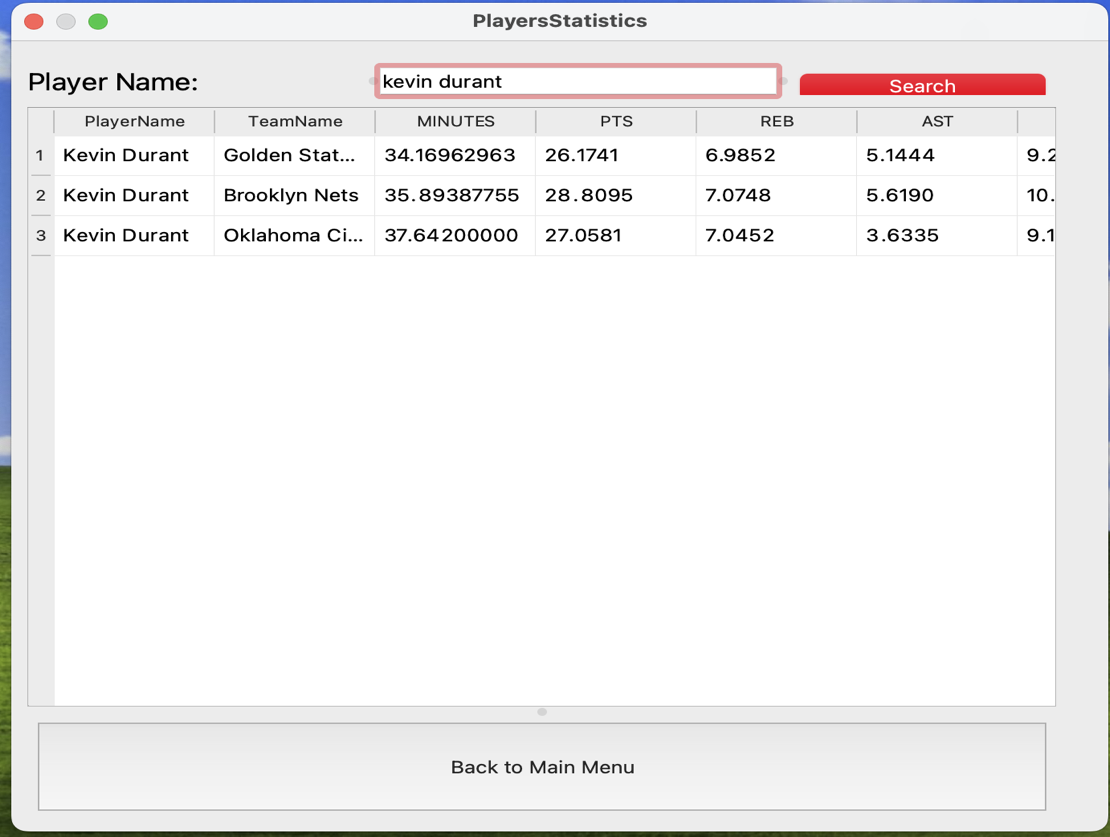
Get player statistics over the season for different teams the player has played.

Fig 12. Player Statistics of Kevin Durant

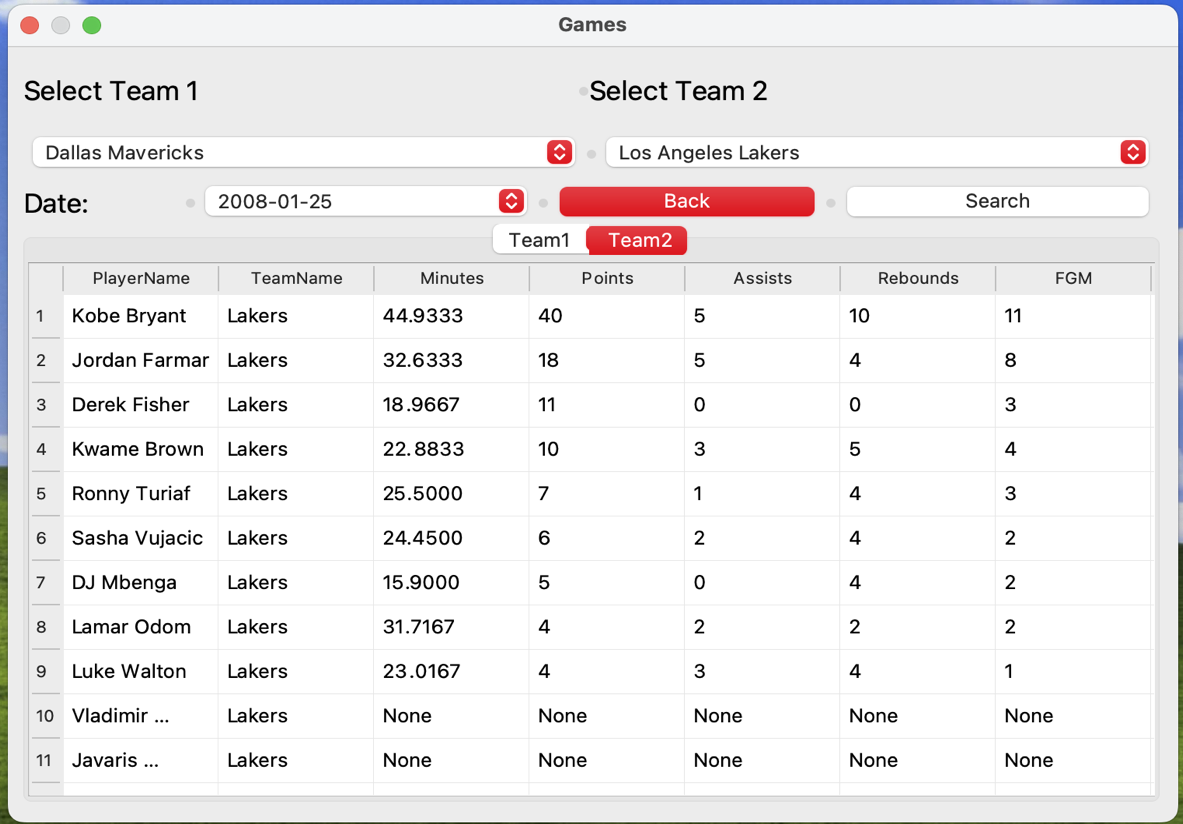
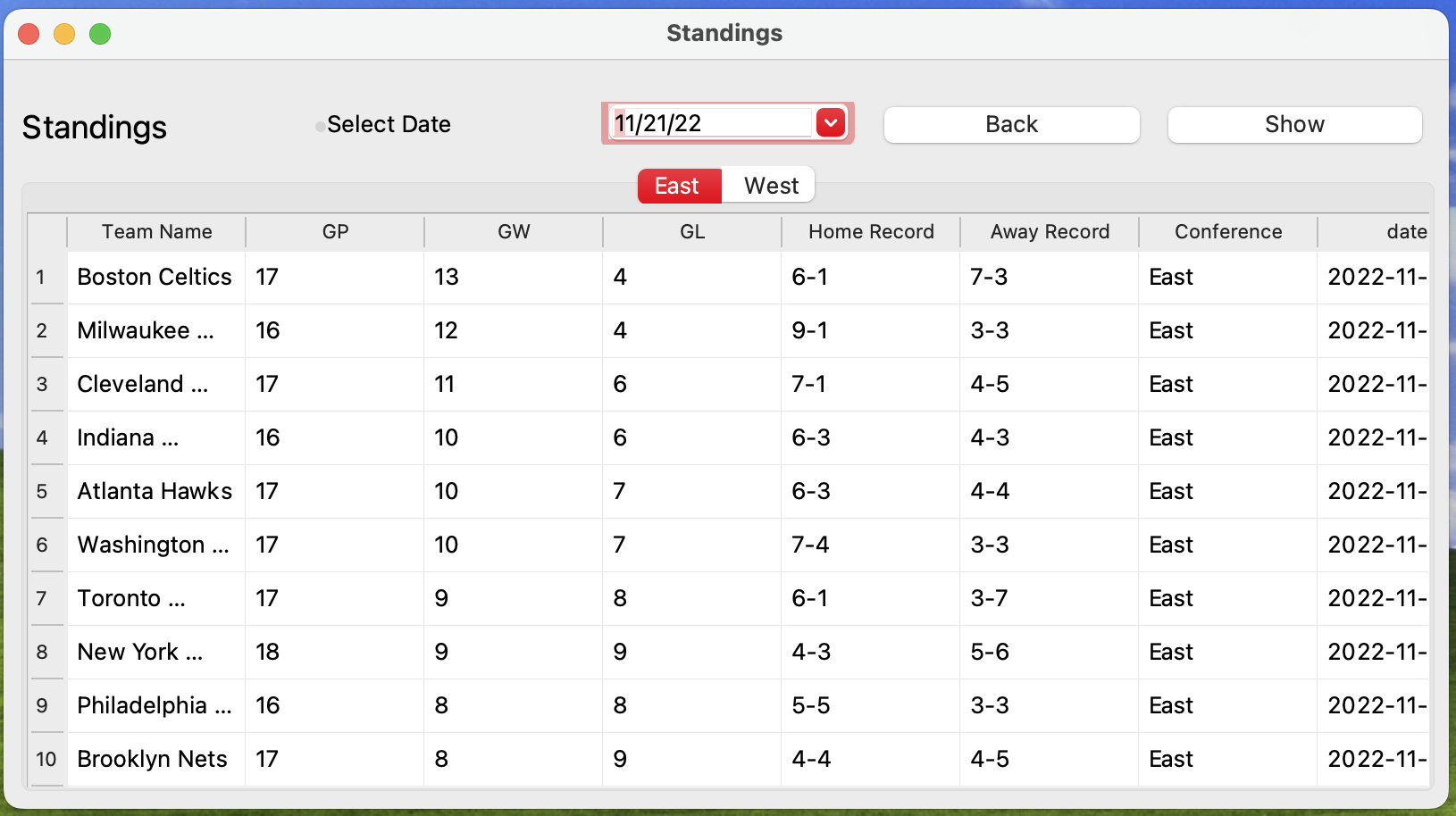
**Search via Game**

Fig 13. BoxScore of a game played by Team1 and Team2 seleted by the user on Date.

**Search by Standings**

Fig 14. Standings of the league on a date selected by the user.

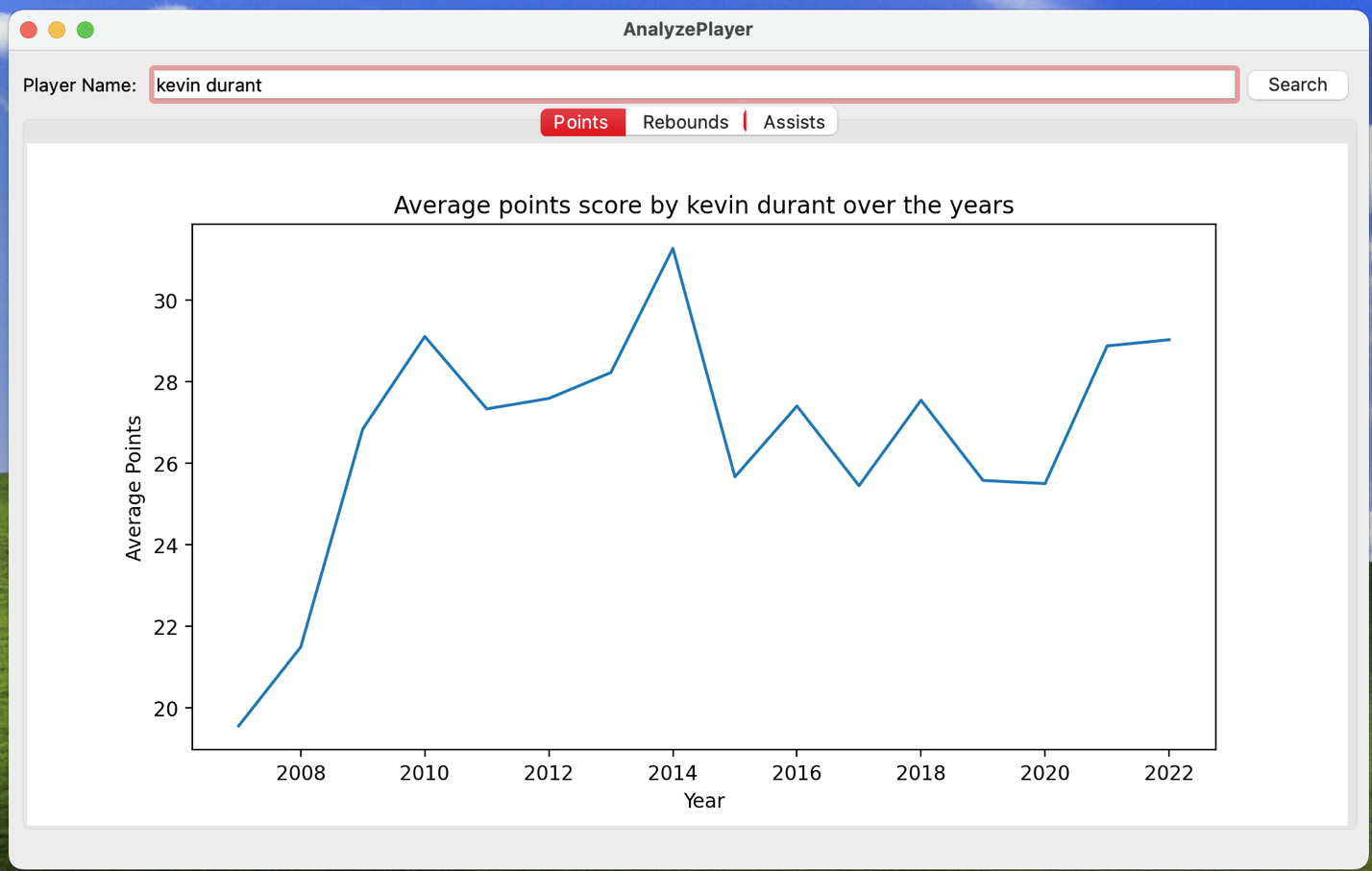
**Analyze Data**

**A screenshot of a screenshot of a game

Description automatically generated with low confidence**Fig 15. Menu for Analyzing data either by Player or Team

**Analyze by Player**

This shows the player performance over the years, performance metrics include Points, Rebounds and Assists and the aggregation done here is Average.

Fig 16. Line graph of Average Points per year for Kevin Durant

**Analysis by team**

This shows the team performance over the years, performance metrics include Win percentage, Points per game, Points allowed per game, Home win percentage, Away win percentage and Margin of Victory.

A picture containing text, software, plot, line

Description automatically generatedFig 17. Line graph of the Win Percentage of Chicago Bulls over the years.

**References**

**[1]** [**https://www.kaggle.com/datasets/nathanlauga/nba-games**](https://www.kaggle.com/datasets/nathanlauga/nba-games)

**[2]** [**https://www.nba.com/stats**](https://www.nba.com/stats)

**[3]** [**https://github.com/swar/nba\_api**](https://github.com/swar/nba_api)