

NBA Database Application (NBADBA)

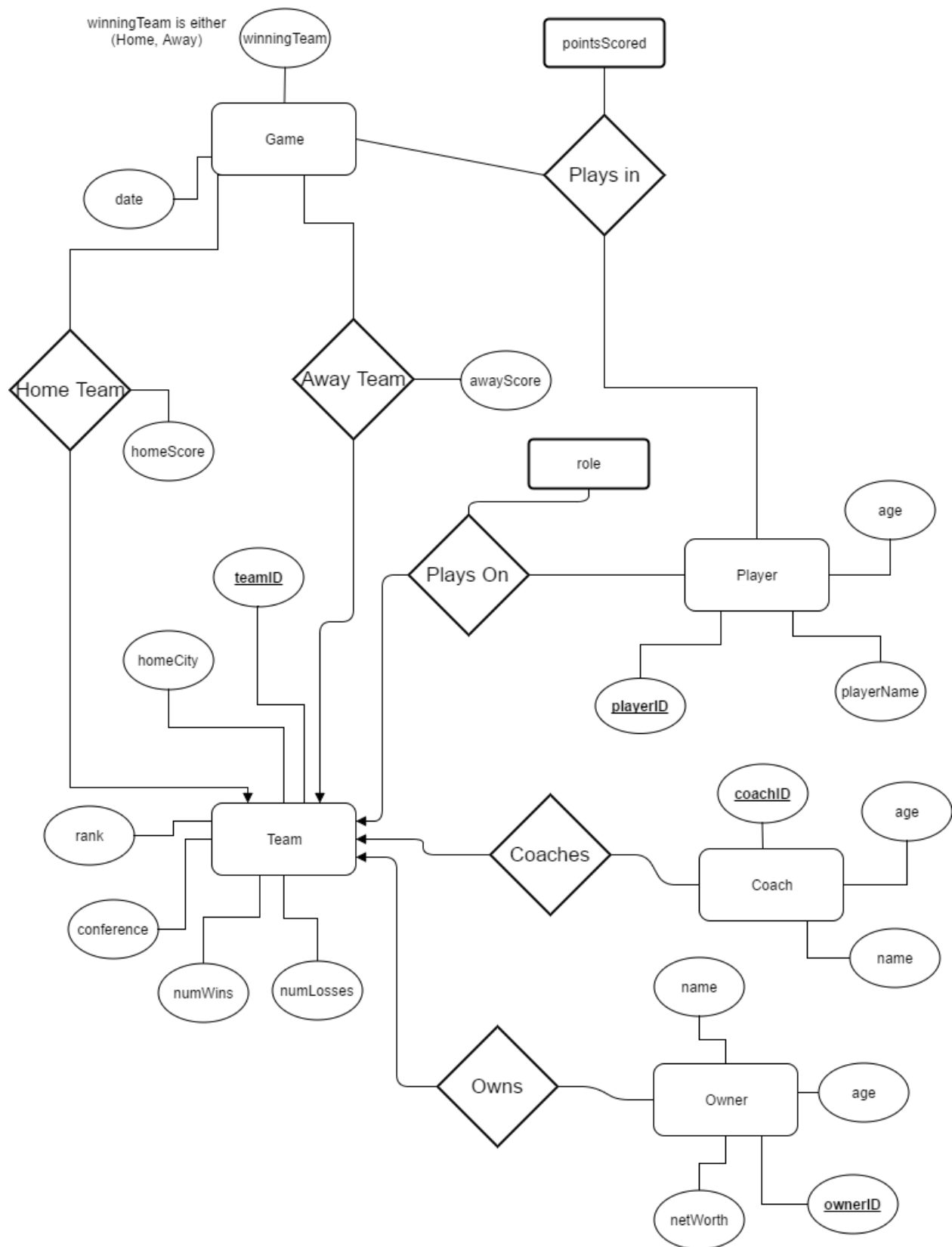
Rahul Pokharna, Ayush Karnawat, Sibi Sengottuvel
EECS 341

Application Background:

The NBA Database Application, or NBADBA for short, is an application that we created to represent the 2015-2016 season of the NBA. A possibility for the future is to use copies of this database for each season, with specifying the year to choose the specific portion of the database to query from. What our application currently does is give basic stats on players, teams, and games. We have tabs present to display all the players, games, and teams, and some simple SQL statements to fill them with information as well as list them all. We have separate tab for all other queries, which are forms where a user can input information and other variables to complete the forms for search queries, finding average heights of roles, and other such queries discussed below. We have information regarding all teams, players, coaches, and owners of each team. The main idea of this project is to create a place to show all of the data for a single season of the NBA in one place. [A link to our hosted application.](#)

Data Description:

The data is of every game of every team, player, coach, owner, game, and each player that plays in each game. There are over 200 coaches, 1230 games, 23000 instances of players playing in games, 30 teams, 380 players, and so on. The data itself is very large, and we restricted it to a single season for this reason. All players are on a team, we have their heights, how many points they scored in every game they played it, all the games, who won and what the scores were, and other data as seen in our tables below. Further in the document, we see a better description of the specifications of our data.



Our E/R Diagram

Functional Dependancies and Schema:

Foreign Keys are italicized, primary keys are underlined

Teams (TeamID, TeamName, Rank, Conference, numWins, numLosses)

F: (TeamID \rightarrow {TeamName, HomeCity, Rank, Conference, numWins, numLosses})

Players (PlayerID, PlayerName, age, *TeamName*, Role)

F: (PlayerID \rightarrow {PlayerName, age, TeamName, Role})

GameScore (GameID, Date, HomeTeamID, AwayTeamID, homeScore, awayScore)

F: ({Date, HomeTeamID, AwayTeamID} \rightarrow {homeScore, awayScore, GameID}, {GameID} \rightarrow {Date, HomeTeamID, AwayTeamID})

Games (GameID, Date, HomeTeamID, AwayTeamID, winningTeam)

F: ({Date, HomeTeamID, AwayTeamID} \rightarrow {WinningTeam, GameID}, {GameID} \rightarrow {Date, HomeTeamID, AwayTeamID})

Coach(coachID, CoachName, age, *TeamName*)

F: (coachID \rightarrow {age, CoachName, TeamName})

Owner (ownerID, name, age, netWorth, *TeamName*)

F: (ownerID \rightarrow {name, age, netWorth, TeamName})

PlaysIn(PlayerID, Date, HomeTeamID, AwayTeamID, PointsScored)

F: ({PlayerID, Date, HomeTeamID, AwayTeamID} \rightarrow {PointsScored})

All of these tables are in BCNF because the primary keys are the only attributes that determine any other attributes. In Games and GameScore GameId also determines the other attributes, but this is still in BCNF because GameId is still a superkey as it is a candidate key. The only reason we included GameId instead of leaving it out is to make it easier to work with the Django implementation

Example Queries Supported by our Database:

For our queries, many of them are not seen easily or are obvious, but we do have many queries in our application. We have some to display every team, game and player, as well as the information inside of them, and some joins to get greater information regarding teams, such as the owner information in the teams tab. Many of our queries are built into the application pages themselves, showing all of the pertinent information there. Our first query below is built into the players tab as well, similar to the query mentioned above. Our Query 2 can be used to

determine many different queries, depending on how the user inputs the data. If the user is able to utilize it fully, that query can be used to determine many things, such as the points scored on average by every single Guard, or the average points on every player taller than 84 inches (7 feet), and so on. The % is a wildcard character, meaning that any length of characters can be in that area, from no characters to the max length, as long as something in the database is able to fill that area to complete the query, that condition will be satisfied.

For dynamic queries, we indicate the dynamic attribute with [attribute_name].

Query 1: Getting the average points scored for each player over all the games from the season. This query is done every time a user clicks on any player, and the resulting value is shown in a table on the specific player's page.

SQL:

```
SELECT avg(PlaysIn.points_scored)
FROM Players, PlaysIn
WHERE Players.player_id = PlaysIn.player_id
GROUP BY Players.player_id
```

RA:

$$\prod_{avg(points_{scored})} (player_id \mathcal{G}_{avg(points_{scored})} \sigma_{playerId=playerId}(Players \bowtie PlaysIn))$$

TRC:

The book says “tuple relational calculus does not have any equivalent of the aggregate operation, and we aren't sure how to implement it since each t is its own tuple, so we have added the aggregate function at the end of the query, even though it logically does not make sense.

$$\{t | \exists (player \in Players \wedge player_id = [PLAYER_ID]) \wedge \exists (playsin \in Playsin \wedge playsin.player_id = player.player_id)) \wedge t[avgPoints] = average(player.pointsScored)\}$$

Query 2: The second query is the most dynamic query we have, where the user can specify many things about players, and get all the results with players that fit that criteria. For example, they can ask for players with name containing 'Jim' who are taller than 72 inches, and have avg points for game greater than 15.

SQL:

```
SELECT Player_Name, Team_name, Role, Height, avg(Points_Scored)
FROM Players p, PlaysIn, Teams t
WHERE Player_ID = Player_ID_ID
      AND player_Name like '%'+inputName+'%'
```

AND height > minHeight
 AND height < maxHeight
 AND Team_Name like '%'+inputTeam+'%'
 AND p.team_id = t.team_id
 AND p.role like '%'+inputRole+'%'
 GROUP BY Player_ID
 Having avg(Points_Scored) > minAvg and avg(points_scored) < maxAvg

RA:

$$\prod_{playerName, teamName, avg(points_{scored}), role, height} \sigma_{avg(points_{scored}) > [MINAVG] \text{ and } avg(points_{scored}) < [MAXAVG]}(player_id \mathrel{G_{avg(points_{scored})}} \\
 \sigma_{playerName=[PLAYERNAME], height > [minHeight], height < [maxHeight], TeamName=[TeamName]}, (Players \bowtie PlaysIn \bowtie Teams))$$

TRC:

$$\{t | \exists (player \in Players \wedge player_name = [PLAYER_NAME] \wedge t[player_Name] = player_name \wedge t[height] \\
 = player.height \wedge player.height < [maxHeight] \wedge player.height > [minHeight] \wedge player.role \\
 = t[role] \wedge \exists (playsin \in Playsin \wedge playsin.player_id = player.player_id \wedge \exists (team \in Teams \wedge team.teamID \\
 = player.teamID \wedge teamName = [TEAMNAME])) \wedge t[avgPoints] \\
 = average(player.pointsScored) \wedge average(player.poinstScored) > [MINAVG] \wedge average(player.poinstScored) \\
 < [MAXAVG])\}$$

Query 3: This query gets the win percentage of the home games played of a specified team, and is displayed behind the scenes on the Team Page

SQL:

SELECT Team_Name, (count(*) / 41.)
 FROM Games g, Teams t
 WHERE t.team_ID = g.home_Team_ID_id
 AND g.winning_Team like 'home'
 AND t.team_ID = [team_id]

$$\text{RA: } \Pi_{teamName, \frac{size}{41}} (\sigma_{teamID=[teamID] \wedge winningTeam='HOME'}(Teams \bowtie_{teamID=homeTeamID} Games))$$

$$\text{TRC: } \{t | (\exists te \in Teams \wedge te[teamID] = [inputTeamID] \wedge (\exists g \in Games \wedge g[homeTeamID] = te[teamID] \wedge t[teamName] = te[teamName]) \wedge t[avgWins] = (size/41))\}$$

Aggregate functions do not work in TRC, but I included to represent that it is there after the calculations had been completed

Query 4: This query returns every player that has played in every home game for a specified team

SQL:

For all players that played in every home game of a dynamically specified team.

```
SELECT p.Player_name
FROM Players p
WHERE NOT EXISTS (SELECT *
                   FROM games g
                   WHERE g.home_team_id_id = [team_id]
                   AND NOT EXISTS (SELECT *
                                   FROM playsIn pi
                                   WHERE g.away_team_id_id = pi.away_team_id_id
                                       AND g.home_team_id_id = pi.home_team_id_id
                                       AND g.date = pi.date
                                       AND p.player_id = pi.player_id_id))
```

RA:

$$\prod_{date, homeTeamId, awayTeamId, playerId} (\sigma_{[teamID]=homeTeamId}(Game \bowtie PlaysIn)) \div \prod_{playerID} Players$$

TRC:

```
{t|∃ (player∈Players ∧ t[playerName]
    = player.playerName0
    ∧ ∃ (team∈Teams ∧ team.teamID
    = [team_id]
    ∧ ¬∃ (game∈Games ∧ team.teamID
    = game.homeTeamID
    ∧ ¬∃ (playsin∈PlaysIn ∧ playsin.homeTeamID = game.homeTeamID ∧ playsin.awayTeamID
    = game.awayTeamID ∧ playsin.date = game.date ∧ playsin.playerID = player.playerID)))}}
```

Implementation:

We are using Django based server, and with Django, we used Python and SQLite3. We programmed the website on Windows and Mac machines, and are running the server on linux instance from Amazon Web Services. Django creates the frontend and connecting that to the backend with Python, we have SQLite3 holding our data. In order to ensure accuracy and

completeness, we get all of our data from stats.nba.com. We used endpoints such as [this link](#) to get data in json format for all of our tables. Even though these endpoints are not documented we were able to find all of the endpoints we needed, and make requests to them. The JSON objects we get from these requests are then parsed and put into the database using the sqlite3 package for python. All of the scripts used to scrape stats.nba.com can be found in the Database folder. We used a software to view the database and all of the data before we created the website itself called DB Browser for SQLite, which showed us everything in the database, constraints and any other pertinent information.

Role Contributions:

Each member contributed heavily to the completion of the project with lots of collaboration throughout the project even though we separated the roles.

Rahul came up with the database design and schemas. He also was able to find the endpoints from stats.nba.com needed to populate the database. He also wrote the code to create the sqlite3 database. He also wrote the SQL queries that were later put into Django.

Sibi wrote the python scripts that make all the requests to stats.nba.com and then parses the JSON data, and then put it into the sqlite database using SQL insert queries.

Ayush made the front-end using Django, injected all of the HTML, and CSS, and added in every query to the website.

We worked together to create a variety of queries to add to our application for users to parse through and view.

What We Learned:

In this project, we learned many more things than we did in class. This project allowed us to actually connect a database to a website and parse through data. It allowed us to learn how to use more versions of SQL to choose which fit our demands the best, as well as how other languages can interact with these SQL variants. We learned Python, reinforced our SQL background from the class, learned how to parse through JSON objects. We also learned a lot of HTML and CSS, as well as how to properly do an E/R Diagram. We learned how to develop

an application as a group as well as how to use git to collaborate on a project together. The project was a good experience in allowing us to see the practical applications of the database, and how we can use it to complete tasks that may be difficult to complete before learning it here.

Screenshots of Application Running and Queries Running:

Query 1 (built in on the player's page) and also other queries shown

The screenshot shows the 'NBA DATABASE' application interface. On the left is a blue sidebar with navigation links: OVERVIEW, TEAMS, PLAYERS, GAMES, and QUERIES. The main content area is titled 'Dashboard' and features a profile for 'LaMarcus Aldridge' (Forward | San Antonio Spurs). Below the profile is a table titled 'Games Played in 2015-2016' with columns for DATE, GAME (AWAY VS. HOME), and POINTS. To the right of the table is a summary box showing 'HEIGHT (IN)' as 83 and 'POINTS PER GAME (AVG)' as 17.9864864865.

DATE	GAME (AWAY VS. HOME)	POINTS
Oct. 28, 2015	Spurs vs. Thunder	11
Oct. 30, 2015	Nets vs. Spurs	10
Nov. 1, 2015	Spurs vs. Celtics	24
Nov. 2, 2015	Spurs vs. Knicks	19
Nov. 4, 2015	Spurs vs. Wizards	10
Nov. 7, 2015	Hornets vs. Spurs	16
Nov. 9, 2015	Spurs vs. Kings	16
Nov. 11, 2015	Spurs vs. Trail Blazers	23
Nov. 14, 2015	76ers vs. Spurs	17
Nov. 16, 2015	Trail Blazers vs. Spurs	6
Nov. 18, 2015	Nuggets vs. Spurs	11
Nov. 20, 2015	Spurs vs. Pelicans	20
Nov. 25, 2015	Mavericks vs. Spurs	18

Query 2 running with sample inputs (testing software)

The screenshot shows the SQL Developer interface. The 'SQL' window contains the following query:

```

1 SELECT Player_Name, Team_Name, Role, Height, avg(Points_Scored)
2 FROM Players P, PlaysIn, Teams T
3 WHERE Player_ID = PlaysIn.Player_ID
4 AND PlaysIn.Team_ID = T.Team_ID
5 AND Height > 50
6 AND Height < 80
7 AND Team_Name like 'Knicks'
8 AND p.Team_ID = T.Team_ID
9 AND p.Role like 'Guard'
10 GROUP BY Player_ID
11 Having avg(Points_Scored) > 1 and avg(Points_Scored) < 100

```

The 'Results' window shows the following data:

player_name	team_name	role	height	avg(Points_Scored)
James Jones	Cavaliers	Guard-Forward	80	3.70033333333333

The 'DB Schema' window shows the database structure, including tables like PlaysIn, Teams, and Players, with their respective columns and data types.

Query 2 again, with new inputs

DB Browser for SQLite - D:\College\Sophomore Year\EECS 341\bnbadba\db.sqlite3

File Edit View Help

New Database Open Database Write Changes Revert Changes

Database Structure Browse Data Edit Pragma Execute SQL

SQL 1

```

1 SELECT Player_Name, Team_Name, Role, Height, avg(Points_Scored)
2 FROM Players p, PlaysIn, Teams t
3 WHERE Player_ID = p.Player_ID_ID
4 AND Player_Name like 'ACuk'
5 AND height > 50
6 AND height < 90
7 AND Team_Name like 'h*'
8 AND p.Team_ID = t.Team_ID
9 AND p.Role like 'F*'
10 GROUP BY Player_ID
11 Having avg(Points_Scored) > 1 and avg(Points_Scored) < 100

```

	player_name	team_name	role	height	avg(Points_Scored)
1	Chris McCullough	Nets	Forward	81	4.666666666666667
2	LaMarcus Aldridge	Spurs	Forward	83	17.9864864864865
3	Stephen Curry	Warriors	Guard	75	30.0632911392405
4	Dante Cunningham	Pelicans	Forward	80	6.05
5	DeMarcus Cousins	Kings	Forward-Center	83	26.8923076923077
6	Marcus Morris	Platons	Forward	81	14.1375
7	Seth Curry	Kings	Guard	74	6.79545454545455
8	Marcus Smart	Celtics	Guard	76	9.14754098360656

8 rows returned in 1ms from: SELECT Player_Name, Team_Name, Role, Height, avg(Points_Scored) FROM Players p, PlaysIn, Teams t WHERE Player_ID = p.Player_ID_ID AND Player_Name like 'ACuk'

DB Schema

Name	Type	S
away_team_id_id	varchar(3)	a
home_team_id_id	varchar(3)	h
winning_team	varchar(20)	w
id	integer	i
date	date	d
points_scored	integer	p
player_id_id	varchar(10)	p
away_team_id_id	varchar(3)	a
home_team_id_id	varchar(3)	h
team_id	varchar(3)	t
team_name	varchar(20)	n
home_city	varchar(20)	h
city	varchar(20)	c

SQL Log Plot DB Schema UTF-8

Query 3 with Cavs as an input

DB Browser for SQLite - D:\College\Sophomore Year\EECS 341\bnbadba\db.sqlite3

File Edit View Help

New Database Open Database Write Changes Revert Changes

Database Structure Browse Data Edit Pragma Execute SQL

SQL 1

```

1 SELECT Team_Name, (count(*) / 41.)
2 FROM Games g, Teams t
3 WHERE t.team_ID = g.home_Team_ID_id
4 AND g.winning_Team like 'home'
5 AND t.team_ID = 'CLE'
6

```

	team_name	(count(*) / 41.)
1	Cavaliers	0.804878048780488

1 rows returned in 4ms from: SELECT Team_Name, (count(*) / 41.) FROM Games g, Teams t WHERE t.team_ID = g.home_Team_ID_id AND g.winning_Team like 'home' AND t.team_ID = 'CLE'

DB Schema

Name	Type	S
PlaysIn	CI	
Teams	CI	
auth_group	CI	
auth_group_permissions	CI	
auth_permission	CI	
auth_user	CI	
auth_user_groups	CI	
auth_user_user_permissions	CI	
django_admin_log	CI	
django_content_type	CI	
django_migrations	CI	
django_session	CI	
sqlite_sequence	CI	
Indices (34)		
Coaches_team_id_d3cee...	CI	
GameScore_away_team_i...	CI	
GameScore_date_home_t...	CI	

SQL Log Plot DB Schema UTF-8

Query 4 with Cleveland as the input (showing all players who played in every home game)

The screenshot shows the DB Browser for SQLite interface. The main window displays a SQL query in the 'SQL 1' editor. The query is designed to find players who have played in every home game for the Cleveland Cavaliers. The results pane shows one player: Tristan Thompson.

```
1 SELECT p.Player_name
2 FROM Players p
3 WHERE NOT EXISTS (SELECT *
4 FROM games g
5 WHERE g.home_team_id_id = 'CLE'
6 AND NOT EXISTS (SELECT *
7 FROM playsIn pi
8 WHERE g.away_team_id_id = pi.away_team_id_id
9 AND g.home_team_id_id = pi.home_team_id_id
10 AND g.date = pi.date
11 AND p.player_id = pi.player_id_id))
12
```

Results:

player_name
1 Tristan Thompson

1 rows returned in 6ms from: SELECT p.Player_name FROM Players p WHERE NOT EXISTS (SELECT * FROM games g WHERE g.home_team_id_id = 'CLE' AND NOT EXISTS (SELECT * FROM playsIn pi WHERE g.away_team_id_id = pi.away_team_id_id AND g.home_team_id_id = pi.home_team_id_id AND g.date = pi.date AND p.player_id = pi.player_id_id))

DB Schema:

Name	Type
PlaysIn	CI
Teams	CI
auth_group	CI
auth_group_permissions	CI
auth_permission	CI
auth_user	CI
auth_user_groups	CI
auth_user_user_permissions	CI
django_admin_log	CI
django_content_type	CI
django_migrations	CI
django_session	CI
sqlite_sequence	CI
Coaches_team_id_d3cee...	CI
GameScore_away_team_L...	CI
GameScore_date_home_t...	CI

Application UI: Overview

The screenshot shows the NBA Database Application UI Overview. The interface includes a sidebar with navigation links, a main content area with sections for Teams, Players, and Games, and a top navigation bar with 'Admin' and 'Log out' links.

NBA DATABASE Dashboard Admin Log out

OVERVIEW

TEAMS

PLAYERS

GAMES

PLAYERS QUERY

TEAMS QUERY

Teams

- Atlanta Hawks
- Brooklyn Nets
- Boston Celtics
- Charlotte Hornets
- Chicago Bulls
- Cleveland Cavaliers
- Dallas Mavericks
- Denver Nuggets
- Detroit Pistons
- Golden State Warriors

View all teams

Players

- Quincy Acy
- Steven Adams
- Aron Affalo
- Alexis Ajinca
- Cole Aldrich
- LaMarcus Aldridge
- Lavoy Allen
- Tony Allen
- Al-Farouq Aminu
- Chris Andersen

View all players

Games

- Oct. 27, 2015: Pistons vs. Hawks
- Oct. 27, 2015: Cavaliers vs. Bulls
- Oct. 27, 2015: Pelicans vs. Warriors
- Oct. 28, 2015: Wizards vs. Magic
- Oct. 28, 2015: 76ers vs. Celtics
- Oct. 28, 2015: Bulls vs. Nets
- Oct. 28, 2015: Jazz vs. Pistons
- Oct. 28, 2015: Hornets vs. Heat
- Oct. 28, 2015: Pacers vs. Raptors
- Oct. 28, 2015: Nuggets vs. Rockets
- Oct. 28, 2015: Cavaliers vs. Grizzlies
- Oct. 28, 2015: Knicks vs. Bucks
- Oct. 28, 2015: Spurs vs. Thunder
- Oct. 28, 2015: Mavericks vs. Suns
- Oct. 28, 2015: Pelicans vs. Trail Blazers
- Oct. 28, 2015: Clippers vs. Kings
- Oct. 28, 2015: Timberwolves vs. Lakers
- Oct. 29, 2015: Grizzlies vs. Pacers
- Oct. 29, 2015: Hawks vs. Knicks
- Oct. 29, 2015: Mavericks vs. Clippers
- Oct. 30, 2015: Heat vs. Cavaliers
- Oct. 30, 2015: Thunder vs. Magic
- Oct. 30, 2015: Jazz vs. 76ers
- Oct. 30, 2015: Raptors vs. Celtics

Team View

NBA DATABASE

OVERVIEW

TEAMS

PLAYERS

GAMES

PLAYERS QUERY

TEAMS QUERY

Chicago Bulls

Eastern Conference | Rank: 9

Players

ID	NAME	HEIGHT (IN)	ROLE
201166	Aaron Brooks	72	Guard
202710	Jimmy Butler	79	Forward
203915	Spencer Dinwiddie	78	Guard
2399	Mike Dunleavy	81	Guard-Forward
1626245	Cristiano Felicio	81	Forward-Center
2200	Pau Gasol	84	Center
201959	Taj Gibson	81	Forward
203200	Justin Holiday	78	Guard
203926	Doug McDermott	80	Forward
202703	Nikola Mirotic	82	Forward
202734	ETwaun Moore	76	Guard
201149	Joakim Noah	83	Center
1626171	Bobby Portis	83	Forward
201565	Derrick Rose	75	Guard
203503	Tony Snell	79	Guard-Forward

Owners

NAME	AGE	NETWORTH
Jerry Reinsdorf	81	\$1200000000

Coaches

COACH ID	NAME	TYPE
HOI088615	Fred Holberg	Head Coach
BR0603029	Randy Brown	Assistant Coach
HEN687510	Charlie Henry	Assistant Coach

Player View

NBA DATABASE

OVERVIEW

TEAMS

PLAYERS

GAMES

PLAYERS QUERY

TEAMS QUERY

Dashboard

Joakim Noah

Center | Chicago Bulls

Games Played in 2015-2016

DATE	GAME (AWAY VS. HOME)	POINTS
Oct. 27, 2015	Cavaliers vs. Bulls	0
Oct. 28, 2015	Bulls vs. Nets	0
Oct. 30, 2015	Bulls vs. Pistons	2
Nov. 1, 2015	Magic vs. Bulls	8
Nov. 3, 2015	Bulls vs. Hornets	0
Nov. 5, 2015	Thunder vs. Bulls	4
Nov. 7, 2015	Timberwolves vs. Bulls	3
Nov. 13, 2015	Hornets vs. Bulls	3
Nov. 16, 2015	Pacers vs. Bulls	8
Nov. 18, 2015	Bulls vs. Suns	0
Nov. 20, 2015	Bulls vs. Warriors	0
Nov. 24, 2015	Bulls vs. Trail Blazers	7
Nov. 27, 2015	Bulls vs. Pacers	0
Nov. 30, 2015	Spurs vs. Bulls	8

HEIGHT (IN)	POINTS PER GAME (AVG)
83	4.27586206897

Query #1: With Sample Input

Search for players with common attributes

Player name:

Min height:

Max height:

Team name:

Role:

Min points scored:

Max points scored:

Results Returned From Query #1

NBA DATABASE

OVERVIEW

TEAMS

PLAYERS

GAMES

PLAYERS QUERY

TEAMS QUERY

WIN PERCENTAGE QUERY


Dashboard

Admin

Log out

Players

NAME	TEAM	ROLE	HEIGHT (IN)	POINTS PER GAME (AVG)
James Johnson	Raptors	Forward	81	5.0350877193
James Ennis III	Pelicans	Forward	79	7.22727272727
James Michael McAdoo	Warriors	Forward	81	2.85365853659
LeBron James	Cavaliers	Forward	80	25.2631578947
James Jones	Cavaliers	Guard-Forward	80	3.70833333333

Made with  in 2017

Query #2: With Sample Input

Find the Home Win Percentage for Any Team

Team id:

Results Returned From Query #2

Teams	
TEAM	HOME WIN-RATE PERCENTAGE
Cavaliers	0.80487804878

Query #3: With Sample Input

Who played in every home game for this team?

Team id:

Results Returned From Query #3

Teams	
NAME	TEAM
Trevor Ariza	HOU
Corey Brewer	HOU
James Harden	HOU