Group 8

NBA Database Web Application

Group members: Yingzhe Chen, Hui Sui, Benjamin Wong, Zhihao Yan

1. Introduction

For the NBA league, player and franchise data are critical for them to make better decisions, thus a thorough and convenient system will help them to access data much easier. Here, we propose a system for the team managers to get the player data for each game during the season so that they can make better decisions on trading. The system also provides coaches a better platform so that they can find a perfect lineup for each night. A player can find their performance for each game in our system so that they know what to improve on. It stores all the historical data so all the hall of fame election nominations have a different perspective. Therefore , our system will promote the development of the league in the long term. We believe our system will have a comparative business value.

Our web application has four pages: homepage, players page, teams page, and games page. The homepage contains a table that displays all of the players information, a table that displays all of the teams information, and a table that displays all of the games information with a dropdown to filter seasons. On each of the rest pages, users can find more detailed information on players, teams and games and do some search and filtering.

2. Architecture

In this project, we used AWS DBMS to store our data and MySQL RDS to create data schemas and fill in tables. We used Node.js for server side development, and React.js for client side development. Also, we used github for version control and collaboration.

3. Data

Link to dataset:
https://www.kaggle.com/datasets/nathanlauga/nba-games?resource=download&select=ranking.cs

We utilized the following datasets:

Games.csv - All NBA games from year 2004 to 2021 with generic details such as Game season and game id. It contains 25796 rows x 21 attributes, including GAME_DATE_EST, GAME_ID, etc.

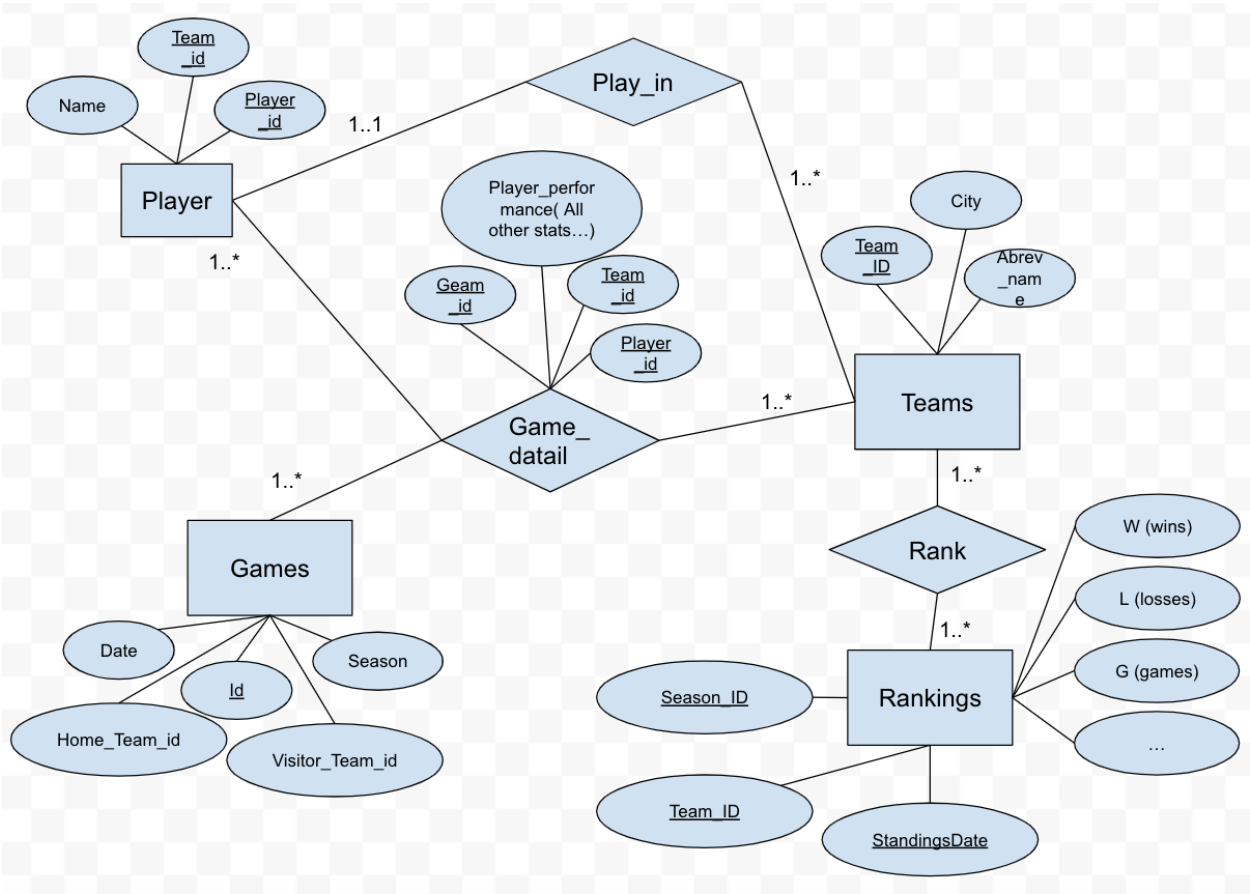
Games_details.csv - All NBA games detail including all the player on the court and their statistics (Time, Score, Rebound, Assist, steal….). It contains 645953 rows x 29 attributes, including TEAM_ID, TEAM_CITY, PLAYER_ID, etcc

Players.csv - data description for each NBA player, with columns such as player id and team id. It contains 7228 rows x 4 attributes, including TEAM_ID, PLAYER_NAME, etc.

Ranking.csv - Ranking of NBA teams for each day, given their stats. It contains 201793 rows x 13 attributes, including TEAMS, CONFERENCE, TEAM_ID, etc…

Teams.csv - All NBA teams, with columns such as year founded, team nickname. It contains 30 rows x 13 attributes, including TEAM_ID, TEAM_NAME, CITY, etc…

4. Database



Games.csv - 25796 rows x 21 attributes
Games_details.csv - 645953 rows x 29 attributes

Players.csv - 7228 rows x 4 attributes
Ranking.csv - 201793 rows x 13 attributes
Teams.csv - 30 rows x 13 attributes

5. Web App description

Our web application has four pages: homepage, players page, teams page, and games page.
- Homepage: contains a table that displays all of the player's information, a table that displays all of the teams information, and a table that displays all of the games information with a dropdown to filter seasons. When a user clicks on a row, it brings the user to the corresponding players/teams/games page and displays more detailed information.

- Players page: on the top, there are three search conditions: name, team and season. Users can filter players using any of all of these three conditions. After the search field, it displays each players performance in each season. Attributes in this Table includes a player's name, a player's team, season, season average points, season average assist, and season average rebound. When a user clicks on a player's name, another table will Display at the bottom part of the page. This table displays the game details of all the games that player has been played in. Attributes in this table include player name, season, date of the Game, minutes, FG3_PCT, FG_PCT, FT_PCT, PTS, AST, and REB.

- Teams page: On the top, it displays Team Rankings for a particular season. The teams are ranked based on their Wins for each season, which is a drop down option. Below, there is a dropdown menu to filter teams based on their locations (conference) such as East, West and All. The information includes average points earned as Home and average points earned as Away for each team. When a user clicks on a team, all current players for the team are displayed and the team's logo and conference logo are displayed as well..

- Games page: On the top, there are three search conditions: home team, visitor team and season. Users can filter players using any of all of these three conditions. After the search field, it displays detailed information of all games including home team, visitor team, home points, visitor points, and date of the game. When a user clicks on a game, below the table it displays a bar graph comparing the two teams' performance in that game. Performance metrics include points, assist, rebounds, FG accuracy, 3 point FG accuracy, and free throw accuracy are shown.

6. API Specification

See appendix

7. Queries

1. The following query displays the top table on the players page. It shows all players

information

```
WITH g AS (
    SELECT games.SEASON, games.GAME_DATE_EST, gd.PLAYER_ID, gd.MIN,
    gd.FG3_PCT, gd.FG_PCT, gd.FT_PCT, gd.PTS, gd.AST, gd.REB
    FROM games
    JOIN games_details gd on games.GAME_ID = gd.GAME_ID
)
SELECT p.PLAYER_NAME AS Name, g.SEASON, g.GAME_DATE_EST AS Date,
g.MIN, g.FG3_PCT, g.FG_PCT, g.FT_PCT, g.PTS, g.AST, g.REB
FROM players p
LEFT JOIN g ON p.PLAYER_ID = g.PLAYER_ID
WHERE g.PLAYER_ID = ${id}
ORDER BY GAME_DATE_EST DESC
LIMIT ${pagesize} OFFSET ${start}
```

---

2. The following query displays filtered players based on the search conditions name, team and season on the players page

---

```
SELECT p.PLAYER_ID, p.SEASON, p.PLAYER_NAME, AVG(g.PTS) AS AVG_PTS,
AVG(g.AST) AS AVG_AST, AVG(g.REB) AS AVG_REB
FROM players p
JOIN games_details g on p.PLAYER_ID = g.PLAYER_ID
JOIN teams t on g.TEAM_ID = t.TEAM_ID
WHERE PLAYER_NAME LIKE '%${Name}%' AND t.NICKNAME LIKE
'%${Team_name}%'  AND  SEASON = ${Season}
GROUP BY p.PLAYER_ID, p.SEASON, p.PLAYER_NAME
ORDER BY p.PLAYER_NAME
LIMIT ${pagesize} OFFSET ${start}
```

---

3. The following query filters game details based on home team, visitor team and season on the games page:

```
SELECT game_id AS GameId, game_date_est AS Date, t.NICKNAME  as Home,
PTS_home, p.NICKNAME as Visitor, PTS_away
FROM games g
```

```
JOIN teams t on g.HOME_TEAM_ID = t.TEAM_ID
JOIN teams p on g.VISITOR_TEAM_ID = p.TEAM_ID
WHERE t.NICKNAME LIKE "%${home}%"
AND p.NICKNAME LIKE "%${visitor}%"
AND season = ${season}
ORDER BY game_date_est DESC
```

---

4. The following query displays the team logo and current players on a team on the teams page:

```
SELECT team_id AS TeamId, nickname AS name, LOGO AS  logo,
CURRENT_PLAYER_1 as cp1, CURRENT_PLAYER_2 AS cp2,
CURRENT_PLAYER_3 AS cp3, CURRENT_PLAYER_4 AS cp4,
CURRENT_PLAYER_5 AS cp5, CURRENT_PLAYER_6 AS cp6,
CURRENT_PLAYER_7 AS cp7, CURRENT_PLAYER_8 AS cp8,
CURRENT_PLAYER_9 AS cp9, CURRENT_PLAYER_10 AS cp10,
CURRENT_PLAYER_11 AS cp11, CURRENT_PLAYER_12 AS cp12,
CURRENT_PLAYER_13 AS cp13, CURRENT_PLAYER_14 AS cp14,
CURRENT_PLAYER_15 AS cp15, CURRENT_PLAYER_16 AS cp16,
CURRENT_PLAYER_17 AS cp17, CURRENT_PLAYER_18 AS cp18,
CONFERENCE_LOGO AS conf_logo
FROM teams
WHERE team_id = ${id}
```

---

5. The following query displays team statistics based on selected region:

```
SELECT team_id AS TeamId, nickname, AVG(g.PTS_home)  AS avg_pts_home,
AVG(p.PTS_away) AS avg_pts_away
FROM teams
JOIN games g ON teams.team_id = g.home_team_id
JOIN games p ON teams.team_id = p.VISITOR_TEAM_ID
WHERE conference LIKE "%${conf}%"
AND g.season=${season}
AND p.season=${season}
GROUP BY TeamId;
```

---

8. Performance evaluation

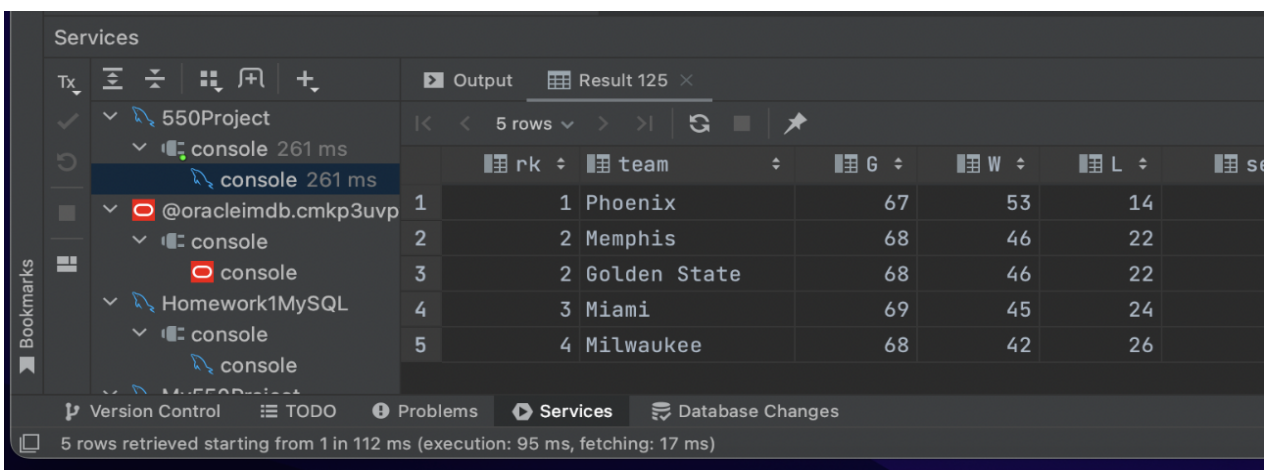
- Optimization: Two basic techniques we have applied to optimize our query are using index and decomplex the query.. By setting index on season_id, we are able to reduce the running time of the following queries from 95 ms to 59 ms because the query has a where clause in season_id which require a full scan of the dataset by season_id, using index reduce the amount of time of find specific season_id from O(n) to O(logn) or even O(1):

Query :

(we call it max_query for latter)

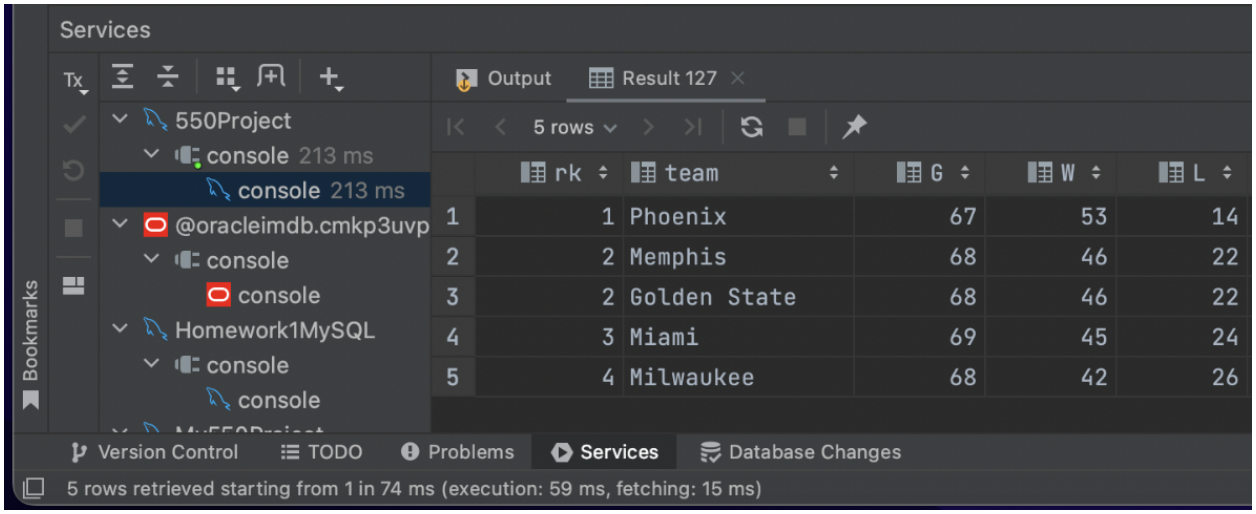
```
WITH TEMP AS (
        SELECT season_id, team, MAX(G) AS G, MAX(W) AS W, MAX(L) AS L FROM ranking
        WHERE SEASON_ID=2021
        GROUP BY season_id, team)
Select rk, team, G, W, L, season_id AS season from (
SELECT dense_rank() OVER (PARTITION BY season_id ORDER BY W DESC) AS rk, team, G, W, L, season_id
FROM TEMP) a WHERE rk <= 5
LIMIT 5;
```

Before index optimization:



	rk	team	G	W	L
1	1	Phoenix	67	53	14
2	2	Memphis	68	46	22
3	2	Golden State	68	46	22
4	3	Miami	69	45	24
5	4	Milwaukee	68	42	26

After index optimization:



	rk	team	G	W	L
1	1	Phoenix	67	53	14
2	2	Memphis	68	46	22
3	2	Golden State	68	46	22
4	3	Miami	69	45	24
5	4	Milwaukee	68	42	26

Decomplex the query:

By reducing the amount of feature we need to selection in intermediate table, performing join instead of where clause, reducing the amount of tuple per table before join, avoiding multiple join, we are able to reduce the running time significantly, one of the example is showing below

Before decomplexing:

```
NBA_DATA> WITH p AS ( SELECT PLAYER_NAME AS Name, PLAYER_ID, t.NICKNAME AS Team, SEASON AS Season
                          FROM players p
                          JOIN teams t on p.TEAM_ID = t.TEAM_ID
                          WHERE SEASON = 2019
                          )
          SELECT p.Name, p.PLAYER_ID AS id, p.Team, p.Season, AVG(g.PTS) AS AVG_PTS, AVG(g.AST) AS AVG_AST, AVG(g.REB) AS AVG_REB
          FROM games_details g
          JOIN  p on g.PLAYER_ID = p.PLAYER_ID
          GROUP BY p.SEASON, p.Name, p.PLAYER_ID, p.Season
          ORDER BY p.Name
[2022-12-16 20:39:46] 500 rows retrieved starting from 1 in 1 s 219 ms (execution: 1 s 166 ms, fetching: 53 ms)
```

(we call it avg_query for latter)
After decomplexing:

```
NBA_DATA> SELECT p.PLAYER_ID, p.SEASON, p.PLAYER_NAME, AVG(g.PTS) AS AVG_PTS, AVG(g.AST) AS AVG_AST, AVG(g.REB) AS AVG_REB
          FROM players p
          JOIN games_details g on p.PLAYER_ID = g.PLAYER_ID
          JOIN teams t on g.TEAM_ID = t.TEAM_ID
          WHERE SEASON = 2019
          GROUP BY p.PLAYER_ID, p.SEASON, p.PLAYER_NAME
[2022-12-16 20:39:05] 500 rows retrieved starting from 1 in 1 s 17 ms (execution: 975 ms, fetching: 42 ms)
```

In Summary:

	Naive	**Decomplexing**	**Indexing**
Max_query	116ms	**95ms**	**59ms**
Avg_query	1s166ms	**975ms**	**968ms**

9. Technical challenges

One of the technical challenges we encountered was that we made use of multiple datasets, however, when we were trying to join them, we found they had mismatched values in the common rows. So, we spend some time proprocess the mismatched values. Another technical challenge was that since we were not familiar with Github, we spent some time figuring out how to use Github on code collaboration and version control. In addition, all of our group members are new to React.js, we spent a lot of time looking over examples and tutorials online to make this happen.

10. Extra Effort

We have written a python file to preprocess data, including drop unnecessary columns, combining columns, drop nan values, drop unmatched Ids etc. We also wrote a file to scrape the logo for each team from online sources.

Credits - NBA team logos: https://boundtoball.com/all-30-nba-team-logos/

Appendix

1
Route : /player

Description: In this function, given a input player_id, return the corronsding player's behavior statistic in every single game.
Route Parameter(s): id (int) (default: 201166),
Query Parameter(s): **page** (int)*, **pagesize** (int)* (default: 10)
Route Handler: player(req, res)
Return Type: JSON
Return Parameters: { (JSON array of { Name (string), Season (int), Date(date), MIN(string), FG3_PCT(int), FG_PCT(int), FT_PCT(int), PTS(int), AST(int), REB(int) }) }
Expected (Output) Behavior:

- Case 1: If the page parameter (**page**) is defined. Return match entries with all the above return parameters for that page number by considering the **page** and **pagesize** parameters. Return the players with the given player_id
- Case 2: If the page parameter (**page**) is not defined. Return all match entries with all the above return parameters. Consider only the player specified by **player_id**

2
Route: /search/players

Description: This function is used to get players information that corresponds to searched conditions. Users can search for players by players' names, their team names and seasons they played.
Route Parameter(s): **Name**(String), **Team**(String), **Season**(int) (default: 2019)
Query Parameter(s): **page** (int)*, **pagesize** (int)* (default: **10**)
Route Handler: search_players(req,res)
Return Type: JSON
Return Parameters: { (JSON array of { { Name (string) ,Team (string), Season (int) , AVG_PTS (int), AVG_AST(int), AVG_REB(int)}) }
Expected (Output) Behavior:
- Case 1: If the page parameter (**page**) is defined. Return player entries with all the above return parameters for that page number by considering the **page** and **pagesize** parameters. Return the players with the given **name**, **team**, and **season**.
- Case 2: If the page parameter (**page**) is not defined. Return all player entries with all the above return parameters. Consider only the player specified by the given **name**, **team**, and **season**.

#3
Route: /search/game_details

Description: This function is used to get games information that corresponds to searched conditions. Users can search for players by home team, visitor team, and seasons.
Route Parameter(s): **home team**(String), **visitor team**(String), **Season**(int) (default: 2021)
Query Parameter(s): **page** (int)*, **pagesize** (int)* (default: **10**)
Route Handler: search_players(req,res)
Return Type: JSON
Return Parameters: { (JSON array of { { Home (string) ,visit (string), HomePts(int), HomePts(int), Date (date))}) }
Expected (Output) Behavior:
- Case 1: If the page parameter (**page**) is defined. Return match entries with all the above return parameters for that page number by considering the **page** and **pagesize** parameters. Return the players with the given **home team**, **visitor team**, and **season**.

- Case 2: If the page parameter (**page**) is not defined. Return all match entries with all the above return parameters. Consider only the player specified by the given **home team**, visitor **team**, and **season**.

4
Route: /games/:season

Description: Gets all the games based on the season filter
Route Parameter(s): Season
Query Parameter(s): season (int), page (int), pagesize (int)
Route Handler: all_games(req, res)
Return Type: JSON
Return Parameters: { (JSON array of { Game_id (int), game_date_est (datetime), nickname_home (string), nickname_visitor (string), Pts_home (int), Pts_away (int) }
Expected (Output) Behavior:
- Case: If team home id found, team visitor id found, and season found, then return corresponding information for all game details between home team and visitor team in that particular season.

#5
Route: /teams

Description: Gets all teams
Route Parameter(s): None
Query Parameter(s): page (int), pagesize (int)
Route Handler: all_teams(req, res)
Return Type: JSON
Return Parameters: { (JSON array of { team_id (int), nickname (string), abbreviation (string), year founded (int), arena (string) }
Expected (Output) Behavior: Prints information for all teams

#6
Route: /team_players

Description: Gets all current players belonging to the given team. Return their names with the logo of their team and conference.
Route Parameter(s): id(int)
Query Parameter(s): None
Route Handler: `team_players`(req, res)
Return Type: JSON

Return Parameters: { (JSON array of { team_id (int), name (string), LOGO (string), current_players * 18 (int), onf_logo (string) }
Expected (Output) Behavior: Prints current players' names with the logo of their team and conference.

7
Route : /player_in_game

Description: Returns an array of players who played in a given game
Route Parameter(s): None
Query Parameter(s): game_id (int) default:52000211
Route Handler: player_in_game(req, res)
Return Type: JSON
Return Parameters: { (JSON array of { Player_name (string), Team_id (int), Player_id (int), Season (int) }) }
Expected (Output) Behavior: If the game_id is defined then return the players who played in that game. Otherwise, return players who played in the default game_id 52000211

8
Route : /top_teams

Description: Returns an array of teams win most games in the given season.
Route Parameter(s): None
Query Parameter(s): season(int)
Route Handler: top_5_teams(req, res)
Return Type: JSON
Return Parameters: { (JSON array of { Rank (int), Team (string), Games (int), Wins (int), Losses (int) }) }
Expected (Output) Behavior:
- Return an array of names, total game number, win number, loss number ,and ranking of teams with 8 highest scores in each season and which season they played in.

#9
Route: /players

Description: Returns an array of stats of all players in the database
Route Parameter(s): None
Query Parameter(s): None
Route Handler: all_players(req, res)

Return Type: JSON
Return Parameters: { (JSON array of { { Name (string) ,Team (string), Season (int) , AVG_PTS (int), AVG_AST(int), AVG_REB(int)}) }
Expected (Output) Behavior: Return all players with all the above return parameters

#10
Route: /game

Description: Returns an array of stats of a single game in the database
Route Parameter(s): None
Query Parameter(s): id (int)
Route Handler: game(req, res)
Return Type: JSON
Return Parameters: { (JSON array of { GameId (int), game_date_est (datetime), t.NICKNAME (string), PTS_home (int), p.NICKNAME (string), PTS_away (int), AST_home (int), REB_home (int), AST_away (int), REB_away (int), FG_PCT_home (float), FT_PCT_home (float), FG3_PCT_home (float), FG_PCT_away (float), FT_PCT_away (float), FG3_PCT_away (float) }) }
Expected (Output) Behavior: Return information for a specific game detail

#11
Route: /teams_conference

Description: Returns an array of teams details based on the conference (east, west, all)
Route Parameter(s): None
Query Parameter(s): conference (string), season (int)
Route Handler: teams_conference(req, res)
Return Type: JSON
Return Parameters: { (JSON array of { team_id (int), nickname (string), AVG(g.PTS_home) (float), AVG(p.PTS_away) (float)}) }
Expected (Output) Behavior: Return information for team stats in 2021 based on conference the teams belong in