Stage 5: Query Design

Group 24 – Aaron Herrera, Jerome De Guzman, Gurwinder Khandal

Note: This is based on our normalized tables

1. Output the roster for a team in a particular season.
   - select firstname, lastname, jersey
     from player natural join play natural join team
     where seasonID = season and teamName = tName;
   - Variables:
     - season refers to the specific season that the user will input (e.g. "2019/2020")
     - tName is the name of the team the user wants to see the roster of (e.g. "Lakers" or "Chicago Bulls")
       - these variables will most likely use wildcards (i.e. "%" so that partial names/season years can be still be related back to a team or season)
   - The results would be interesting to an analyst because it would allow them to see/compare the different rosters a team has had over the years. This is also basic information that can be utilized for input or as a motivator for other queries of interest.
2. Get game appearance percentage for each player for each season (i.e. number of games played/number of games their team played. This allows players who didn't make the playoffs to still be ranked highly even though they played less total games than playoff teams) and rank them. The user can input the top x players (e.g. top 10, top 25, etc.)
   - with gamesPlayedPlayer as (
     select playerID, firstname, lastname, seasonID, count(gps.gameID)
     as playerGP
     from players natural join gamePlayerStats gps left join regularGame rg
     on gps.gameID = rg.gameID
     left join playoffGame pg on gps.gameID = pg.gameID
     group by playerID, firstname, lastname, seasonID),
     gamesPlayedTeam as (
     select teamName, seasonID, count(gts.gameID) as teamGP
     from gameTeamStats gts left join regularGame rg on gts.gameID = rg.gameID
     left join playoffGame pg on gts.gameID = pg.gameID

group by teamName, seasonID)
select seasonID, teamName, firstname, lastname, 100.0 * gpp.playerGP/gpt.teamGP as appearancePercentage
from gamesPlayedPlayer gpp join play on gpp.seasonID = play.seasonID
join gamesPlayedTeam gpt on play.teamName = gpt.teamName order by appearancePercentage desc limit ?;

- This can show analysts who the most reliable players are in terms of being available for their team which can play a role in discussions revolving around player rankings.
- variables:
  - ? is the number players the user wants to see (e.g. "10" represents Top 10)

3. Show all the teammates of a particular player (for the seasons our database covers)
   - with seasonTeamsPlayed as (
     select seasonID, teamName
     from play natural join player where firstname = xfirst and lastname = xlast and teamName = xTeam),
     xroster as (
     select playerID, firstname, lastname
     from players natural join play where (seasonID, teamName) in seasonTeamsPlayed
)
     select playerID, firstname, lastname
     from players where (playerID, firstname, lastname) in xroster
     and playerID not in (
     select playerID from players natural join play where firstname = xfirst and lastname = xlast and teamName = xTeam); -- player X is not his own teammate!

   - This would be interesting to an analyst as it can show the different people and level of talent that certain players have had as teammates throughout their career. This knowledge can also fuel other interesting queries/questions (e.g. if player X played with player Y and player Y was a really good player, how successful was their season when they played together?). This knowledge can also be useful in debates when analyzing the supporting cast (i.e. teammates) specific players have had or comparing the performance of a player among different teammates.

- variables
    - xfirst is the player of interest's first name (e.g. "Lebron") which the user will input
    - xlast is the player of interest's last name (e.g. "James") which the user will input
    - xTeam is one of the teams that player X has played for (which the user will most likely know if they know the player). It acts as an extra identifier for the specific player since some players might have the same name but are unlikely to play on the same team

4. Rank the officials who have officiated the most games.
    - select officialID, firstname, lastname, count(gameID) as numGames from officials natural join officiate group by officialID, firstname, lastname order by numGames desc;
    - This would be interesting for an analyst working for the NBA as it can help determine who the most experienced NBA officials/referees are. With this knowledge, it can help determine which referees to officiate the most important games such as games in the playoffs or in the finals where this is more pressure to make correct calls.
5. Show the all the teams and the arenas they play in and the capacity of that arena
    - select teamName, capacity from team natural join stadium order by teamName asc;
    - This basic query can be used as a means to gain information as input for other queries. In addition, it can help answer any inquiries regarding the home arena of certain teams.
6. Show the total number of players from each country in a particular season
    - select country, count(playerID) from play natural join player natural join location where seasonID = ? group by country order by country asc;
    - This would be interesting to an analyst since they can see how the nation representation in the NBA is distributed across seasons. This can potentially show trends in how talent in the NBA shifts across years (e.g. there's been more discussion recently on how international representation is higher/stronger in the NBA today, as opposed to the greater prevalence of American players in previous seasons)
    - variables:
        - ? = the specific season the user wants (e.g. "2019/2020") that will be taken as user input

7. Rank coaches with highest win percentage in the regular season over the span of all seasons our database covers. The user can input how many coaches they want to see.
    - with coachGamesWon as (
      select coachID, firstname, lastname, count(gameID) as gamesWon
      from regularGame natural join gameTeamStats natural join
      gameTeamInfo natural join team natural join manage natural join
      coach
      where winner = teamName
      group by coachID, firstname, lastname),
      coachGamesPlayed as (
      select coachID, firstname, lastname, count(gameID) as gamesPlayed
      from regularGame natural join gameTeamStats natural join team
      natural join manage natural join coach
      group by coachID, firstname, lastname
)
      select c.coachID, c.firstname, c.lastname, 100.0 *
      cgw.gamesWon/cgp.gamesPlayed as winPercentage
      from coach c join coachGamesWon cgw on c.coachID = cgw.coachID join
      coachGamesPlayed cgp on c.coachID = cgp.coachID order by
      winPercentage desc;
    - This would be interesting to an analyst because it can shed light on who some of the most successful (or least successful) coaches are during the regular season. This data can influence which coaches teams may consider when trying to hire a new coach.
    - variables :
        - ? is the number of coaches the user wants to see (e.g. “10” represents top 10)
8. League leaders in major stat categories based on averages for players in a particular season:
    - This is a fundamental query that casual and expert fans will appreciate. Using this query, users will be able to see which players lead the league in major stat categories for a particular season. This is of interest to analysts as it is crucial in individual player awards at the end of the season. This query can also be expanded beyond just one season. The variables here will be type of stat (points, rebounds, steals etc), season and an integer which sets the limit for what is presented to the user.
    - SQL:

        i. SELECT Player.playerName, AVG(GamePlayerStats.stat?) as avg FROM GamePlayerStats JOIN Player on GamePlayerStats.playerID = Player.playerID JOIN Game ON GamePlayerStats.gameID = Game.gameID WHERE Game.seasonYear = S? GROUP BY Player.playerName ORDER BY avg DESC LIMIT L?;
    - Some remarks about the SQL:
        i. stat? represents the variable for type of stat, S? represents the variable for season, and L? represents the variable for limit number.
9. Compare a players' regular season career averages against their playoff career averages:
    - This is a particularly simple yet interesting query which is often brought up when fans debate if a player is a playoff riser or not. Jimmy Butler for example, his regular season career averages are lower than his playoff career averages. This means he is a playoff riser. This argument is often used in the GOAT (Greatest of All Time) debate as well. The variables here will be playerID, and the selected statistical category.
    - SQL:
        i. SELECT Player.playerName, AVG(GamePlayerStats.stat?) as avg FROM Player join GamePlayerStats ON Player.playerID = GamePlayerStats.playerID JOIN RegularGame on GamePlayerStats.gameID = RegularGame.gameID GROUP BY playerID
        UNION
        SELECT Player.playerName, AVG(GamePlayerStats.stat?) as avg FROM Player join GamePlayerStats ON Player.playerID = GamePlayerStats.playerID JOIN PlayoffGame on GamePlayerStats.gameID = RegularGame.gameID GROUP BY playerID
    - Some remarks about the SQL:
        i. Stat? represents the variable for type of stat user wishes to compare. The other variable is playerID.

10. Major stat averages for a player for all the teams he played for in his career:
    - This query would allow users compare how well a player played on several different teams throughout their career. For example, Lebron's averages on the Cavaliers versus the Lakers. We can also edit this query to compare based on seasons to see what year a player was peak for a particular team. This is of interest to analysts and casual fans because it

can be used to assess the players' surroundings and how they might have an impact on the player's individual performance. The variables here are playerID.

- SQL:
  i. SELECT Player.playerName, Play.teamName, AVG(GamePlayerStats.points), AVG(GamePlayerStats.rebounds), AVG(GamePlayerStats.assists) FROM Player JOIN Play on Player.playerID = Play.playerID JOIN GamePlayerStats on Player.playerID = GamePlayerStats.playerID JOIN Team on Play.teamName = Team.teamName GROUP BY Player.playerName, Play.teamName ORDER BY Player.playerName, Play.teamName;
- Some remarks about the SQL:
  i. Based on the feedback, we have edited this query to show the stats for all the teams a player has played for rather than just one team.

11. List all the champions in chronological order:
    - This is a simple query that can be used to check which team won the championship during a particular season.
    - SQL:
      i. SELECT seasonYear, champion as teamName FROM Season ORDER BY seasonYear;
12. Highest player efficiency rating (PER) on a single team, in a single season.
    - This query will calculate the PER statistic by the formula:
      i. Positive – negative statistics
    - This can be useful for understanding how efficient players on a team can be while on the court. Spanning this out over the course of a single season for a single given team, we can see how players rating can average out for their team. Enabling the ability to discern the players with the highest PER on a team.
    - We can use this statistic to track how well a team has performed for the given season year.
      i. E.g. Top 10 Highest PER on Lakers from 2023-2024 and compare with Top 10 Highest PER on Lakers from 2024-2025 season.
      ii. The query will contain 2 variables, the team (singular) - xTeamName and the season year (singular) - xSeasonYear
    - SQL:
      i. SELECT p.playerID, p.playerName,
      ii. AVG((gps.points + gps.rebounds + gps.assists + gps.steals + gps.blocks) - (gps.turnovers + gps.fouls + gps.missedFG)) AS avgPER
      iii. FROM Play szn
      iv. JOIN Player p ON szn.playerID = p.playerID
      v. JOIN GamePlayerStats gps ON gps.playerID = szn.playerID

vi. LEFT JOIN RegularGame rg ON rg.gameID = gps.gameID
vii. LEFT JOIN PlayoffGame pg ON pg.gameID = gps.gameID
viii. WHERE szn.teamName = xTeamName
ix. AND (rg.seasonYear = xSeasonYear OR pg.seasonYear = xSeasonYear)
x. GROUP BY p.playerID, p.playerName
xi. ORDER BY avgPER DESC
xii. LIMIT 10;

- Some remarks about the SQL:
  i. Corrected the abiguity by clarifying we will be showing the result of the PER for the top 10 highest PER for a given team.

13. Given a specific draft round of players, measure their performance through multiple seasons and teams.
    - This query is fixed on the group of players that were drafted for a given year and for a given draft round, and measures their performance as they grow from their rookie season to later in their career.
    - This query search would be valuable to analysts because this can help to evaluate the performance of specific draft rounds.
      i. E.g. Evaluate if a class of later draft round picks could outperform the earlier draft round class.
    - The query will contain 2 variables, the draft year – xDraftYear and the draft round – xDraftRound
    - SQL:
      i. WITH DraftYrRound AS(
      ii. SELECT di.draftYear, di.round, di.pick, di.playerID
      iii. FROM DraftInfo di
      iv. WHERE di.round = xDraftRound AND di.draftYear = xDraftYear)
      v. SELECT dyr.draftYear, dyr.round, p.playerName, dyr.pick, szn.seasonID,
      vi. AVG((gps.points + gps.rebounds + gps.assists + gps.steals + gps.blocks) - (gps.turnovers + gps.fouls + gps.missedFG)) AS avgPER
      vii. FROM DraftYrRound dyr
      viii. JOIN play szn ON szn.playerID = dyr.playerID
      ix. JOIN player p ON p.playerID = szn.playerID
      x. JOIN GamePlayerStats gps ON gps.playerID = p.playerID
      xi. LEFT JOIN RegularGame rg ON rg.gameID = gps.gameID
      xii. LEFT JOIN PlayoffGame pg ON pg.gameID = gps.gameID
      xiii. WHERE szn.seasonID >= dyr.draftYear
      xiv. GROUP BY dyr.draftYear, dyr.round, p.playerName, dyr.pick, szn.seasonID

xv. ORDER BY p.playerName, szn.seasonID;

- Some remarks about the SQL:
  i. From the feedback given, it was suggested to form thelist of players in a view or CTE, for readability.
  ii. We have adhered to the feedback by implementing a CTE for readability.

14. Get the total “statistic” metric by a player across all games (all time)
    - This is useful for tracking how many of a certain statistic metric a player has so far.
      i. The statistic could be points, assists, rebounds, etc.
    - An analyst could use the results of this query to compare specific statistical metrics between two or more players.
    - The output of finding the total of a specific statistical metric is also a quite popular lookup.
      i. E.g. NBA games broadcast these metrics for a player on screen when there is something significant happening.
    - This query contains one variable, the statistic that will be tracked as xStatistic, and relabel output as totalxStatistic
    - SQL:
      i. SELECT p.playerName, SUM(gps.xStatistic) AS totalxStatistic
      ii. FROM Player p
      iii. JOIN GamePlayerStats gps ON p.playerID = gps.playerID
      iv. GROUP BY p.playerName;