**Group Number:** Group 8

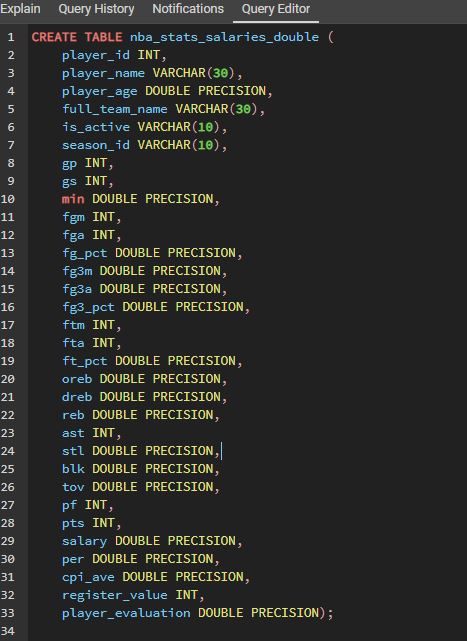
**Data:** 10/27/2020

**Project Topic:** NBA Player Statistics/ Salaries Comparison

**Procedure:** ETL focused project:

**Date Sources:**

* The first source used was an NBA API that can be called through a pip install where it is a built-in programmed API that contains information including common NBA players info, players stats, team info and stats, and player career stats across all seasons. The reason behind using this source is the wide availability of the data across all players and their history.
* The second source was current consumer price where we obtained the inflation data (Consumer Price Index) to compare player’s salaries across all years back to 1913 to 2020 but we had to narrow this down based on each player's season history. Inflation data was obtained because the value of the dollar changes from month to month and year to year.
* The final source was obtaining all player’s salaries by year, this was obtained from a CSV containing NBA player salaries found on Kaggle. Data was obtained to compare player efficiency to salary in order to determine whether the player is overpaid or underpaid
* **Extract-Transform:** 
  + The source was limited only to data requested by player id so it was necessary to create a list of all player’s ids to loop through the provided API and request all data for each player across all seasons since career start.
  + The API limited us to 30 reads per request so the team performed a try/except process to notify us when the code breaks to run it again. Luckily we had to do this twice including a short break between each run, and ended up with over 26,000 rows of data for about 4500 NBA players.
  + The data was pulled and pushed to a dataframe then merged with a framed player info table that included the full name of each player. The newly created data frame included all players info along with career stats.
  + The merged data frame was then pushed to a CSV. The next step was exporting each player's efficiency to analyze each player separately comparing salaries based on efficiency of each player. The salaries table was then merged along with the player stats table including all efficiencies.
  + Web scraping was applied to obtain the salaries data and inflation data from each provided URL, once the data was obtained it was transformed to html tables and then converted to dataframes to be merged with the other tables mentioned above.
  + The data frames were cleaned and formatted using the correct data type for each column. Cleaning the data also included splitting the SEASON\_ID column in order to be able to merge both player stats data frame and salaries data frame.
  + Overall process required us to do data munging to all gathered data from different sources before loading it into our database. Extracting included sources mainly from urls and csvs. Transforming including data cleaning, summarization, selection, filtering and joining the data framers
* **Loading:**
  + We created a Postgres schema for creating a table with the necessary columns. We chose to use postgres because we were not using dynamic data, and therefore the database would not need regular updates.
  + Then we made some final modifications to the player stats data, removing unnecessary columns and fixing some column names and values to not include spaces and to be more compatible with SQL.
  + We then created a connection to the database in the jupyter notebook and inserted all the values from our player stats dataframe into the database table.
  + Next we created a python flask application to view data, which allows the user to enter the name of an NBA player and a year, and the application will present the players stats, salary and over-underpay estimation in JSON format.
* **Client Use:**
  + Clients can route through the provided url that was designed through the built flask to obtain each player's info. He would enter the player’s name within each year and our url will provide access to player stats and salaries. The accessible data is unique because it includes a player’s salary as well as their stats for a given year, as well as the calculated PER and player pay evaluation.
* **Schema:**



**Player efficiency and salary Data Normalization**

The purpose of this study was to verify if player efficiency could be compared to a player's salary and to do a simple calculation to estimate if a player is overpaid or underpaid. NBA players efficiency was calculated by using the Hollinger NBA Player formula for player efficiency rating (PER). PER formula measures a player's per-minute performance, while adjusting for team pace. PER considers accomplishments, such as field goals, free throws, 3-pointers, assists, offensive rebounds, defensive rebounds, blocks, and steals, and subtract negative results, such as missed shots, turnovers, and personal fouls. However, for the purpose of this project PER was not adjusted for pace (non-adjusted PER) as the team pace data was not found in any data set available.

PER=((FGM\*85.9)+(steal\*53.8)+(3PTM\*51.7)+(FTM\*46.8)+(blocks\*39.1)+(OREB\*39.0)+(assists\*34.6)+(DREB\*14.7)-(fouls\*17.7)-(FT\_Miss\*20.0)-(FG\_miss\*39.1)-(TOV\*53.89))\*(1/minutes)

Players with a PER higher than 25 are considered as all-stars players, players with a PER of ranging from 20 to 25 are considered borderline all-stars players, PER of 12 to 20 as average players and players with a PER below 12 are considered as rotation players.

To estimate if a player was overpaid or underpaid (player evaluation) according to their respective PER, the salary and PER of each player was normalized to the salary of an average player today ($15,000,000) using the Consumer Price Index (CPI-U) for specific year and a PER of 15 (average player). The player evaluation data indicate if a player was overpaid or underpaid. Players with a value below zero indicate underpaid and players with a value higher than zero indicate overpaid. Values closest to zero is an indication that the salary of the player is the right one according to their PER. However, it is noteworthy to know that this estimation can underestimate or overestimate the value of a player as many important data was not taken into consideration such as defensive skills, salary cap, commercial value of the player, and experience.