NBA Players Salary and Stats

For the project of Extract Transform and Load (ETL), the topic that our group decided to extract data was for NBA players’ salaries, stats and basic information for the 2018-2019 season. In the database we wanted to provide the ability to compare statistics, player dynamics and salaries. During the process of providing all this information we had to import CSV files, rename multiple column headers and merge the three different pandas dataframes into one dataframe. After we completed this set-up, we then had to upload it into PostgreSQL. We need to create database and tables in PostgreSQL for it to then be imported.

Our data sources for this extraction came from Kaggle which gave us player stats from the 1950’s all the way through the 2018-2019 NBA season for all players. The other source we used was basketball-reference.com and this gave the information for the contracts of the 2018-2024 seasons. For the information of the stats it gave information on everything from points scored in a game to how many minutes they played each game… etc. For the contracts it showed us how much each player is set to make in 2019, 2020, 2021, 2022, 2023 and 2024 and the total amount to be made over that time frame.  Both of the data came from their respective websites and then downloaded as CSV files and imported into pandas as CSV files.

The type of transformations that was performed on the data was first reading in the CSV files into pandas dataframe and naming the salary information “Player\_Salary\_df”. Then bringing in the CSV file of the player’s basic information which was named “Player\_Basic\_df”. The last one that was brought in was the player stats from the 2018-2019 season and this was accordingly named “Player\_Stats\_19\_df”.

The first thing that took place was renaming the column header in “Player\_Basic\_df” from “Name” to “Player”. Later all of our data will be joined on the column header “Player”. Next, we found out what columns were in “Player Basic”. In “Player\_Basic” is a column with the “Year End” or the year a player retired from the NBA. We decided to drop this column because the stats we are looking for is only for the 2018-2019 season. After this the next thing was to rename “PLAYER” to “Player” in “Player\_Stats\_19\_df” to make sure when things are merged, they all have the same column heading. Also, in “Player\_Stats\_19\_df” we had a column that was called “0.625” which was for 3-point field goals made. This column was renamed to “3pm” to make it easier for the reader to understand what stat is being displayed. The other thing that was changed was dropping the columns “DD2” and “TD3”. We didn’t find any relevance for these two categories of stats which was the purpose of dropping them. After all these changes were made, we finally had everything ready to be merged. The first thing that was merged was “Player\_Basic\_df” with “Player\_Salary\_df” and it was merged on “Player” and was called “df”. Once these two were merged we then merged “Player\_Stats\_19\_df” with “df” on “Player” again and named the merged files “NBA\_19”. Once all these mergers took place there were 6 columns that contained the year of operation (“2019-20) and renamed to include the word “Salary” (ex: Salary\_2019To2020).

We then created a connection to the database to Postgres and created an engine to connect to Postgres. Then we went into PgAdmin and created a new database along with a table based on the column headers from the Juypter file. When we attempted to insert the data from Pandas into the database, we got error messages. The issue that arose from utilizing Pandas to insert the data into the PostgreSQL was that the column headers didn’t match.

The column headers were the culprit due to the first letter in each header being upper case in Pandas. However, as we found out, Pandas is case sensitive but Sql is not. Thus, we had to make all the column headers lowercase so they would match the Sql file. What we did to rectify this issue was we wrote a “for loop” to turn all the column headers to lowercase so they would match the Sql database. Another thing we had to do was change the “rk” which was the rank into the primary key. As we found out the rank was a unique value for each player.

After the column issues were remedied, we were able to insert the data into the database through the connection. And we checked our work by running a query in PgAdmin and in Pandas.