**Final Report: ETL Project**

Project Members: Max Patterson, Anastasios Tomazos, Erin Wills

Abstract

The objective of this project is to initially *extract* National Football League (“NFL”) data from at least two different sources, then *transform* such data into a comprehensible and usable format, followed by *loading* the data into a central database. Although there seems to be numerous sources for NFL data, the dearth of free and publicly accessible data sources is a major stumbling block for the interested fanbase, like us, in performing significant football analytics. Be that as it may, we found NFL data from two different sources with which we created a useful database composed of two tables comprising player and gameday data for the 2009 – 2013 seasons.

Breakdown of Tasks

The overall ETL process is comprised of *Extract*, *Transform*, and *Load* subprocesses.

*Extract*

Our data sources are extracted from the **nflscrapR-data** repository and the **NFLsavant.com** website. **nflscrapR-data** was created by a group of [Carnegie Mellon University statistical researchers](http://www.stat.cmu.edu/) who released their data to the public**,** which is basically an R package that “uses an API maintained by the NFL to scrape, clean, parse, and output clean datasets at the individual play, player, game, and season levels.”[[1]](#footnote-1) **NFLsavant.com** is a website that obtains publicly-available NFL play-by-play data on the internet.

Our plan is creating the following database in PostgreSQL with the following two tables:

* Player data from the **nflscrapR-data** and append to this dataset the NFLsavant.com player information to include the college of each player
* Game data from **nflscrapR-data** and append to this dataset the NFLsavant.com weather data for each game

The data we are interested in are formatted in CSV files. We limited our project to 2009 to 2013 NFL data because this is the period both datasets overlap. In accordance with our plan, we set out to use the following 12 CSV files:

* **nflscrapR-data:** 
  + NFL gameday information for 2009, 2010, 2011, 2012 and 2013 (5 CSV files)
  + NFL player rosters for 2009, 2010, 2011, 2012 and 2013 (5 CSV files)
* **NFLsavant.com**:
  + All NFL Weather from 1960 to 2013 (1 CSV file)
  + Every NFL player information from 1920 (1 CSV file)

*Transform*

We merged the player roster data from **nflscrapR-data** and the NFLsavant.com player information to create a pandas dataframe that includes basic ***player*** information with the college where they played football. We also merged the ***gameday*** data from **nflscrapR-data** and the **NFLsavant.com** weather data for each game to include weather information (i.e., temperature, wind chill, humidity, wind speed, weather summary) for each home game from 2009-2013. After we retrieved the data, out initial step was to read the twelve CSV files directly through the websites (html) into separate jupyter notebooks.

***Player dataframe***

Initially, we thought it would be advisable to filter the **NFLsavant.com** player information dataset to players that began their NFL career no earlier than a certain year (we chose, 1985) in order to limit the amount of duplicate player names. However, when we joined the two player datasets using pandas, we realized that the “start year” column in the **NFLsavant.com** dataset itself was clearly inaccurate as it included players that are currently playing in the NFL with “start year” dates in the previous decades. So, we proceeded to eliminate the 1985 filter. In order to further transform the pandas dataframe so as to exclude redundant player information for the 5 years (e.g., Aaron Rodgers – Aaron – Rodgers - QB – California – GB (\* 5 season – 2009, 2010, 2011, 2012, 2013)), we deleted the “Season” column in our dataframe and then used the drop\_duplicates method.

Despite this cleaning, the player dataframe continued to duplicate school/team for some players with identical names who played during 2009-2013 (e.g., Adrian Peterson). Undaunted, we forged ahead by creating a blank list in order to see which indexes are duplicates, then created a temporary dataframe where we removed the “College” column so that we can see true duplicates (i.e., whether the first index of each player is correct). Further, we removed the duplicates from the original player merged dataframe and reset the index. Despite our relentless transformation efforts, we nevertheless found about ten duplicate records of players with identical names (e.g., Adrian Peterson) in the dataframe consisting of 1,380 rows of players. We came to the conclusion that we had cleaned and transformed the data as best we could given the flaws in the original data source and that these duplicates were outliers that would not affect the usefulness or integrity of our database.

***Gameday dataframe***

Initially, we needed to clean both tables before merging them into a pandas dataframe because each table labelled home teams differently. The process for doing this required formatting an existing ID string in both tables and merging the team names (e.g., Arizona Cardinals) with team abbreviation (e.g., ARI). These two values, when merged into a dataframe, created a unique ID enabling a simple merger of the two tables. Because there are only unique values in the two tables, the dataframe only includes relevant gameday data and weather data, eliminating redundant columns and thereby making the table more presentable.

*Load*

We loaded the two dataframes into a PostgreSQL relational database. First, we imported sqlalchemy to create an engine in jupyter notebook in order to connect to Postgres. Secondly, we retrieved the data for both pandas dataframes in order to push them into SQL tables. Thirdly, we inserted each dataframe to SQL and confirmed that the data had been added by querying the two tables.

1. See, https://www.kaggle.com/maxhorowitz/nflplaybyplay2009to2016 [↑](#footnote-ref-1)