

**ETL Project**

**Connor Vergara**

**Damita Zweiback**

**March 5, 2019**

Contents

[1 Introduction 2](#_Toc2703213)

[2 Extract 3](#_Toc2703214)

[3 Transform 5](#_Toc2703215)

[4 Load 7](#_Toc2703217)

1. Introduction

The purpose of this project was to extract, transform, and load data related to NFL teams. Particularly, we were interested to see if we could draw relationships between teams that were more popular (based on subreddit following) and other data points such as the team’s value, the population of the team’s city, and how long the team has been in existence.

This document will walk you through the “ETL Project - NFL Stats.ipynb” file on our GitHub repository.

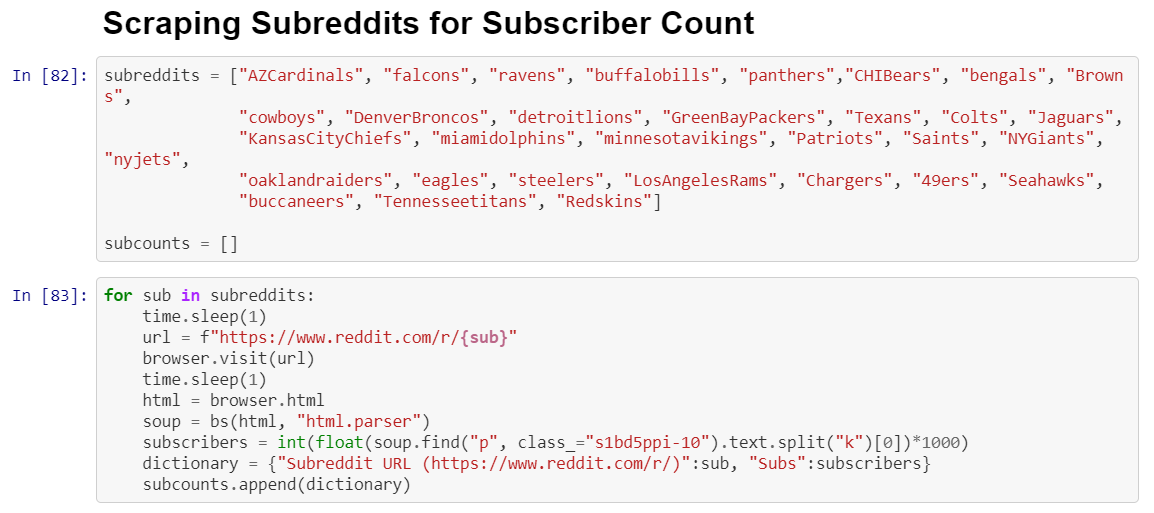
1. Extract

Nearly all of our data for this project was scraped from different websites. We began by creating a local Excel file that contained the team name, the team’s subreddit URL, the team’s stadium city, and the team’s “real city”. It’s important to note that we made some assumptions during this step. For example, even though the New England Patriots play their home games in Foxborough, MA, we thought it would be more appropriate to use Boston, MA as the “real city”. The original Excel file can be found embedded below:

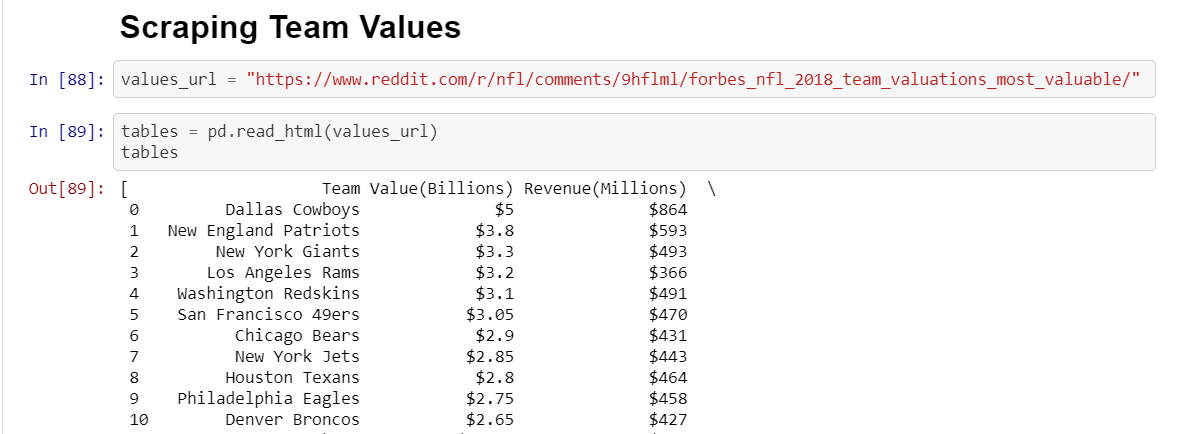
****

The first step in our extract process was reading this local CSV as a Pandas dataframe. This dataframe ultimately served as the skeleton for our final dataframe. As we scraped for more and more information, we added to this dataframe.

The second step in our extract process was finding the subreddit subscriber counts for each team. This was accomplished by using splinter to visit each subreddit and pulling the subscription data from each page. We created a dictionary to store the subreddit URL and the subscriber count, and used a for loop to append that dictionary to an empty list. Below is a screenshot of the code that was used to perform the scraping:



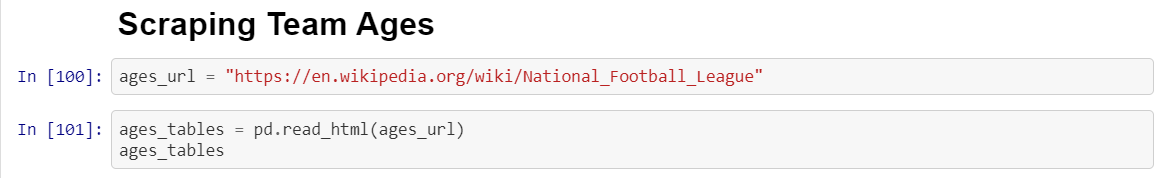
The next step in the extract process was scraping the web for the team values. We found a [Reddit post](https://www.reddit.com/r/nfl/comments/9hflml/forbes_nfl_2018_team_valuations_most_valuable/) that included a simple table of Forbes’ 2018 NFL team evaluations. To ingest this information, we used the Pandas read\_html method and passed the URL of the post. Below is the code used for the final scraping:



The next step involved scraping for city populations. After transforming some of the “Real City” data (see the Transform section for more details), we collected 2016 population data for each team’s city using [city-data.com](http://www.city-data.com/). This was accomplished again through the use of splinter, html parsing, and storing the values in a dictionary and appending the dictionary to an empty list. Below is the code used to scrape the city population data:

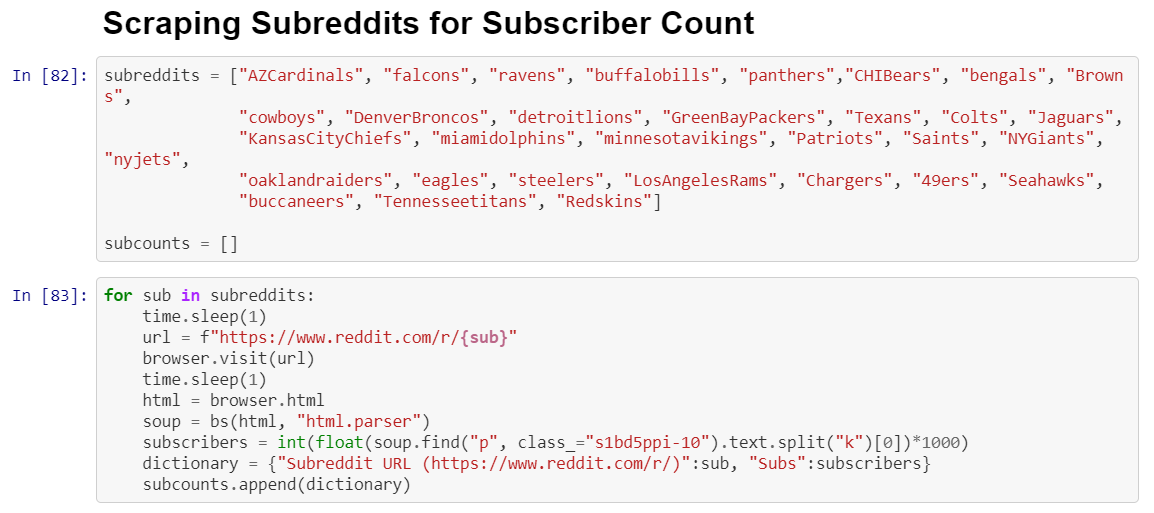


The final step of the extract process involved scraping a [Wikipedia site](https://en.wikipedia.org/wiki/National_Football_League) to determine each team’s founding date. Again, we used the Pandas read\_html method and passed the URL of the Wikipedia article. We then specified the index to return the table that we were looking for. Below is the code used to extract the table from this site:

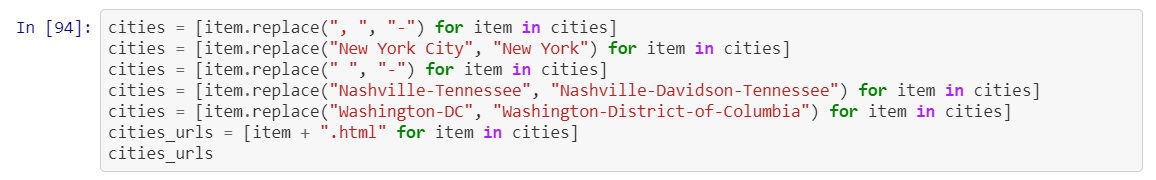


# 3 Transform

Throughout the extract phase, we needed to transform our data at different phases. The first transformation needed was converting the subscriber information from a string into an integer. We wanted to store the information as an integer so that it could be used in plotting or further analysis in the future. Highlighted below is the line of code that was used to accomplish this:



Another important data transformation involved taking the “Real City” from the local Teams CSV and changing some of the characters and values so that we could successfully use splinter to pull the population data. Particularly, spaces in cities were replaced with hyphens and certain cities were replaced with alternate names. Below is the code used to manipulate the city data into a usable format:



We also wanted to store the information as an integer, so we used a very similar approach as the subscriber information. Highlighted below is the line of code that was used to accomplish this:

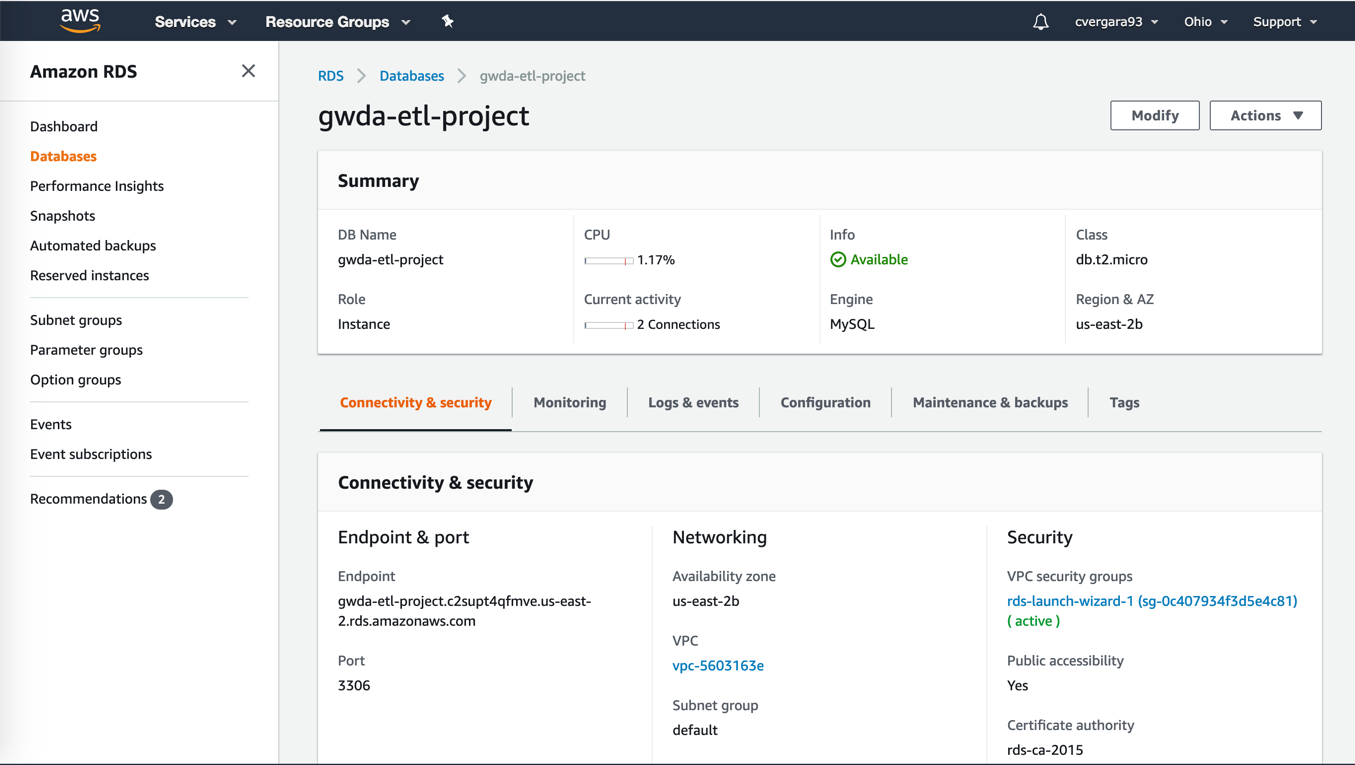


Another significant transformation was needed when calculating a team’s age. First, we needed to convert one of the columns from our read\_html dataframe into a list. Once we had the values stored in the list, we needed to take only the first four characters of each element and then convert it into an integer. Another tricky thing here was that some of the “Teams” from the table had a few characters at the end such as \* that we wanted to remove so that we could properly merge our data. We also had to drop some undesirable row entries. Lastly, we used the datetime library to calculate a team’s age based on the current year. The code below represents the steps described above:



# 4 Load

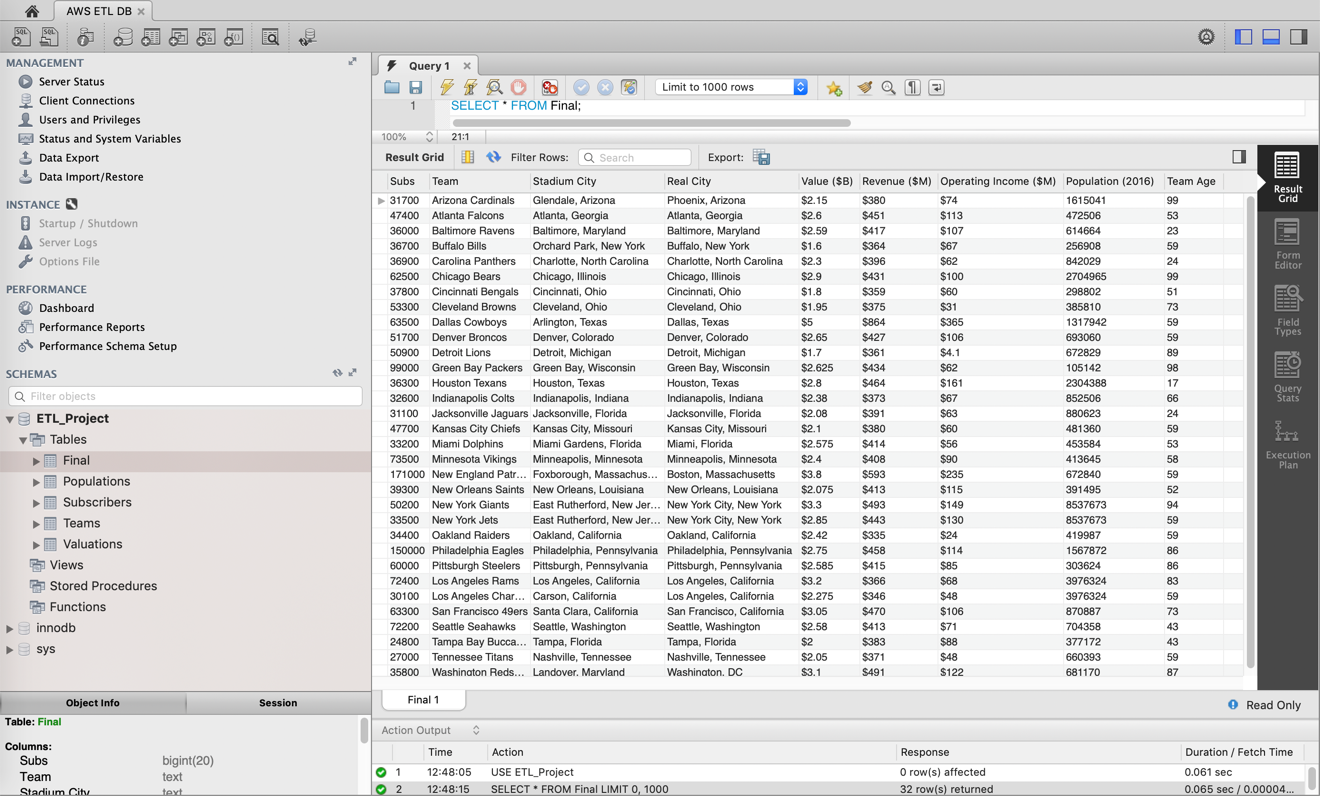
The final step in our ETL process was taking our data and storing it into an AWS RDS database. We began by creating a new MySQL database and adjusting the inbound security settings to allow any machine to access it. Below is a screenshot of the AWS database details:



To get our data into this dataframe, we used sqlalchemy to establish a connection in our Jupyter notebook. We stored our AWS credentials in a config.py file and pulled in the required fields to establish the connection. Then, we read in our various CSV files that we generated throughout the extract/transform phases as separate dataframes. We created five different tables within our database – Teams, Subscribers, Valuations, Populations, and Final. Below is the code that was used to establish the connection, declare the final dataframes, and load them into the various tables listed above:



To confirm that our data was successfully stored into the AWS database, we created a connection in MySQL Workbench with the database. We then ran a simple SELECT \* FROM query to see the contents of our “Final” table within the database. Below is the result of the query:



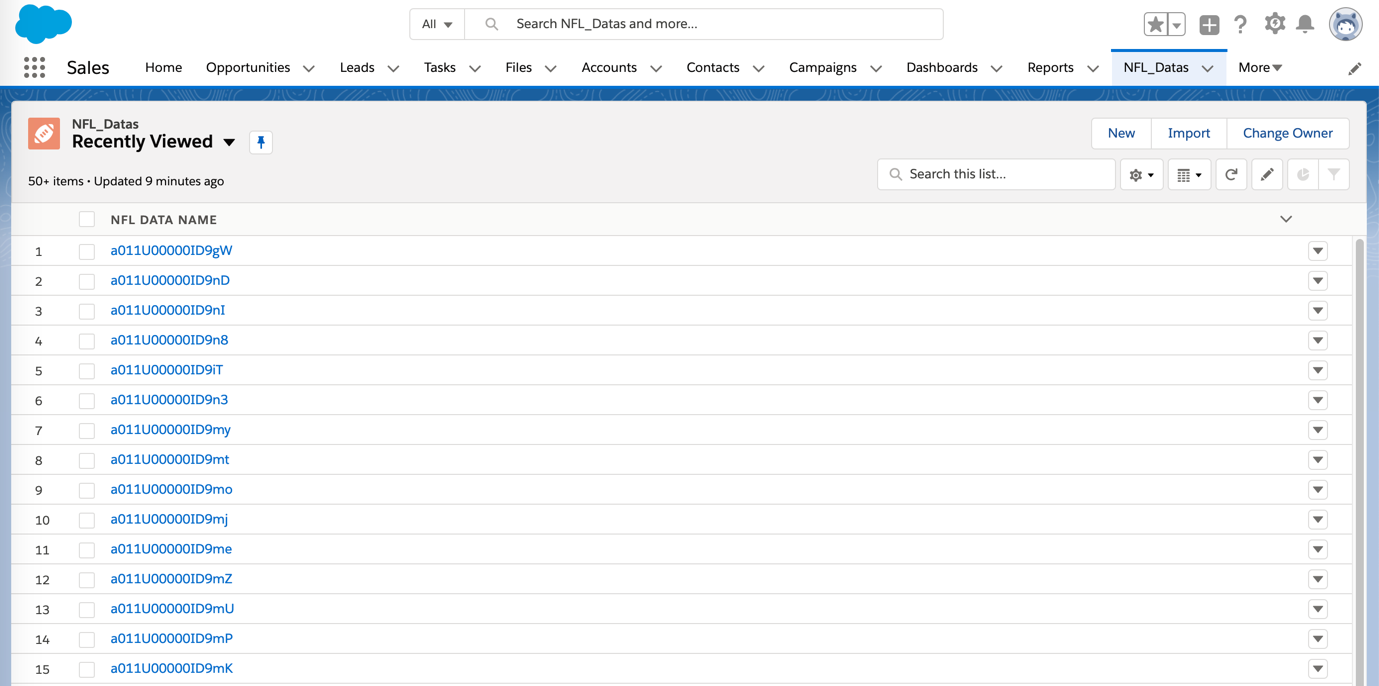
We were also able to store our data on Salesforce (with some last minute help from Dartanion to show us how to view records). To begin, we created a new custom object called “NFL\_Data” and built out the various fields based on the columns in our final dataframe. Again, we pulled in our credentials from a config.py file. Below is the code that establishes the Salesforce connection:



To send our records to the Salesforce database, we first connected to our AWS database, ran a simple SELECT \* FROM Final query, stored the results in a Pandas dataframe, and finally used the iterrows function to pass each record. Below is the code to perform that:



To confirm our results, we logged in to Salesforce and checked our application. Below is an image that shows the records that were stored in the database:



Below is a screenshot of an individual record that includes all of the data points that we scraped during the extract/transform phases:

