ETL – NFL QB STATS

Extract - CSV (Pro Football Reference - stats) | HTML (USA Today rankings)

Transform – Web scrape → Cleaned up CSV → Jupyter Notebook → List of python dictionaries

Load – Load list of python dictionaries into Mongo DB using 'import pymongo' in Jupyter Notebook

Data Cleanup & Analysis:

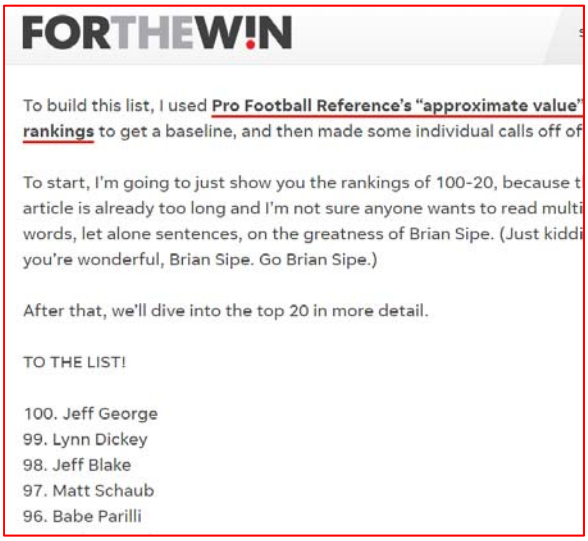
Data Sources –

Pro Football Reference

We used csv files exported from the Pro Football Reference website. You can use the search tool below to generate the quarterback stat charts. To get the tables, you select a stat to analyze, specify career or season stats, choose all available data to date (2019), then click the green "Go!" button. Once the table is generated you can click the 'Share & more' toolbar to export the chart to a csv file that we can manipulate.

USA Today

In order to get our list of all-time quarterback rankings we used BeautifulSoup to scrape the HTML code from the USA Today sports blog 'FOR THE WIN' to get a list of names to then compile data for. We needed their list in order to have a name to perform lookup functions in excel (index/match) and retrieve stats. We also needed their ranking data associated with the quarterback names.



Type of Transformation:

For Transformation we took our csv exports from Pro Football Reference and dumped them into one master Excel file (12 files total – CSV Excel tables). In the master Excel file, we used built-in Excel functions such as “text-to-columns”, “find and replace”, “concatenate”, and “index(match())” to clean up the data and generate a final table that we could read in Jupyter notebook to further our transformation.

For example, in the web scrape from the USA Today article, ranking and quarterback name were combined in one string/column, so we separated them using space as the delimiter. We then put them in their own separate column and removed the period from the ranking string (using “find and replace”). Finally, we concatenated the first and last name back together into one column.

We did this so that the name formatting would match that of the csv tables. That way when we pulled all the statistical data across all the csv tables, the name match would allow us to retrieve the statistical data we needed. We used the index/match function in excel to add statistical data from the csv exports to our master list generated from the web scrape. We then copy and pasted that data into a separate csv file so that the file could be easily read using Python and Jupyter Notebook.

Once we appended each excel column to a python list in Jupyter notebook we used a for loop to convert a zipped list of tuples into a list of python dictionaries with key value pairs (‘stat type’ : ‘data’). We then imported our list of dictionaries into MongoDB so that you could query relevant quarterback data using their full name. We chose a non-relational database to store our data.

The final tables or collections:

```
> use nfl_qbs
switched to db nfl_qbs
> db.collection.find({"Name":"Tom Brady"})
{ "_id" : ObjectId("5dd6d7036764ececaf3f9098"), "Rank" : 1, "Name" : "Tom Brady", "P
9,777 ", "Completion_Percentage" : "64.03%", "Passer_Ratings" : "97.30", "YPG" : "26
```