ETL project

The Objective: For this assignment is to utilize the predominant data flow of ETL. The data extracted is from the premier league website. We want to allow the user to access player information and their images. We will build a flask app that will combine CSV data with web scrapped data to create an interactive website that will return our ETL information.

Data Extraction: The data was read in csv format in the beginning. Part of the reading the data included initializing pymongo to work with mongo DB. Web scrapping was utilized to obtain player images. Merging information was required players information and players images from the url. A major obstacle during the extraction process was having to scrape from a site that requires continuous scrolling to obtain all the data. Once that was figured out, we also had to figure out how to clear a pop up ad that was preventing the continuous scroll coding. Outside of those two issues, the data was quite manageable. The HTML from the website was rather clear and the use of well-named classes made it easy to figure out where we wanted to scrape.

Transformation: The csv was manipulated with pandas, it was converted into a html. This was rather straight forward and didn’t require any coding that wasn’t covered in our Pandas instruction week. We only needed one step to transform the data scraped from the web; we needed to add the front of the URL needed to return the image we were scraping. A good transformation practice is to ensure the integrity of the data was maintained for instance, making sure a specific player was listed for a specific position for that season. This was something we feel we were able to accomplish at a high level.

Load: A lot of the coding required for the web scraping was tested in Pandas. We were then able to take that to create our app.py file which, along with Flask, would be utilized to load our data into the HTML file for output. In the app.py file, the web-scrapped data is uploaded into MongoDB. The CSV data had already been added to SQL and we actually merged that data with the MongoDB data in the load app. (This is where the transformation and the load ended up overlapping a little). There was a small issue with the link to player information not directing correctly, but that was able to be overcome by adding https: to the beginning of that specific render.