Colton Grutsch
ECON 8320
Professor White
May 10, 2023

Starting and working through a project to its end has displayed the capability that data collection and organization has, even on a scale as small as this. Throughout our semester, we were subject to many pieces of information and tools that can be utilized to instruct the computer in what we wish for it to do. Researching various methods and writing working code to reach an end goal allowed for some insightful firsthand experience as to what coding has to offer. Throughout the course of this project, there were many difficulties to match the simplicities. All in all, it allowed for a different perception into how computers operate.

As with every body of work, there were areas that went well, and some that were more difficult. To begin, locating the specific data in which the data frame was to be constructed went relatively smoothly. Writing a for loop that utilizes BeautifulSoup to parse through the website and select pages was successful. Originally, the code was going to be a while loop that used status_code == 200. However, using web scraping and parsing through 540 pages was inefficient during testing, so instead I opted to loop through a select number of pages for testing, with option to change the code to the maximum number for anyone wishing to wait for every page. As a result, the code was commented to suggest that the range could be made up to 541 for every available page. For the purpose of creating an example dataframe, I found that parsing through ten pages (150 entries) as an example was enough to demonstrate the code's capability. There originally was some difficulty in getting the url to change from page to page, but utilizing f-strings allowed the variable "i" to be inserted into each iteration of the loop as the URL changed and resulted in the code working properly.

Scraping through the website's code went faily smoothly to find the information to be pulled, but correctly looping through the elements of the code had its difficulties. It was quick to see that the information needed all shared the "li" tag, but in addition, each piece of data had its own specificities. Using li.find to gather data for the athlete name, sponsor, school, and sport was much easier than the value, since that information needed to follow a path to a new page of the website. Adding the additional details to the end of the website's base url that contains the value was a difficult process, but once done allowed me to use regular expressions to find numerical values. Difficulties arose from inconsistencies in how the website displayed the values, with some containing commas between the digits, and a select few containing $ symbols. As such, the regular expression code contained these, but then needed to utilize the replace() feature in order for it to be displayed in a uniform fashion, and allow for the mean to be taken if wished.

A minor problem that arose came in the sponsor name column of the data frame. I struggled for while to find why there were some rows that had the data correctly displayed, while others remained blank. Only after looking deeper into the data and doing some digging did I find that some were listed as agencies rather than sponsors. As such, I needed to alter my find_sponsor code to find correctly display both sponsors and agencies in the column in the data frame.

A large struggle in finalizing the dataframe came from attempts to pull the athlete's social media handles. Being able to parse through the website to find this information was difficult, as the url changes with https://nilcollegeathletes.com/athletes, rather than my original https://nilcollegeathletes.com/deals url. The web address changed with each athlete, instead of being consistent like the deals page numbers were. I ultimately was able to use the athlete's

name and add it to the website's original url to find each specific athlete's overall information, which then allowed me to compile both the Instagram and Twitter accounts from that page's scraped data.

Creating query functions that allow a user to input information to search the data frame was a process I did not have much knowledge or experience with, so doing some research that allowed for this code to operate properly was necessary. While these code blocks do allow the user to input the data they wish to find and be returned all associated results, I was unable to determine if there is a more simple or better format that would allow for a keyword to be typed to prompt a user to search, rather than multiple code lines. However, I did find that creating these queries to search the data frame went well, although it could have likely been organized and cleaned better.

While having a data frame in place that pulls relevant information from the website can be useful, I found that it was not necessarily the best indicator of the overall information given. As such, I wished to implement plotly express to create a figure that better shows what information is given. I played around with several figures to determine the best one to display this information properly, and ultimately ended up on the pie chart that shows the sport getting the most nil deals by percentage. Plotly express is a straightforward yet useful tool that I found was easy to pick up and provide useful information the user, and I could have used more such as a heatmap of the United States to show what states and associated areas are received the most deals.

Overall, I feel as though my code does a great job of organizing the data that is available from the website and presents it in a manner that is easy to read. If I were to do it over, I would have my data be pulled from the original website only, as the information for athletes, sponsors,

school, and sport are all pulled from the deals section of the website. I had originally planned to use the deals page since the "see details" link allowed the value to be seen, however many of the athletes either do not have a monetary value, or that data was not disclosed. As such, the Value column of the data frame has a majority of it listed as NaN, barring a select few. Since this data serves little to no purpose to someone wishing to see NIL deals in a broad scope, I could have fully omitted it and cleaned the data up accordingly.

Collectively I found that this project used many of the tools we learned to display information in an easy-to-read format for a set of data that it interesting to many people. While finding the data to pull using webscraping was generally simple, finding the lines of code to pull it and impliment it appropriately to the data frame had its difficulties. In the very early stages of the planning, I had prepared to use an API to find the specific data I wanted, but this thought was scrapped after finding that webscraping was not blocked. If I were to attempt to create a set of code to perform similar functions in the future, I would likely use one. Finding the information of the athlete, sponsor, school, and sport was much more straightforward than the value and social media handles. This was beneficial, however, as it allowed me to realize how to pull data from different types of elements. Altogether the information found in this data frame and the information that can be searched from it performs its intended functions, and properly utilizes many tools that data analytics has to offer.