ETL Report

Leah Lindy

Jose Miguel Davila

April 28, 2020

**Extract**

The extraction process involved scraping job posting information from 3 sites to create a database of available jobs. The scraping process was completed with chrome webdriver, which performs actions on the websites programmatically. For example, when reaching the landing page of each site ‘Data Analytics’ (job title) was typed into the search bar, then the ‘Find Jobs’ button was clicked. These actions were all defined by unique ‘class’ and ‘id’ of the fields, found in the HTML devtools. Once the search results had rendered the wanted results, this webpage could then be scraped with BeautifulSoup. The chromedriver was used again to iterate through multiple pages of results, and each page was scraped.

Example:

for x in range(10): # for 10 pages of results

html = browser.html

soup = BeautifulSoup(html, 'html.parser')

job\_postings=[] # empty list for job posting

job\_postings = soup.find\_all('div', class\_='jobsearch-SerpJobCard')

for job\_posting in job\_postings: # iterate through the posting to retrieve information

# Title appended to title list

title\_search = job\_posting.find('h2',class\_='title').find('a', class\_='jobtitle turnstileLink')

title.append(title\_search['title']) # Title appended to title list

# company appended to company list

company\_search= job\_posting.find('div', class\_='sjcl').find('div').find('span').text

company.append(company\_search.strip('\n'.))

# location appended to location list

location\_search=job\_posting.find('div', class\_='sjcl').find(class\_= 'location accessible-contrast-color-location').text

location.append(location\_search)

**Transform**

The transformation was completed concurrent to the extraction, as the page had to be scraped to retrieve data to be cleaned and organized. The soup received through iterations of each job posting on a single page was parsed through. The find\_all function looks through the soup by tag, then class, for example to find a job posting on the page, this line of code was used: soup.find\_all('div', class\_='jobsearch-SerpJobCard').

The individual job-posting would then be iterated through, as the remaining information (summary, salary, company etc) was nested within the job card. Remaining information was retrieved from the with the find\_all function as well and held in lists. One main issue that was faced during the transformation process was when scraping the websites there was often inconsistent layout, class names, formatting or missing information from one posting to another. For this reason, try and except code was often used to allow for these inconsistencies. This is an example where the try/except code needed to be included to allow for ‘N/A’ (missing summary):

try:

summary\_search=job\_posting.find('div', class\_='summary').find('ul').text

except AttributeError:

summary\_search="N/A"

This line of code also needed to be included to clean the data, as sometime ‘\n’ was written to go to new lines, or other code was indcluded in the HTML that was not necissayr to be included for our purposes:

summary.append(summary\_search.strip('\n'))

Information from the lists was then placed into a pandas dataframe.

**Load**

The database chosen to hold the information was SQL. This structured database was selected because the tables and the information they hold are defined and constant. The tables were defined in SQL then the pandas dataframes were loaded to SQL.

Analysis:

Some queries were run once the tables were loaded.