**Extraction, Transformation, and Load Technical Report**

**Where to Live:**

Population Density, Job Availability, and Other Factors to Take Into Account

**Table Of Contents**

1. Introduction ………………………………………………………………….. 3

1.1 Summary ………………………………………………………….. 3

1.2 Scope ……………………………………………………………… 3

1.3 Technologies and resource contributions ……………………… 3

1.4 Definitions, Acronyms and Abbreviations ……………………… 4

2. ETL Details ……………………………………………………………….. 4

2.1 Data Import/Extract Sources and Method ……………………… 5

2.2 Data Acquisition …………………………………………………… 5

2.3 Data Transform ……………………………………………………. 6

2.4 Data Integrity ……………………………………………………… 7

2.5 Data Refresh Frequency ………………………………………… 7

2.6 Data Security ……………………………………………………... 7

2.7 Data Loading and Availability …………………………………... 7

3. Data Quality ………………………………………………………………... 7

# 

# 

# 

# 

# 

# 

# 

# **1 Introduction**

*The purpose of the Extraction, Transformation, and Load (ETL) Technical Report is to capture details that pertain specifically to ETL portion of the data pipeline that is to be used in a data science project. This however does keep in mind the final target objective while performing the ETL.*

# **1.1 Summary**

We set out to analyze the population density and job markets of the United State’s largest cities. We wanted to see the ratio between city population and metro-area population, and to compare the availability of data-related jobs in these cities to help people make informed decisions on where to live and work.

# **1.2 Scope**

**Population Data**

We used Wikipedia as our source of population data for the United States’ largest cities and metropolitan areas. These lists were based on 2018 population estimates and - once we merged the two datasets - we had both city and metro area population data on about 150 cities.

**Job Data**

We scraped CareerBuilder for job data. We explicitly searched for jobs containing the term “data” in all of the cities present on our city and metro area dataset. We then took this data and created a new dataset containing the location, job title, company name, employment type, salary and job description of the jobs listed in each area.

# **1.3 Technologies and resource contributions**

**Team Members and Contributions**

* **Aaron Adams**: Led web scrape, troubleshooting during transformation of city and metro data, etc.
* **Patrick Kelly**: Assisted with web scrape, outline of project, and troubleshooting during city and metro data ETL process.
* **Keaton Stewart**: City and metro data transformation, general troubleshooting.
* **Austin Hayden**: City and metro data transformation, research to support city and metro data transformation.
* **Will Belcher**: City and metro data transformation and loading.

**Technologies Used**

**Pandas**: We scraped data from Wikipedia by using the Pandas “pandas.read\_html” function to read in all the table data contained on the page.

**Splinter**: We used Splinter to automate our browser actions and support our web scrape of CareerBuilder.

* In particular, we needed to add in search terms (eg. city name, the word ‘data’) and navigate through the many pages of results.
* We also had to add a “sleep” period of three seconds to allow result pages to load.

**Beautiful Soup**: We used BeautifulSoup to parse through the HTML.

**1.4 Definitions, Acronyms and Abbreviations**

**ETL**: Extract, Transform, Load

* **Extract**: This is the first step in the ETL process. During extraction, data is read, often from multiple databases, and then collected. In this project, data was extracted using web scraping, and collected into pandas dataframes.
* **Transform**: This is the second step in the ETL process. During this step, the data that was extracted in the extract step is converted (transformed) into a form that prepares the data to move on to the final step of the ETL process. Often times, this step involves table lookups or combining data with other data. In this project, the population tables were joined, cleaned up, and a new column was created with a density calculation.
* **Load**: This is the final step in the ETL process. During this phase, the transformed data is written into a new database, which can take on many forms. In this project, the transformed data was uploaded into a CSV file as a backup, but the data was also uploaded from a pandas dataframe into a SQL database.

**SQL (Structured Query Language)**: Query language used for requesting information from a database.

**Pandas**: Software library under Python that is used for data manipulation and analysis. In this project, this was used to transform data extracted from HTML.

**Web Scraping**: The process of collecting information across the internet. It emulates human Web surfing to collect specific pieces of information from selected websites.

* **Splinter**: Python module used to test Web applications and automating browser actions

**HTML (Hypertext Markup Language)**: This is a system used on the Web to tag text files, in order to set certain attributes such as font, color, structure, graphics, etc.

* **Beautiful Soup**: Python package used to parse through HTML and XML documents, by creating a parse tree for parsed pages.

**SSL (Secure Sockets Layer)**: We encountered a security error having to do with certification and verification in scraping the Wikipedia pages. We thought this might have to do with our use of the pandas read\_html function to access our URLs - we got around this error by visiting the sites in-browser, which loaded our SSL certifications somewhere we were able to call on in scraping.

**2.1 Data Import/Extract Sources and Method**

**Web Scraping**: This is a method of data extraction used on multiple Web pages.

**Wikipedia**: Used 2 Wikipedia pages to scrape data from. We specifically called upon pandas.read\_html() to read the HTML files:

1. <https://en.wikipedia.org/wiki/List_of_metropolitan_statistical_areas>
2. <https://en.wikipedia.org/wiki/List_of_United_States_cities_by_population>

**Career Builder**: We initially wanted to use other websites to give us job data, but had trouble with Indeed.com in dealing with pagination, occasional pop-ups / prompts for accounts verification. We landed on CareerBuilder because it wasn’t timing us out or blocking us from scraping. We used the Beautiful Soup module to perform this web scrape, so that we could mimic human interaction with the Web pages.

1. <https://www.careerbuilder.com/?cbRecursionCnt=1>

# **2.2 Data Acquisition**

**City and Metro Data**

* This dataset is largely static. The populations and relative density of each city wouldn’t need to be updated too frequently.
* We would probably only update this dataset in the event of a new census (every 10 years) or some other drastic, unlikely change.
* To do this, we would simply revisit the Wikipedia pages in question once they are updated, and re-run our script.

**Job Data**

* This dataset is quite fluid and may well need to be updated frequently. We could easily do this by simply running our CareerBuilder scrape again. Since job postings change regularly and for various reasons like economic fluctuations, changing seasons, etc, it would make sense to update this dataset regularly (daily, weekly, or monthly depending on user preferences).

# **2.3 Data Transform**

**City and Metro Data**

* For the city and metro data, the transformation process was largely straightforward. We were ably to quickly pull the data from Wikipedia straight into a dataframe. From there, we had to delimit the metro area names (eg., New York-Newark-Jersey City) and then match the city names to these delimited strings. We also had to clean up the city names, because many of them had footnotes and references. In addition, we converted numbers from strings to integers where we could. Once that was done, we performed an inner join on the city and metro dataframes on the city name, which took our list down from about 300 cities on each list to about 150 that were present on both lists.
* We also created a new field in this dataset called “density score” that was created by simply dividing the city population by the metro population of each city.

**CareerBuilder Data**

* The main challenge here was going through the HTML and finding the elements under each tag that we wanted to pull. We ended up doing this by using list comprehension with exception handling.

**Unforeseen Challenges**

* Some cities on our lists shared names, like Columbus, GA and Columbus, OH. For these, we simply kept the largest city in our dataset and dropped the others.
* For salary data, it would be ideal to extract that as a single number, but based on the formatting after Web Scraping, it was challenging to get a single integer value for many of the jobs. Moving forward, we would create lists of the minimum and maximum salaries for each job posting where that data was provided.

# 

# **2.4 Data Integrity**

* Several of the column headers in our city and metro data had unicode characters buried inside them that we could not initially see. We fixed it with the below code:
  + df.column = [x.encode('utf-8').decode('ascii', 'ignore') for x in df.column ]
* The jobs data has some inconsistencies. Salary data is listed as annual or hourly, and sometimes missing altogether. Company names have inconsistencies.

# **2.5 Data Refresh Frequency**

* The city and metro data are unlikely to be updated regularly, and even if they are, it wouldn’t be significant changes. Los Angeles isn’t going to lose five million metro-area residents in a year.
* The dataset that needs to be updated regularly is the jobs data. Job availability can change wildly with little or no clear reason behind it. Because of this, we would probably update our dataset daily.

# **2.6 Data Security**

This data set utilized publicly accessible information, that was easy to access, and does not contain any personal or sensitive information. Therefore, there were no HIPPA considerations for this project, and no additional privacy measures need to be taken for Encryption, Data Masking, Auditing, Backups, etc.

# **2.7 Data Loading and Availability**

We would simply post our result datasets to a static webpage in table formats.

# **3 Data Quality**

**Success Criteria**:

* Ability to extract data from Web pages
* Ability to organize data into manipulatable dataframes
* Ability to clean the data into usable forms
* Ability to join dataframes
* Ability to find useful data
* Ability to load new data format into SQL database and CSV files

**End User Success Criteria**:

* Ability for user to locate desired city and see population statistics
* Ability to identify job postings in the city of their choice

The success of this project hinged on having population available for the United States’ biggest cities, and then being able to find job data in each of these cities. Our key performance indicators included the success criteria listed above, and if accurate information was actually extracted from the Web pages. The total amount of results generated using Web Scraping was very close to our expectations, since we did mimic human interaction with Web pages, when it came to loading more job results for each city prior to extracting the data from the Web page.

Using the data found, we could create a website that displays the found information in a user friendly manner, so that users can select how dense of an area they wish to live, and then see salaries for jobs in those areas (or even select salaries, and see what cities have those salaries, and then see where the people live in those cities in terms of being more dense in metro area or city core). Our division of urban population by the total Metropolitan Statistical Area population renders what we have deemed a density score, which is given as a ratio of Urban population to Metropolitan population.

In terms of site acceptance testing, I think a good indicator would be to see if we (the group members) can find cities in which we want to live, find jobs in those cities using this project, and then applying to those jobs and seeing if we get hired! If we can find jobs in the cities we wish to reside, it is successful (as long as it is on the list of cities from Wikipedia). If we cannot find any job postings for the cities in which we desire to live, we were unsuccessful.