Web Scraping Career Postings

ETL Project

Scrape career websites for data scientist job postings

Chris Lichliter, Daniel Carmona, Brian Yu, Ana Gill

**Purpose/background**

The purpose of this project is to build a web scraper to extract and parse out job posting data from career sites.

**Data was extracted from the following website:**

* Indeed.com – (URL = 'https://www.indeed.com/jobs?q=Data+Scientist&explvl=entry\_level&fromage=3'

**The objectives of the ETL project are:**

* To explore data science related jobs posted for locations across the United States on indeed.com
* To upload a transformed dataset of job posting for Data Scientist positions for daily new career postings.
* To provide users with a dataset that provides the ability to query and find the latest career postings.

# Description of Analysis, Extraction and Data Cleanup

The scraping was conducted using the get “requests” and “BeautifulSoup” libraries in Python to gather and parse information from indeed. Pandas library was used to assemble the data into a DataFrame for further cleaning and analysis.

Data was loaded to PostgreSQL to create a database that uses a live script to run daily queries.

**Data Analysis**

* First, using the inspect tool (Figure 1) in Chrome we explored the indeed website to examine the page structure and identified what to extract from the webpage by using either an element, ID, or class.
* After website evaluation it was determined that various classes could be used to extract the data as seen in the image below.

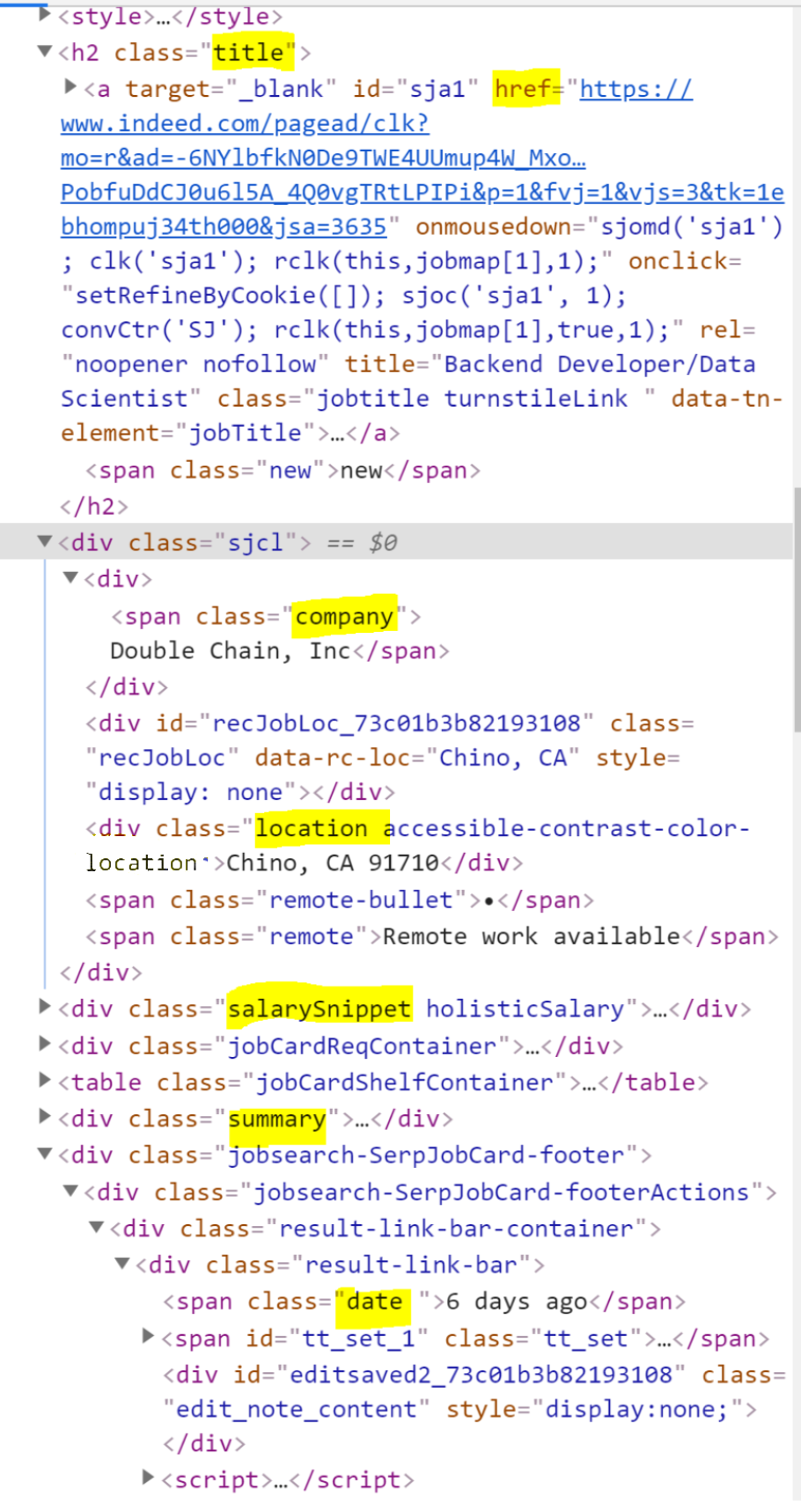


Figure 1. Inspect Tool



Figure 2. Scraping posting

**Data Extraction and Cleanup**

* Scraped job title and company class and removed “\n”.
* Scraped location class and split the field into two columns, city and state. Also, excluded fields that contained remote work.
* Scraped date class and converted results to “int”. Subtracted from current date with date time. Using try/except, all values with “Just posted” or “today” were changed to 0.
* Scraped salary class and added null value if salary was not included.
* Scraped job abstract.
* Used a loop to get the data for every single listing.
* Made a Dataframe (figure 5) to store for further analysis.



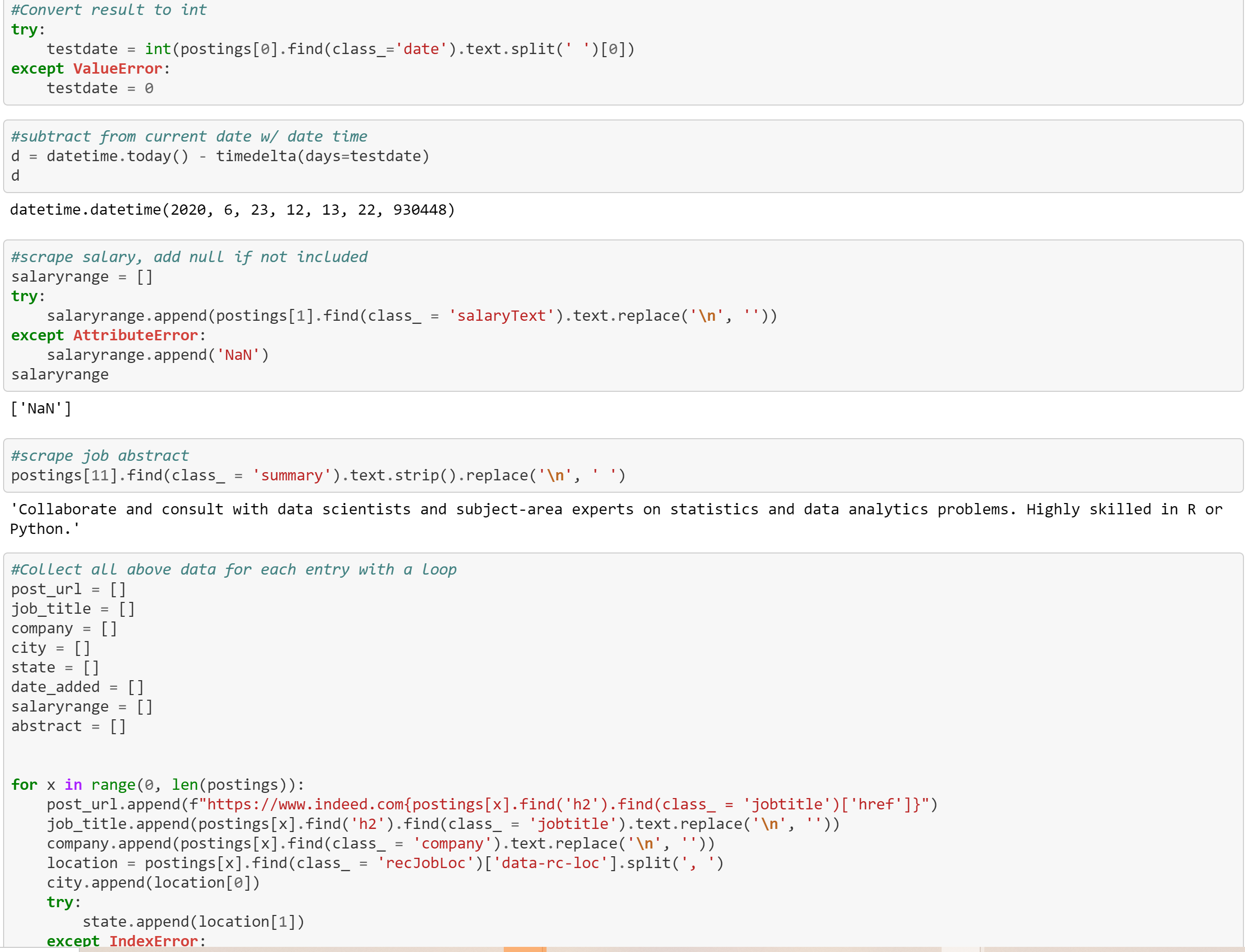


Figure 3. Extracting and Cleaning Data

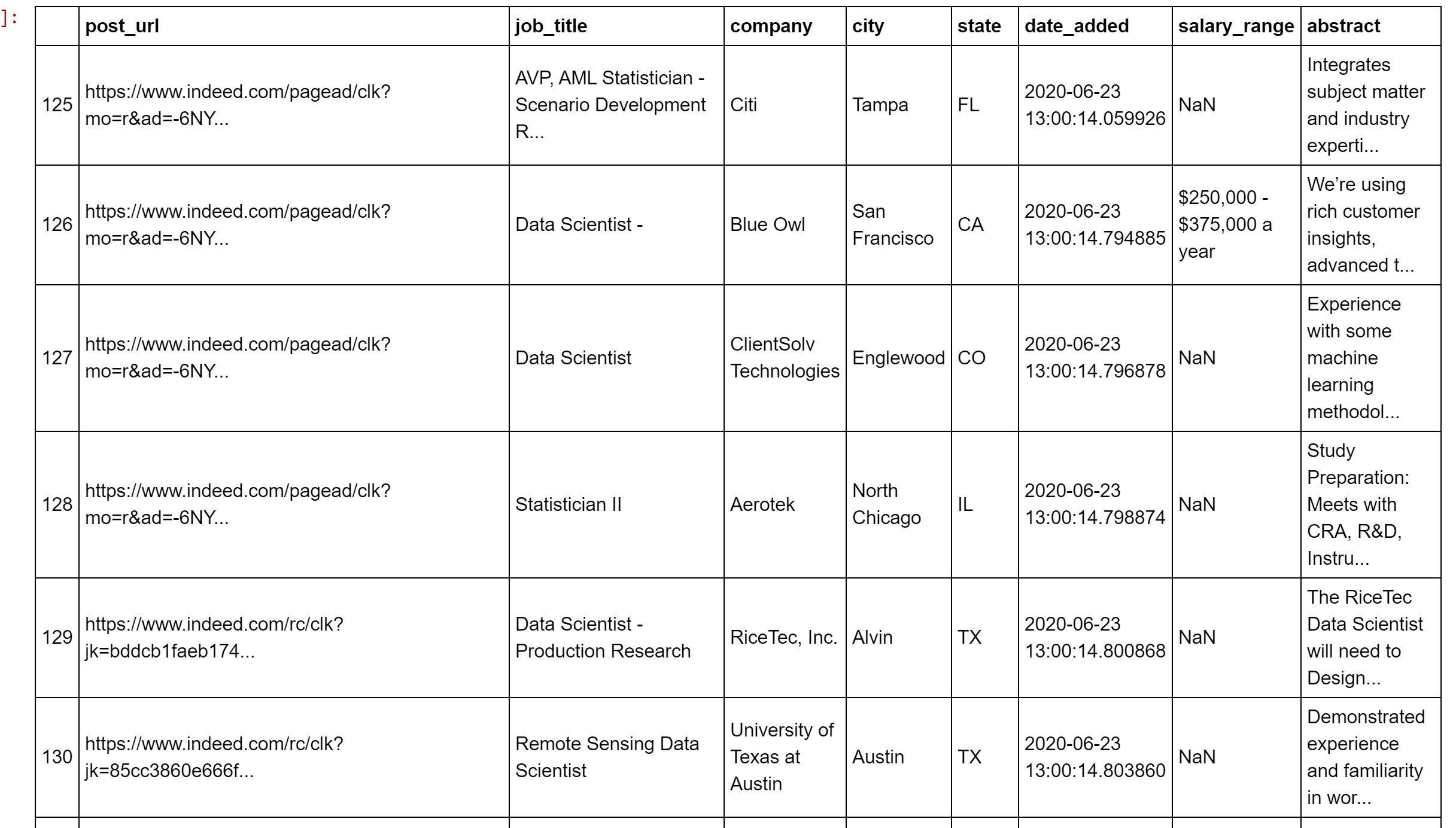


Figure 4. Scraped Data

**Loading to Postgres**

* Used Pandas to load CSV converted DataFrame into database.

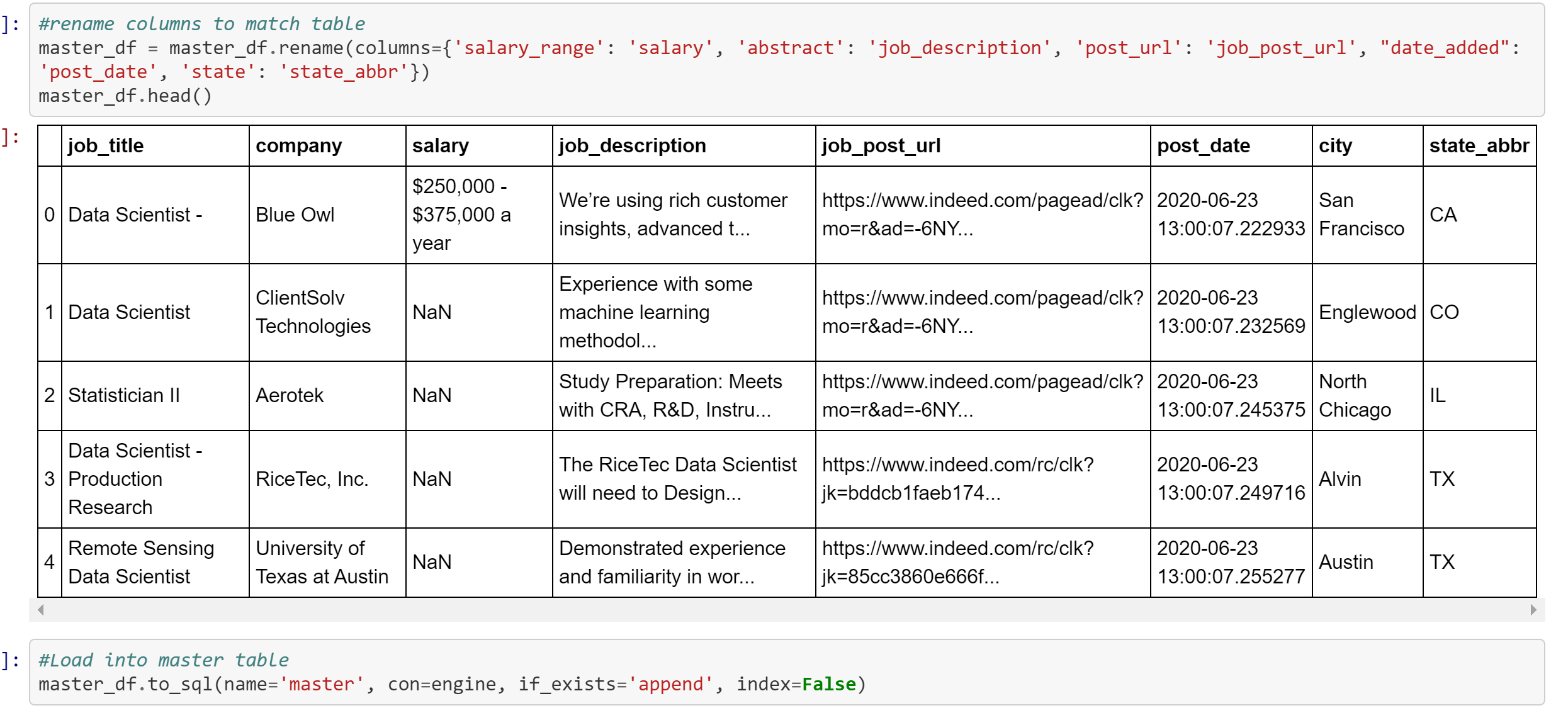


Figure 5. DataFrame



Figure 6. Tables

**Limitations/Future Work**

Some of the limitations on this analysis include:

* Time constraint.
* Limited skills with ETL tools for automation techniques.

Future work includes:

* Expand database to include additional data sources.
* Automation – generating the code to pull the top 10 job postings with the highest salary.
* Load database to a front end – create webpage to display the database.