Programming Take-home:

Part One - Web scraping and DB loading

Note that this problem uses two different definitions for the acronym API. It can refer to and oil and gas wellhead identifier API (American Petroleum Institute), and software API (Application Programming Interface).

We hope the context will make it apparent which definition applies, but please ask if it is not clear.

All oil and gas wells are assigned a unique API number. The **apis_pythondev_test.csv** file contains a list of several hundred API numbers.

You will scrape oil and gas well data from a website and load the results into a **sqlite** database.

View this example of a well on the webpage:
https://wwwapps.emnrd.nm.gov/OCD/OCDPermitting/Data/WellDetails.aspx?api=30-045-35432

Scrape data from the website for each API and load into a single SQLite table called *api_well_data* that has these columns:

Operator, Status, Well Type, Work Type, Directional Status, Multi-Lateral, Mineral Owner, Surface Owner, Surface Location, GL Elevation, KB Elevation, DF Elevation, Single/Multiple Completion, Potash Waiver, Spud Date, Last Inspection, TVD (*), API, Latitude, Longitude, CRS (*)

(*) Note that TVD is an abbreviation for True Vertical Depth, and CRS is an abbreviation for Coordinate Reference System.

Part Two – Set up an API to serve this data

1) A 'well' endpoint will return all database fields for a single API number.

Write a GET endpoint where the customer can pass an API number to retrieve all available data for that well.

2) A geospatial polygon search endpoint that will return a list of well API numbers that are within a polygon.

Write a GET endpoint where the customer can pass a list of (latitude, longitude) pairs that define a polygon to retrieve all API numbers located within the polygon.

Additional Info:

Create a public GIT repo containing your solution files and send the repo URL to felix@synmax. The repo will contain:

1) Your files containing web scraping and DB loading code for Part One.

2) Your python files containing your API written in Part 2.

3) All necessary information to spin up the API locally.

4) The **sqlite.db** file containing the web scraped data

5) A csv file containing a list of API numbers returned by your API polygon endpoint for this polygon:

[(32.81,-104.19),(32.66,-104.32),(32.54,-104.24),(32.50,-104.03),(32.73,-104.01),(32.79,-103.91),(32.84,-104.05),(32.81,-104.19)]

Evaluation:

It is up to you how you want to structure your code, but you will be assessed on the following things:

- Correctly scraping the fields for all API wells and loading them into the database.
- A functioning API that returns the correct data.
- Creativity of your solution.
- The quality of your Python. Are you using functions, classes etc. appropriately.
- In-code documentation and readability