Python Exam: Insight Data Engineering

Introduction

As a data-driven project, the main objective of this challenge is to collect some data from an external source (Yelp FUSION API) and then perform data modeling for an analytics platform.

Technology Stack

- Python Programming Language
- Database (Mysql, Postgresql or MongoDB)

Goal

The goal of this test is to evaluate the following of the applicant's abilities:

- be able to apply problem solving skills to formalize general problem statements into precise algorithmic solutions
- demonstrate an understanding of the documentation from the Yelp FUSION API
- overall knowledge about the chosen technology stack
- basic knowledge of git (don't get stuck on the 'git part' of the exam; this is not mandatory knowledge; you can ask for help)

What should be done

Scraping

- **Create** a python program that can be run via command line. This program must execute HTTP requests to Yelp Fusion API endpoints (https://www.yelp.com/fusion) and **persist their data on a local database**. Note that you will need to read and comprehend the API documentation to properly select the endpoints that should be used.

    About the Yelp Fusion API

    The Yelp Fusion API allows you to get the best local content and user reviews from millions of businesses across 32 countries.
    Yelp Fusion API uses private API Keys to authenticate requests. To authenticate the call to an endpoint, there are only 2 steps:
    - ***Create an app to obtain your private API Key**.*
    - *Authenticate API calls with the API Key.*

***Be aware that you can make 5,000 calls per day**. Any call you make after that on the same day will get back a response with status code 429 (Too Many Requests). Daily limits are reset every midnight UTC time.*

- It must accept a *location* string (like "New York City") or *latitude* and *longitude* if the location is not provided. Additionally, it should accept an optional argument *term* that should expect a string (like "food" or "restaurants").
- Using the provided input (*location, latitude, longitude, term*), the program should extract data from the Yelp Fusion API, **persisting up to 50 businesses sorted by review count (in descending order) and their respective reviews**. (Note that the reviews can be extracted using the **business ID** from the reviews endpoint)
- The local database must contain the following data from the Yelp service:
    - Business ID
    - Business name
    - Business address (zip code, city, state, country)
    - Business Review count
    - Business Review rating
    - Latitude and longitude
    - Business categories
    - For each review: review ID, review rating, review user ID, review user name, review date time created, review text

Data Analysis I

- Second, a python program that can be run via command line and will perform data analysis using as input the same database of the first program (Scraping). Such program must accept a date string (format: "YYYY-MM-DD") as an argument and then **create a csv file** containing the following columns:
    - Business ID
    - Business name
    - Business city
    - Business state
    - Number of reviews since the given date
    - The mean (and its standard deviation), median and mode of the reviews rating since the given date

Data Analysis II

- Create a python program that accepts an input csv file (we will provide such file) containing a list of cities. The program must then generate an output file containing the original input, but now with a new column *'top_category'*. The top category of a city is the most frequent business category of the businesses located in that city. This program

should use the business data stored in the **local database** (as used by the first and second programs above), not the API.
- If the city has more than 1 category as the *top category* then concatenate them using commas. Ex.: category1,category2
- Do note the input CSV file and the Yelp API data come from completely different sources, and thus may contain minor differences in the spelling of the city names. Therefore, you'll need to perform at least **some degree of normalization** (casing and accentuation, at a minimum) before comparing them.

Optional

A. Use mongodb instead of a relational database.
B. Adapt the third program (Data Analysis II) to also **accept a kml file** (we'll provide such file) as the input, instead of a list of cities. The kml file defines a geofence, which the program should use as the business filtering criteria (i.e. it must first identify on the database **all businesses located inside the geofence**). Then, the program should output the top category for such businesses (more about kml files in https://developers.google.com/kml).
C. Using the **string distance metric** Levenshtein distance (https://en.wikipedia.org/wiki/Levenshtein_distance) compute, for each business in the **local database**, the other business (also in the database) with the most similar **name**.

Nothing is true. Everything is permitted

The applicants are allowed to use third-party libraries and add external open-source tooling. Just beware of time restrictions. Feel free to use the internet for research as well. PS: copying and adapting proprietary code that is brought from home or from a former company you worked at is NOT ALLOWED because it misses the point of a hands-on exam and the abilities we are trying to assess, plus we consider it unethical in an exam (try to work with the instructions provided and research)