Code test for data engineering candidates

Purpose

This test is designed to showcase your understanding of databases and data processing, together with your aptitude in a programming language of Python.

There are two stages to this code test:

1. Preparing code at home ahead of your interview.
2. Presenting the results to us in a technical interview.

The technical interview is to give us an indication of what it will be like to work together, and should be regarded as a collaborative effort.

Prerequisites

- Knowledge of relational databases, including how to create tables, insert data, and query data. For the purpose of this test, we are using MySQL.
- Knowledge of a Python, including how to read and write files, process data, and access a MySQL database.
- Familiarity with Docker for container management, which we use through the Docker Compose tool. You will need Docker and Docker Compose installed on your development machine.
- Familiarity with Git for source control, and a github.com account which will be used for sharing your code.

We have included example data and programme code. The example schema creates a simple table, with example code in several common programming languages to load data from a CSV file and output to a JSON file. There are instructions towards the bottom of this document explaining how to use the Docker containers, start the database, and use the examples.

Background

We have provided You with a folder containing:

- A **docker compose.yml** file that configures a container for the MySQL database, and the example scripts' containers.
- An **images** folder containing example programmes showing how the database can be accessed from Python.
- An **example_schema.sql** file containing a table schema used by the example scripts.
- A **data** folder containing four files:
  - **example.csv** A tiny dataset, used by the example scripts.
  - **places.csv** 113 rows, where each row has a city, county, and country name.
  - **people.csv** 10,000 rows, where each row has a first name, last name, date of birth, and city of birth.

- **sample_output.json** Sample output file, to show what your output should look like.

Problem

There are a sequence of steps that we would like you to complete. We hope this won't take more than a couple of hours of your time.

1. Create your answers in a github account of your own.
2. Devise a database schema to hold the data in the people and places CSV files, and apply it to the MySQL database. You may apply this schema via a script, via the MySQL command-line client, or via a GUI client.
3. Create a Docker image for loading the CSV files, places.csv and people.csv, into the tables you have created in the database. Make sure the appropriate config is in the docker compose file. Your data ingest process can be implemented in any way that you like, as long as it runs within a Docker container. You may implement this via programme code in a Python, or via the use of ETL tools.
4. Create a Docker image for outputting a summary of content in the database. You may implement this using Python. The output must be in JSON format, and be written to a file in the data folder called **data/summary_output.json**. It should consist of a list of the countries, and a count of how many people were born in that country. We have supplied a sample output **data/sample_output.json** to compare your file against.
5. Share a link to github repo with us so we can review your code ahead of your interview.

We have provided an example schema and code that shows how to handle a simple data ingest and output.

Details of how to run and connect to the database are below, together with how to use the example schema and code.

Notes on completing these tasks

- There is no right way to do this. We are interested in the choices that you make, how you justify them, and your development process.
- Consider how normalized your schema should be, and whether or not you should be using foreign keys to join tables.
- When you create a container, make sure that you add the container config to the docker compose.yml file, and add your Dockerfile and code to the images folder.
- Make sure that your code is executable, and if you are working in a scripting language, make sure that your script has an appropriate “hash-bang” line (as featured in our example scripts).
- Most of the example code uses ORM libraries to connect to the database. This is _not_ essential for the purpose of this test: your code should connect to the database and your queries should be implemented in whatever way you are most comfortable with.
- Consider what kind of error handling and testing is appropriate.
- All data input, storage, and output should be in UTF-8. Expect multi-byte characters in the data.

- The MySQL database storage is ephemeral; it will not persist, so make sure all schema and data queries are repeatable.
- You may find it easier to work with a subset of the data when developing your ingest.

Notes on using the images in the git repo

Requirements

Make sure you have recent versions of Docker and Docker Compose.

Building the images

This will build all of the images referenced in the Docker Compose file. You will need to re-run this after making code changes. (You can also specify individual services to build if that is more convenient.)

```
docker compose build
```

Starting MySQL

To start up the MySQL database. This will will take a short while to run the database's start-up scripts.

```
docker compose up database
```

Optional: if you want to connect to the MySQL database via the command-line client. This may be useful for looking at the database schema or data.

```
docker compose run database mysql --host=database --user=codetest --password=swordfish codetest
```

Example scripts

We have provided example code written in Pythont. This show how to use a programme in a separate Docker container to connect to the database, using an ORM library where appropriate, to load data from a CSV file, and to query data to output as a JSON file. There should be regarded as illustrative; it is fine to use any of these examples as the basis of your own solution, but we would prefer that you use technologies that you feel comfortable with.

Make sure the MySQL database is running, and then load the example schema with:

```
docker compose run --no-TTY database mysql --host=database --user=codetest --password=swordfish codetest <example_schema.sql
```

Then make sure that the containers have been built with `docker compose build` and run one or more of the sample programmes with:

```
docker compose run example-c
docker compose run example-node
docker compose run example-python
docker compose run example-r
docker compose run example-ruby
docker compose run example-swift
```

In each case, the programme loads data from the data/example.csv file into that table, and exports data from the database table to a JSON file in the data folder. Note that the scripts do not truncate the table, so each one you run will add additional content.

Cleaning up

To tidy up, bringing down all the containers and deleting them.

```
docker compose down
```