Programming Challenge

<u>Requirements:</u>

Please use **Python**, and ONLY Python, to solve the coding part of this task. Java or similar languages such as Rust, Go, should not be used. JavaScript and CSS may only be used to enable interactivity and style, respectively. HTML may only be used as a template.

Please use a ".py" Python script for all parts needed to complete task 1. For task 2, please provide a jupyter notebook and if you need to show something specific from task 1 (e.g., some visualization), you may also, additionally, provide this in the same jupyter notebook.

<u>General Task Description</u>

You are presented with an income dataset from a Census. We are going to build a **well-orchestrated microservice** which integrates multiple services and uses the dataset as a base to drive decision making. This dataset is presented in the file called "**Input_Dataset**" in this folder. This dataset provides insights about the general population at the time the Census was conducted. The gathered information includes data on age, gender, country of origin, marital status, housing conditions, marriage, education, employment, and some other fields. The task will be to try to persist this data for future use and derive some insights from it, in a web application.

Some column descriptions:

- **fnlwgt** represents a final weight which indicates the number of units in the target population that the responding unit represents.
- **education_num** represents the number of years of education in total, and this correlates with the column "**education**".
- **relationship** represents the respondent's role in the family.
- **capital_gain** and **capital_loss** are income gain/loss from investment sources other than wage/salary.

You can ignore the **fnlwgt** column in your insights, but for completeness, please still do save this in the database.

This task is divided into five parts, with a bonus section:

<u>Task 1: Ingesting the data into a database</u>

Your task here is to:

1. Inspect the dataset
2. Determine which columns are relevant, in order to help us create a database using the data from the *Input_Dataset* file.
3. Clean the dataset, if needed
4. Create a **normalized** database schema from the identified columns in point 2 above and establish relationships between the tables in your database
5. Using the database schema sketched out in point 4, create a data model script (in a .py file) that can create a simple physical SQLite .db database (please don't use an in-memory database)
6. Building on point 5 above, create a .py script which can upload the data from the *Input_Dataset* file to the SQLite database .db created

Hint: You should have at least two scripts: one for the database schema and one for uploading data using the database schema. Please only use modern ORMs like SQLAlchemy

After creating the SQLite database and wrangling the data from the *Input_Dataset* file into it, you can proceed to Task 2

<u>Task 2: Data insights</u>

(Please use a jupyter Notebook for this section)

Your task here is to:

1. Thoroughly inspect the dataset. You may either use the database you created, or you may directly use the *Input_Dataset* file
2. Please use the dataset to **derive 6 key insights** which are not immediately obvious when the file is opened. Ideally, these insights should have a **wow** effect
3. For each of the insights discovered in point 2 immediately above, explain briefly in a comment (in the notebook) why you think this insight is important and how you arrived at the insight
4. Create a question out of each insight and using the appropriate answer you discovered, add these to a JSON file. The JSON file should have the key as the question and the value as the answer

Task 3: Web Application

We have been asked to create a web application with which analysts can perform three different kinds of operations which have strict requirements.

- **In one tab**, they should be able to view a human-readable form of a dataset pulled from a table in the database. Please provide a dropdown with which they can target a table in the database for which they want to display data. **Please also provide them the opportunity to filter out columns which they don't need**. For example, they might want to have a dataset in their display view which only has data from columns related to age and income
- **In a second tab**, they should be able to visualize data from the database based on filters. The goal here is to be able to filter the dataset in such a way that they can derive useful insights. They should be able to export the visualizations as a .png or .jpg
- **In a third tab**, they should be able to export a subset of the dataset which is pulled from the database

It's important that the UI of the application is user friendly. Please **only** use web frameworks in Python for this step.

Task 4: Making our app distribution ready

We have been asked by upper management to make our web application written in Task 3 production ready. To do this, we have chosen to containerize our web application:

- Please create a working Dockerfile which describes all the languages, steps, frameworks, packages, and others required to run the application
- Create a docker compose file which orchestrates services
    - In the docker compose file, one of the services should be a Postgres database. The other service(s) should be one that runs the web application. We will use this Postgres database, instead of a SQLite database, in production, to serve data to our application. Be sure to have your upload script also pointing to this, perhaps using a switch from SQLite to Postgres?

Task 5: Updating, testing, and documenting

To ensure integrity of our processes, we would like to make updating the application, which we plan to host on a Linux server, very easy. The data engineers should be able to build and restart a container using a docker compose file, all the needed python scripts, and a bash script.

- Please create a bash script which we can use to pull code from a repository, rebuild our application, and run the docker compose file in the repository
- Please also add unit tests to test the upload scripts and connection to the SQLite and/or Postgres database
- Finally, please include appropriate documentation in a readme (.md) of important instructions which data engineers can use to launch the application and carry out all the infrastructure requirements in your absence

(**Bonus task section, if you feel up to the challenge**): Create a simple API endpoint with which users can query the database for defined filter parameters (e.g., age, gender, marital status) from the dataset in the database and receive a JSON response. Please don't use Django to create the API endpoint. There are easier and faster frameworks in use today.

<u>What should you send to us?</u>

1. Please upload the scripts and files you want to showcase to us to a public GitHub repository and **send us a link to this GitHub repository**
2. Ideally, the repo should contain the scripts and files from task 1 to task 5, including the jupyter notebook from task 2 and the JSON result (and if you worked on it, the script from the Bonus task to serve the API endpoint). We don't need the .db file you will create, as we should be able to create this using the provided scripts.

That's it from us! Happy coding and looking forward to what you orchestrate 😊