Technical Test - CSV Data Cleansing

Instructions

1. There will be 3 tests. All tests is **MANDATORY**.

2. You will be given **8 HOURS to finish all tests.**
   Please do maximize the time you have, stay hydrated, and do not forget to eat your healthy food.

3. This test requires you to install your own database (for Test 1, 2, and 3), and docker engine (for Test 2).
   Please refer to their documentations to download and install.

   - Docker Engine
   - Mysql Local or Containerized
   - Postgresql Local or Containerized
   - ClickHouse Local or Containerized
   - DuckDB Local or Contanerized

   P.S. only one database needed!

4. It is **MANDATORY** to create **ONE** document file as README.md
   which covers both Test 1 and Test 2 with Markdown format.

   The document must contain:

   - A short explanation about the script.
   - How to run the script.
   - Anything you want to explain about the script,
     e.g. How to test, expected result, etc.
   - Any possible improvements you made.

   P.S. please spare your time to create the document as it will be reviewer **FIRST FOCUS**

5. Please upload the solution to your Google Drive with folder name format:
   [Your Full Name] - Senior Data Engineer,
   set the general access to “Anyone with the link”, and **write the share link in body Email**.

6. Additional information will be given in Email, please read it carefully.
   If any instruction given via Email contradicts with the instruction above,
   please follow the instruction given in Email.
   We may dynamicly change the instruction via Email.

7. Please do not hesitate to contact us if you have any questions.

Test 1 - Clean CSV Data with Python Script

Requirements

You have been given raw CSV data. It has lot of duplicate rows and unproper data type.

Your are assigned to clean, transform, and insert the data into a database table.
The raw CSV file located in:

- Path: **/source**
- Name: **scrap.csv**

And, database location is available at:

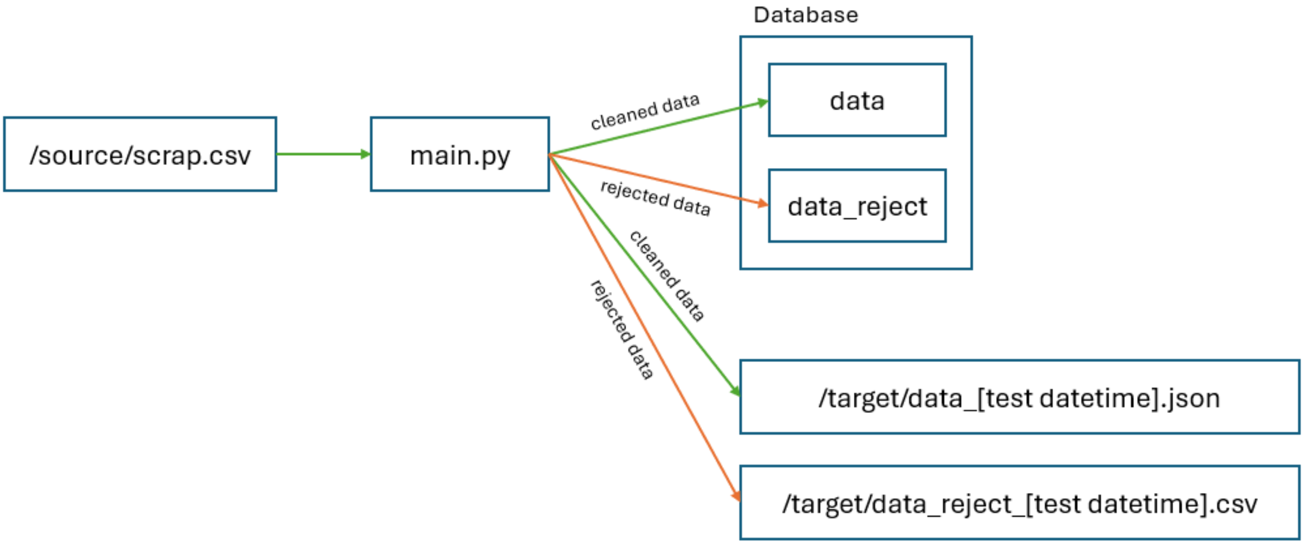
- DB Address: **any**
- DB Port: **any**
- DB Schema Name: **any**

Also, he ask to create a proper backup CSV and JSON file for duplicate and clean data respectively.
Both file format mentioned below.

There is no specific rules to finish the task, but he recommend to use Python programming language as it is the common language our division used.

Tasks

The overall task can be seen below.



1. Read and clean the CSV file from **duplicate *ids***.
   Record in column *ids* has to be unique for all rows.

2. Insert clean and duplicate record to database table with:

   - DB Table Name (duplicate data): **data_reject**
   - DB Table Name (clean data): **data**

   Remember: DB columns must correspond with CSV columns for both tables. Means, if CSV file has 10 columns, so both table does.

3. Create CSV file for duplicate record with:

   - Path: **/target**
   - Name: **data_reject_[Test Datetime, with format YYYYMMDDHHMMSS].csv**

   The CSV file must have same format with raw CSV file.

4. Create JSON file for clean record with:

   - Path: **/target**
   - Name: **data_[Test Datetime, with format YYYYMMDDHHMMSS].json**

   The JSON file must follow format below.

   ```
   {
       "row_count": [integer],
       "data": [
           {
               "dates": [string, with date format YYYY-MM-DD],
               "ids": [string],
               "names": [string, uppercase],
               "monthly_listeners": [integer],
               "popularity": [integer],
               "followers": [integer],
               "genres": [[string], ...],
               "first_release": [string, with year format YYYY],
               "last_release": [string, with year format YYYY],
               "num_releases": [integer],
               "num_tracks": [integer],
               "playlists_found": [string],
               "feat_track_ids": [[string], ...],
           },
           ...
       ]
   }
   ```

   For example, examine how */example/scrap.csv* was cleaned and backed up in */example/data_20240302101010.json* with duplicate rows stored in */example/data_reject_20240302101010.csv*.

Rules

1. Write your solution in *main.py* file given.
   It is allowed to use any PIP module needed, and add more class or function.

2. The database address, port, and schema is not defined.
   You have to install the database on any location, either in your local machine, VPS, VM, or any cloud provider you have.
   It is **NOT NECESSARY** to submit the database address, but you have to submit one DDL script as *ddl.sql* file which contains SQL create table statement for table data and data_reject.

   Please provide screenshots that show the total row count for each tables.

3. For DB table column type, please choose it properly as it will be considered in assesment evaluation.
   Hint: You can reflect to JSON data type mentioned above.

4. **Ensure** the JSON and CSV file name has proper file name according the test datetime.
   For example:
   If test held on 2 March 2024 10:15:20, the file name suffix has to be *data_20240302101520.json* and *data_reject_20240302101520.csv* respectively.

Notes

1. Assessment will be made from:

   - The *main.py* and *ddl.sql* files.
   - The JSON file contain clean data. It must has correct name, path, and format.
   - The CSV file contain reject data. It must has correct name, path, and format.

2. It will be a **point plus** if you:

   - Create any error handler for this task.
   - Create a test script with PIP module *unittest* or *pytest*.
   - Make any other improvisation which does not violate the given requirements.

• • •

Test 2 - Containerize the Application with Docker

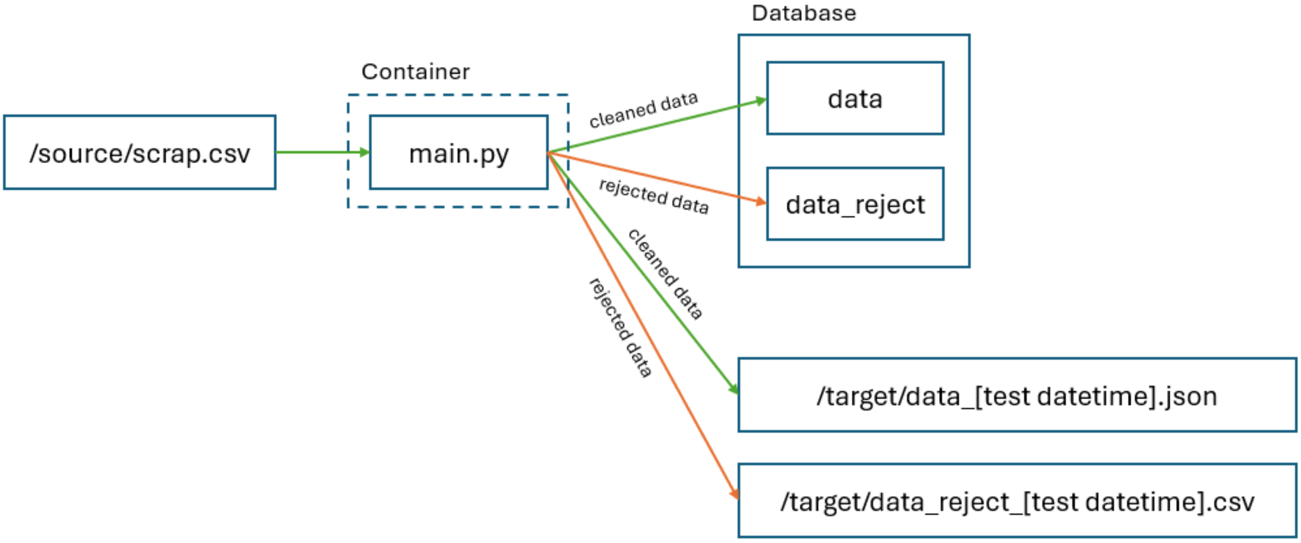
Requirements

Your senior give applause for your work on Test 1. Now, he want to attach hourly schedule to your *main.py* script in Test 1 using a scheduler deployed in Docker Environment.

It is **NOT your task** to create the scheduled pipeline.
But, your senior give you task to **bundle your application into docker image**, and try to **run and debug it as a container on your local machine** before it will be deployed with schedule.

Tasks

The overall task can be seen below.



1. Create a docker image using *Dockerfile* file, to bundle the application you create on Test 1.
2. Run and debug the docker image using *docker-compose.yaml* file. Ensure the application is not malfunctioning.

Rules

1. There is no restriction with *Dockerfile* file.
   Please consider the base image, and the build steps wisely.

2. There is also no restriction with *docker-compose.yaml*.
   It is preferrably to use *docker-compose.yaml* file.
   But, if you choose to use native `docker run ...` command, please write the command in *How To Run* section on your documentation.

3. It is also not an issue of using Dockerfile COPY command, or VOLUME MOUNT method to mount the */source* and */target* path. Please consider it wise and reasonably.

4. There is no rule for image and container name. Use any relevant name.

Notes

1. Assessment will be made from:

   - The *Dockerfile* file.
   - The *docker-compose.yaml* file, or a *How To Run* section in your documentation.

2. It will be a **point plus** if you:

   - Make any other improvisation which does not violate the given requirements.

Test 3 - Implementation on any Scheduler

Requirements

There is no requirements necessary for Test 3. Prepare **ONE** comprehensive ideas.

Tasks

Write your idea (and insert any visual, or chart if necessary) in a file from this question:

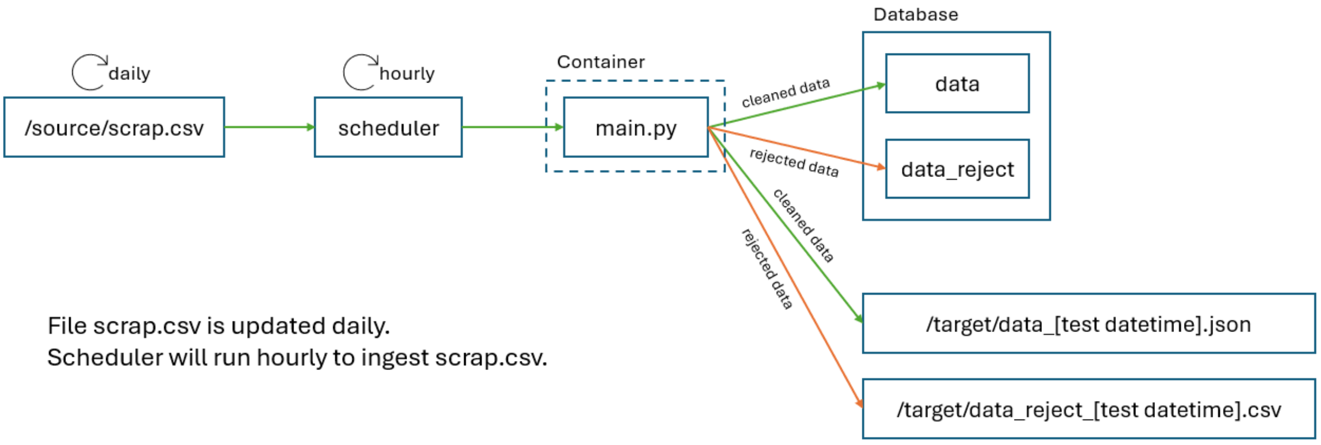
- **How to schedule/implement work you have done on Test 1 and Test 2 with a scheduler?**
- **How you will handle any possible issue might happen with the data (like duplicate data in DB because each scheduler runs may ingest the same data from scrap.csv) after the the pipeline being scheduled?**

Your idea can be developed from the assumptions below (but not limited to):

1. Scheduler can be **ONE** from list below (but not limited to):
    1. Basic Cron
    2. Apache Airflow
    3. etc
2. The schedule will be scheduled hourly.

Also, you are allowed to add more assumptions to build your idea.

To make it easier, you can see diagrams below.



File scrap.csv is updated daily.
Scheduler will run hourly to ingest scrap.csv.

Rules

1. You can start to prepare your answer right after you submit your assignment for Test 1 and Test 2, until interview session.
2. You can bring any helpers like diagram or flowchart during the interview to help you better explain your idea.
3. It is not necessary to create any script related with the tools.

Notes

1. The test form is discussion during interview session. Hence, it is not necessary to make any document for your idea.
   Assessment will be made from the discussion.