Data Platform Engineer
Take Home Test 2024

Please complete this assignment within **5 days**
after the time this email has been sent.

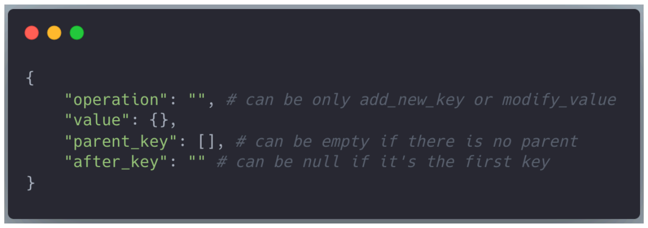
Problem 1 - Make Them Equally

Given two data in JSON format (Let's say Source and Target). Your goal is to find differences and list how to make Source equal to Target concerning the order. The JSON data contains only key-value pairs and can have a nested structure. It does not contain any array of JSON. If Source has a key that Target does not have, you may skip it. To make **Source** equal to Target, there are two operations that you can do which are *add_new_key* and *modify_value*. The *add_new_key* operation is to add a new key-value pair into the data. The *modify_value* operation is to edit the existing key-value pair. In each difference, please log an operation name, a value that needs to be modified or added, a list of parent keys (if have), and a key name that is located before the affected key.

Please create a program to solve the above problem. It can be a function that receives Source and Target, finds the differences, and returns the result. You can use any programming language to solve it. Please note that the Source and Target must have the order of keys (even if the standard of JSON is unordered). You can use any data type that can be exported to JSON format. For example, you can use a Python dictionary, Java GSON, or JSON string to represent Source and Target. The order of the program's output can be unordered actions.

The expected output of the program must be the **list of actions** to make Source equal to Target (No need to write a program to manipulate Source equal to Target). The format of output must be an array of elements where each element must contain the following keys

1. **operation** - is an operation that supports only *add_new_key* or *modify_value*
2. **value** - is a target data that will be added or modified
3. **parent_key** - is a list of parent keys of nested structure of JSON format. If it's a root structure. It can be empty.
4. **after_key** - is a key that is located before the modified / added key. It can be null if the key is the first key in the structure



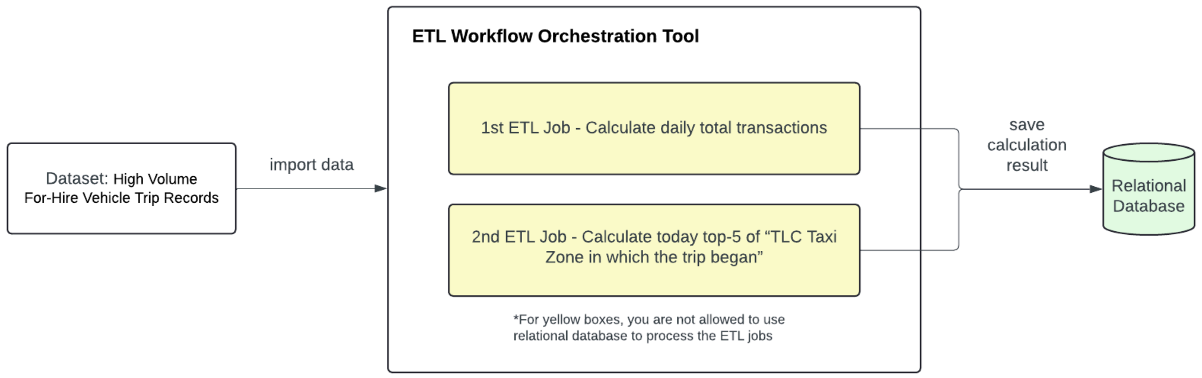
```
{
    "operation": "", # can be only add_new_key or modify_value
    "value": {},
    "parent_key": [], # can be empty if there is no parent
    "after_key": "" # can be null if it's the first key
}
```

Please see an example in zip file, where

1. **source.json** is the Source JSON data
2. **target.json** is the Target JSON data
3. **answer.json** is the expected answer of how to make Source equal to Target
4. **example_differences.png** is the screenshot of differences between Source and Target

Problem 2 - Design and Implement ETL Platform

Please set up an ETL platform, and demonstrate two data pipelines based on a given dataset. The results must be stored in a relational database. Please see the full requirement below

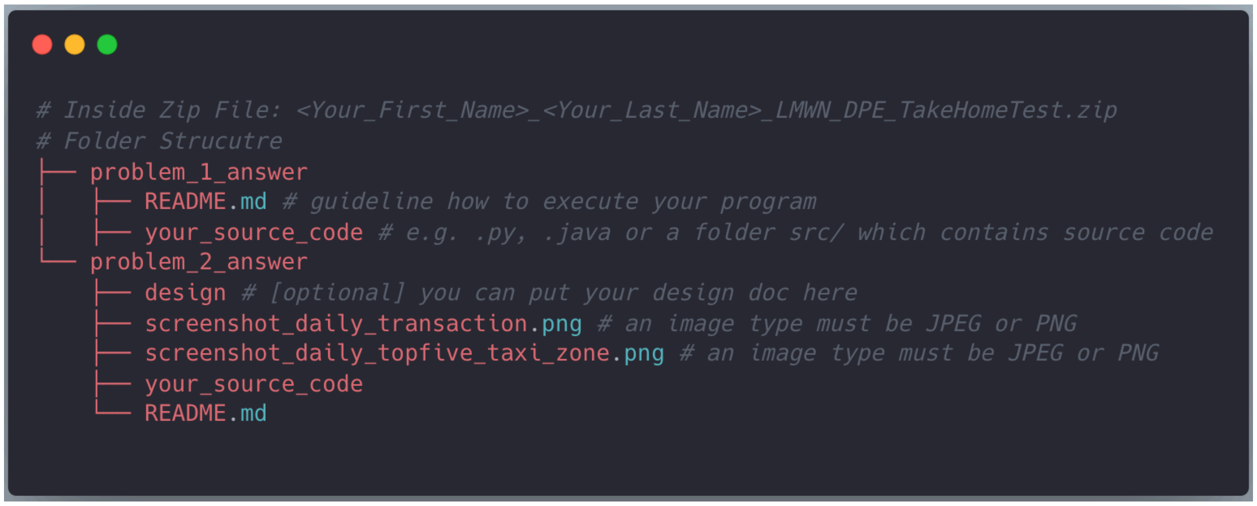

Dataset: High Volume For-Hire Vehicle Trip Records
import data
ETL Workflow Orchestration Tool
1st ETL Job - Calculate daily total transactions
2nd ETL Job - Calculate today top-5 of "TLC Taxi Zone in which the trip began"
*For yellow boxes, you are not allowed to use relational database to process the ETL jobs
save calculation result
Relational Database

1. Please setup one workflow orchestration tool or any tool that can do cronjob / batch scheduling job
2. Please create two data pipelines based on your selected workflow orchestration tool and each pipeline execution must be idempotent
3. You can use any kind of MPP database (Clickhouse, Impala, Doris, etc.) or data processing engines (Spark, Pandas, Polars, DuckDB, etc.) to support your data pipeline's execution. Using the relational database for data pipelining tasks is not allowed e.g. you should not use relational database to process or transform data, it is only a storage
4. You can choose any relation database to store the result (e.g. MySQL, PostgreSQL, etc.)
5. The first data pipeline is to calculate the daily total transactions
    - Please save the result as a table named as daily_transaction with the columns as follows
        - transaction_date = date of transaction (YYYY-MM-DD)
        - total_transactions = number of transactions
        - calculated_at = a timestamp column where the data has been calculated
    - The pipeline must run in daily basis
    - Example of each daily execution
        - Execution of 2023-06-01
            1. calculate total transactions of 2023-05-31
            2. save result into table where the calculated_at is the timestamp of date 2023-06-01 hh:mm:ss
        - Execution of 2023-03-20
            1. calculate total transactions of 2023-03-19
            2. save result into table where the calculated_at is the timestamp of date 2023-03-20 hh:mm:ss

6. The second data pipeline is to calculate the today top-5 of “TLC Taxi Zone in which the trip began”
    - Please save the result as a table named as daily_topfive_taxi_zone with the columns as follows
        - taxi_zone_id = ID of taxi zone
        - rank = rank number (1-5)
        - calculated_at = a timestamp column where the data has been calculated
    - The pipeline must run in daily basis
    - Example of each daily execution
        - Execution of 2023-01-01
            1. collect all data less than 2023-01-01 and find the top-5 of TLC Taxi Zone in which the trip began”
            2. save result into table where the calculated_at is the timestamp of date 2023-01-01 hh:mm:ss
        - Execution of 2023-01-15
            1. collect all data less than 2023-01-14 and find the top-5 of TLC Taxi Zone in which the trip began”
            2. save result into table where the calculated_at is the timestamp of date 2023-01-14 hh:mm:ss
7. Please use the dataset “High Volume For-Hire Vehicle Trip Records” of 2022-2023. All the data is provided in Google Drive
8. Data Dictionary of fhvhv_tripdata can be found in this link

Submission

1. **Problem 1**
    - Please put your source code and README.md file (guideline how to execute your program) in the folder named as problem_1_answer
2. **Problem 2**
    - You are required to take screenshots of query results from both tables (`daily_transaction` and `daily_topfive_taxi_zone`)
        i. please execute query SELECT * FROM <table_name> ORDER BY calculated_at DESC LIMIT 100
    - Please put your source code, screenshots, and necessary documents in the folder named as problem_2_answer
3. Please zip the folder problem_1_answer and problem_2_answer together and named it as <Your_First_Name>_<Your_Last_Name>_LMWN_DPE_TakeHomeTest.zip
4. Please submit the zip file



```
# Inside Zip File: <Your_First_Name>_<Your_Last_Name>_LMWN_DPE_TakeHomeTest.zip
# Folder Strucutre
├── problem_1_answer
│   ├── README.md # guideline how to execute your program
│   ├── your_source_code # e.g. .py, .java or a folder src/ which contains source code
└── problem_2_answer
    ├── design # [optional] you can put your design doc here
    ├── screenshot_daily_transaction.png # an image type must be JPEG or PNG
    ├── screenshot_daily_topfive_taxi_zone.png # an image type must be JPEG or PNG
    ├── your_source_code
    └── README.md
```