Task 1:

Create a FastAPI backend with APIs for performing CRUD operation on the data in SQL tables using payloads and csv. The goal is to design an API that efficiently populates SQL tables based on incoming payloads and csv, adhering to relational database principles.

# **API Functionality:**

## **Insert Data API (Payload):**

* Design an API that inserts data into SQL tables through payloads. The payload structure should allow for flexible insertion of data into relevant tables. For instance, the payload might contain a list of dictionaries, where each dictionary corresponds to a record to be inserted into its respective table, utilizing Many-to-Many relationships.
* Example Scenario: A student can enroll in multiple subjects, and each subject can have a relationship with the teacher's table, indicating that a teacher can teach multiple subjects.

## **Insert Data API (CSV):**

* Implement an API that accepts CSV files as input and populates multiple tables in the database accordingly. Each row in the CSV file can represent data for multiple tables, allowing for efficient bulk insertion.
* Example Scenario: The CSV file may contain information about students, subjects, and their relationships. By parsing the CSV file, the API should populate the corresponding tables with the provided data.

## **Update Data API:**

* Develop an API that updates existing records in the database based on provided payloads. This API should allow for modifications to be made to specific fields of records across multiple tables while ensuring data integrity.
* Example Scenario: Modifying a student's enrollment status in a subject or updating a teacher's contact information across related records.

## **Delete Data API:**

* Create an API that deletes records from the database while preserving referential integrity. Specifically, this API should be able to remove links from Many-to-Many complimentary tables without affecting other related tables.
* Example Scenario: Deleting a student's enrollment in a particular subject without removing the student or subject records themselves, ensuring that other relationships remain intact.

## **Retrieve Data API:**

* Implement an API that retrieves data of a specific student along with all relevant subjects.
* Consider CRUD (Create, Read, Update, Delete) operations and decide whether to encapsulate them within a separate class or distribute them across the backend as per best practices.

# Task 2 (ETL):

You will receive a file containing creator profile details, and creators post details. You have to compute the following things:

| **Metrics** | **Table name** | **Logic** | **Formula** | **Remarks** |
| --- | --- | --- | --- | --- |
| **Active Reach** |  |  | (Total comments + Total Likes + Total Views of last 3 months posts starting from the current date) / Total Posts in last months | 3 months |
| **Followers** |  |  | Total number of followers | 3 months |
| **EMV** |  |  | ( Followers / 1000 \* 2.1) + (comments \* 4.19) + (likes \* 0.09) + (plays \* 0.11) | 3 months |
| **Country** |  |  | country of the creator | 3 months |
| **Average Engagements** |  |  | Sum Engagement (likes+comments+shares+saves) / # of posts | 3 months |
| **Average Video Views** |  |  | Sum Video Views / # of posts | 3 months |
| **Average Story Reach** |  |  | Model / Story Sheet | 3 months  if available on the sheet then use it  else the model |
| Story Engagements |  |  | Average Story Engagements | 3 months |
| Story Views |  |  | Average Story Views | 3 months |
| Saves |  |  | Average Saves | 3 months |
| Likes |  |  | Average Likes | 3 months |
| Comments |  |  | Average Comments |
| Shares |  |  | Average Shares |
| Content |  |  | Paid organic | If post contains @ symbol or keyword اعلان then it is paid post other it is organic |
| Username |  |  |  |  |
| Profile URL |  |  |  |  |
| Media Type |  |  | Photo / Video |  |
| Post (Count) |  |  |  | Total posts in 3 months |

Once you compute the overall metric. Then also compute the same metric by content types as well which is organic and paid.

1. For this task create an end point that takes a profile and posts csvs as input, computes the metrics and saves the final metric in the database.
2. Create another end point in which we can get those metrics based on the username.

## **SQL Schema:**

* Utilise a README.md file to outline the database schema, including Many-to-Many relationships.
* Use a tool like dbdiagram.io to visualise the schema, and include the link to the schema diagram in the README.md. Additionally, export the schema diagram as an image and store it in the docs/ folder within the repository.

# **Deployment:**

1. Create docker and docker compose files to create docker image

# **Additional Guidelines:**

* + Create a Git repository and adhere to best practices of coding and version control.
  + Ensure that the Insert API validates data against the rules defined in the respective tables before populating the data.
  + You can use Students, Teacher, Course, Department examples for creating db schema
  + You are allowed to generate dummy data for insertion in db.
  + When you zip your project, please ensure that the git commit file is within the folder. Your commit history will be reviewed. Please adhere to best practices

# Evaluation Criteria:

## You will be tested on:

1. Project understanding
2. Task distribution/planning
3. Project Structure
4. Database Schema
5. Usage of git
6. Handling of secrets like db credentials
7. How well the readme file is structured to run the code