Data Engineering Test Task (Pre-Internship)

🎯 Task Description

Implement a mini data pipeline that:
1. Loads data from a public REST API.
2. Saves the raw data in file format (JSON or Parquet).
3. Performs data cleaning and enrichment.
4. Loads the processed data into SQLite or PostgreSQL.
5. Outputs basic analytics using SQL queries
6. Generate a .csv/.json report.

Data flow description:
Extract from api -> save locally with transformations -> load to DB-> create analytics queries -> export report

🔧 Technical Requirements

1. Data Acquisition

- Choose any public REST API, for example:
  - https://api.openweathermap.org
  - https://jsonplaceholder.typicode.com
  - https://datausa.io/api

- Implement a Python script to fetch data (using requests or httpx).
- Save raw data in a local S3-like structure:
  /data/raw/yyyy-mm-dd/response.json

2. Data Processing

- Write code to transform raw data:
  - Keep only necessary fields.
  - Rename keys if needed.
  - Clean invalid values.
  - Add a calculated field (e.g., temperature in Celsius).

- Save the cleaned data in Parquet or CSV format:
  /data/processed/yyyy-mm-dd/data.parquet

3. Save to Database

- Create a table in SQLite (or PostgreSQL if preferred).
- Load processed data into the database using SQLAlchemy or psycopg2.

4. Analytics / SQL Queries

- Write 2–3 SQL queries such as:
  - Average value of a field (e.g., temperature)
  - Count of unique records
  - Grouping by category

- Save the results to report.json or report.csv

Tools (Recommended)

- Python 3.10+
- requests / httpx
- pandas
- pyarrow / fastparquet
- SQLAlchemy
- SQLite (or PostgreSQL)

Deliverables

- Python scripts (/src)
- Data: data/raw, data/processed
- Database file: local.db or SQL dump
- README.md with:
  - API used
  - Fields selected and processed
  - Instructions to run the project

Optional (Bonus)

- docker-compose for DB
- Makefile or automation script
- Simple DAG using Prefect or Airflow

Estimated Time

6–8 hours

Evaluation Criteria

- API integration
- Data processing
- Code organization
- Database integration
- Code readability and structure
- Documentation (README)
- SQL query quality
- Understanding of ETL principles

Acceptance criteria

- A file with a description of the implementation steps must be included in the **README** file.
- **Screenshots** proving the application is working:
    - screenshots from the database
    - screenshots of files saved locally
- The **report must be attached**.