dataengineering_final_team7

Code necessary for building Johns Hopkins Data Engineering course final group project (Fall 2025). Main data source is found at https://opendata.maryland.gov/

Data Set 1: Air & Radiation Enforcement Actions

- Under the "Environment" category on Maryland's open data website, there are "enforcement actions" taken by the state of MD to those who violate air and safety regulations. This data gets updated daily and has an API. It contains details about the different sites (companies) and what actions were taken against them for violating air regulations. It also includes county data that can be combined with the other datasets.
  - Sample can be viewed here: Maryland Department of the Environment - Air and Radiation Administration (ARA) Enforcement Actions
  - API endpoint

Data Set 2: Wage Per Job

- The selected dataset "Maryland Average Wage Per Job (in Constant 2024 Dollars): 2014-2024", provides economic data related to employment across Maryland's counties. This is a batch CSV data set, sourced from Maryland Open Data Portal, contains annual information on the average wage per job in the last ten years. This dataset is organized by county and includes columns such as Year, County, and Average Wage (Current Dollars). It allows for analysis of wage trends across different regions in Maryland. This dataset can be combined with other Maryland datasets such as environmental data to explore how environmental or economic patterns may correlate throughout the state. I intend to join this data with other Maryland data based on counties in Maryland.
  - Sample can be viewed here: Maryland Average Wage Per Job (Current Dollars): 2014-2024
  - Static CSV that is stored in repository

Data Set 3: Wage Per Job

- The selected dataset "Maryland Average Wage Per Job (in Constant 2024 Dollars): 2014-2024", provides economic data related to employment across Maryland's counties. This is a batch CSV data set, sourced from Maryland Open Data Portal, contains annual information on the average wage per job in the last ten years. This dataset is organized by county and includes columns such as Year, County, and Average Wage (Current Dollars). It allows for analysis of wage trends across different regions in Maryland. This dataset can be combined with other Maryland datasets such as environmental data to explore how environmental or economic

patterns may correlate throughout the state. I intend to join this data with other Maryland data based on counties in Maryland.
  - Sample can be viewed here: Maryland Department of the Environment - Water and Science Administration (WSA) Enforcement Actions
  - API endpoint

File/Directory Descriptions

- `fetchdata.py` is meant to be an initial script that retrieves data from APIs then processes them into cleaned CSV files that can be uploaded into a database.
- `mdprocessingutils.py` is a module that contains helper functions for processing parts of the Maryland API data before it is ingested into a database.
- `raw_data/` is a intermediate directory used by `fetchdata.py` to store data retrieved from GET requests in their native JSON format.
- `clean_data/` is a directory used by `fetchdata.py` to store processed versions of the data found in `raw_data/`. The data in this directory is in CSV format for the purposes of ingestion into a database.
- `MD_Database.sql` is the creating of the sql database that the cleaned csv files will be uploaded to.

API Service Directories

- `api/` contains the FastAPI service used to query the PostgreSQL database and expose Maryland data through REST endpoints.
- `api/app/` is the main application package for the FastAPI service.
- `api/app/routers/` contains individual route files (e.g., `counties.py`, `wages.py`, `enforcements.py`, `health.py`) that define GET endpoints for each dataset.
- `api/app/reports/` stores summary or combined report logic used by the API.
- `api/app/db.py` includes database connection utilities (SQLAlchemy engine and session creation).
- `api/app/deps.py` provides shared dependencies for the API, including database session injection.
- `api/app/schemas.py` defines Pydantic models representing the expected structure of API responses based on the finalized table schemas.
- `api/Dockerfile` defines how the FastAPI container is built.
- `docker-compose.api.snippet.yml` contains the API service configuration for docker-compose, linking the API to PostgreSQL and Airflow within a shared network.
- `dags/` contains Airflow DAGs used for automated data extraction, trans-

formation, and refresh pipelines.

Set Up Instructions

This project utilzes the postgres container within the `jhu_docker` container. So please make sure you open Docker Desktop first and start that container. Additionally, this project uses the shared network of the `jhu_docker` container. So when you follow the set up instructions, please have the `dataengineering_final_team7/` and it's contents saved within the `jhu_docker/shared/` director.y

Open a new Terminal and follow the instructions. 1. Navigate to where you've saved `jhu_docker/shared/dataengineering_final_team7/` on your host machine. 2. Create your .env file

```
cp api/env.example api/.env
```

3. then overwrite this:

```
cat > api/.env <<'EOF'
FASTAPI_PORT=8088
DB_HOST=postgres
DB_PORT=5432
DB_NAME=md_data
DB_USER=postgres
DB_PASSWORD=postgres
CORS_ORIGINS=*
EOF
```

4. Create Docker shared network

Before building the API container, make sure the shared Docker network exists:

```
docker network create shared
```

If the network already exists, this command will be ignored automatically.

5. Build and run the API container

```
docker compose -f docker-compose.api.yml up -d --build
```

6. Verify the API service

```
docker compose -f docker-compose.api.yml logs -f api
```

6. Open Swagger UI:

- http://localhost:8088/docs

How to create the SQL database and add the data to it

1. Make sure jhu_docker-postgres-1 container is running
2. Run in terminal: docker exec -it jhu_docker-postgres-1 psql -U jhu -d postgres -c "CREATE DATABASE final_project;"
3. Next you will need to run the .sql file to create the schema
4. Copy the file into your Docker container from wherever the file final_project_db.sql is being stored
5. Use the command docker cp '/your/path/final_project_db.sql' jhu_docker-postgres-1:/var/lib/postgresql/data/
6. Verify its container with docker exec -it jhu_docker-postgres-1 ls /var/lib/postgresql/data/
7. Execute the SQL file inside the container: docker exec -it jhu_docker-postgres-1 psql -U jhu -d final_project -f /var/lib/postgresql/data/final_project_db.sql
8. Now you have finished creating the database and schema.
9. To upload the data to the database, make sure you change directories so you are in the working directory/folder with the following folder from our zip file: clean_data.
10. Install package: python3 -m pip install psycopg2-binary pandas
11. Run python script in your terminal from the folder where you have the file stored 'final_project.py' : python3 final_project.py