# REVEST-DE Project Documentation

## Project Overview

This repository contains a solution for the REVEST-DE technical task. The task is divided into two main parts:

1. **Data Warehouse** - A PostgreSQL-based data warehouse to store and query sales data.
2. **Model Deployment** - A REST API that serves product recommendations based on a product ID.

## Part I: Data Warehouse

### Purpose

The data warehouse is designed to store and process sales data for analytical purposes. The data is ingested from a provided CSV file and stored in a PostgreSQL database. The solution allows for querying specific insights like the average order value and monthly revenue.

### Steps:

1. **Ingestion Script**:
   * The sales data from sales.csv is ingested and stored in PostgreSQL.
   * The script ingest\_data.py handles the data ingestion process.
2. **SQL Queries**:
   * Two SQL queries are provided to extract:
     1. Average total value of an order.
     2. Average total revenue for each month.
   * The queries are located in queries.sql.
3. **Export to Parquet**:
   * The data warehouse is exported to a Parquet file for further processing or analysis using the script export\_to\_parquet.py.

### Dependencies:

* Python 3.x
* PostgreSQL
* Libraries:
  + psycopg2 for PostgreSQL interaction
  + pandas for data manipulation
  + pyarrow for Parquet export

## Part II: Model Deployment

### Purpose

This part of the task focuses on deploying a machine learning service that recommends products based on a given product ID. While the recommendation model itself is static (always returning [1, 2, 3]), the implementation demonstrates how to deploy a REST API using Docker and Docker Compose.

### Steps:

1. **API Service**:
   * A simple Flask application (app.py) is created to handle HTTP requests.
   * The API accepts a product ID and responds with a static list of recommended products.
2. **Logging**:
   * A logging mechanism is implemented to store request and response details in a PostgreSQL database (logging.db).
   * The log table contains the following fields: log\_id, timestamp, product\_id, recommended\_product, and log\_type.
3. **Docker & Docker Compose**:
   * Docker is used to containerize both the API service and the PostgreSQL database.
   * docker-compose.yml defines the service and PostgreSQL containers.
4. **Optional Caching Layer**:
   * While not required, a caching layer (e.g., Redis) can be added for performance improvement in a production setup.

### Dependencies:

* Python 3.x
* Flask for API service
* PostgreSQL for logging
* Docker and Docker Compose

## Environment Setup

### 1. Clone the Repository:

git clone https://github.com/Ali-Abuaboud/REVEST-DE.git

cd REVEST-DE

### 2. Docker and Docker Compose:

Ensure that you have Docker and Docker Compose installed on your machine. You can download and install them from the following links:

* [Docker](https://www.docker.com/get-started)
* [Docker Compose](https://docs.docker.com/compose/install/)

### 3. Running the Data Warehouse:

1. **Set up PostgreSQL**:
   * Configure PostgreSQL to create the revest database and set up the required schema.
2. **Run the Data Ingestion Script**:
   * Navigate to data\_warehouse/scripts and run the following command to ingest the sales data into PostgreSQL:
   * python ingest\_data.py
3. **Run the Queries**:
   * Execute the queries in queries.sql to generate insights like average order values and monthly revenue.
4. **Export Data**:
   * Run export\_to\_parquet.py to export the data warehouse to a Parquet file:
   * python export\_to\_parquet.py

### 4. Running the Model Deployment:

1. **Start Docker Compose**:
   * Navigate to model\_deployment/ and use Docker Compose to build and start the services:
   * docker-compose up --build
2. **Test the API**:
   * Once the containers are up, you can test the REST API by sending a POST request to the /recommend endpoint with a product\_id. For example:
   * curl -X POST http://localhost:5000/recommend -d '{"product\_id": "123"}' -H "Content-Type: application/json"
3. **Logging**:
   * The requests and responses are logged to the PostgreSQL database (logging.db), which you can query to see the logs.

## Configurations and Environment Variables

### PostgreSQL (for Data Warehouse and Logging):

Make sure the PostgreSQL database is set up correctly:

* Database name: revest
* User: postgres
* Password: root
* Host: localhost
* Port: 5432

These credentials are used in the scripts (ingest\_data.py, queries.sql, and app.py) and Docker Compose configuration.

## Final Notes

* This project uses Docker to create isolated environments for both the data warehouse and the model deployment.
* If you need to modify configurations, like changing database credentials, you can update the respective files:
  + docker-compose.yml
  + app.py
  + ingest\_data.py
  + export\_to\_parquet.py