TDI PROJECT FOR DATA ENGINEERING WEEK 4

SQLAlchemy Models, PostgreSQL, and Workflow Orchestration

Introduction

Welcome to Week 4! This week, we will be building on the PostgreSQL integration by creating an SQLAlchemy model (`models.py`), transforming and loading data into a PostgreSQL database. You will also orchestrate this process using Prefect and/or Airflow for ETL workflows. The goal is to further extend your project by connecting all the components into a cohesive data engineering pipeline.

PART A: Video Links

Here are some helpful links that will introduce or reinforce the concepts you'll be applying this week:

Requests Library

File Types in DE

Args and Kwargs

Functions in Python

Pandas and Files

argparse Library

Branching in Git

SQLAlchemy

Prefect

Airflow

PostgresSQL

PART B: Assignment Tasks and Directions

In this week's assignment, you will extend your data pipeline from Week 3 to include SQLAlchemy models, transforming the data, and uploading it into a PostgreSQL database. You will also orchestrate this workflow using Prefect and/or Airflow.

File Structure for This Week's Assignment:

```
week4_assignment/
│
├── datalake/                  # Folder to store raw and processed data files
│   ├── raw_data.parquet       # Raw data fetched from the API (e.g., parquet file)
│   └── processed_data.csv     # Processed and cleaned data
│
├── config/                    # Configuration for PostgreSQL connection
│   └── db_config.yaml         # DB connection info (host, port, user, password, etc.)
│
```

```
├── models/                 # Folder to store SQLAlchemy models
│   └── models.py             # SQLAlchemy models for the PostgreSQL tables
│
├── dags/                   # Airflow DAGs folder
│   └── etl_workflow.py       # Airflow ETL workflow script
│
├── src/                    # Source code for Python scripts
│   ├── fetch_data.py         # Script for fetching data from the API
│   ├── transform_data.py     # Script for transforming and cleaning data
│   ├── load_to_postgres.py   # Script for loading data into PostgreSQL using models
│   └── orchestrate.py        # Prefect workflow orchestration
├── .env
├── README.md               # Documentation for the project
├── requirements.txt        # Python dependencies (Prefect, SQLAlchemy, psycopg2, etc.)
```

Use either db_config.yaml or .env

Task 1: Define SQLAlchemy Models (`models.py`)

Objective:
You will define SQLAlchemy models to represent your PostgreSQL database tables, which will make it easier to perform database operations in an object-oriented manner.

Steps:

1. Create a `models.py` file** inside the `models/` directory. Define SQLAlchemy models for storing the data you will fetch from the API from Week 3.

2. What code do you use to define a PostgreSQL table with SQLAlchemy?
Write an `SQLAlchemy` class for your table, such as `SofaScoreData`. Include columns like `id`, `team_name`, `match_date`, `score`, etc.

3. What does the `Base.metadata.create_all()` function do?
Use this function to ensure that all tables are created in your PostgreSQL database.

Task 2: Data Extraction, Transformation, and Loading (ETL)

Objective:
In this task, you will use the API (from Week 3) to fetch the data, clean it, and load the transformed data into your PostgreSQL database using the models you defined in `models.py`.

1. Extract (Fetch Data):
Reuse the API code from Week 3 to collect data from the API. Save the raw data as a Parquet file in the `datalake/` folder.

2. Transform (Clean Data):
Write a Python function that cleans the data, such as filling missing values, binning data, or formatting dates. Save the cleaned data into a CSV file.

3. Load (Upload to PostgreSQL):
Write a function that uses SQLAlchemy models to load the cleaned data into the `data warehouse` table in PostgreSQL. You should use the ORM (Object-Relational Mapper) to insert the data into the table.

Task 3: Workflow Orchestration with Prefect

Objective:
You will use Prefect/Airflow to orchestrate the ETL process, defining tasks for fetching, transforming, and loading the data into PostgreSQL.

Steps:

1. What is a Prefect task?
   Convert your ETL pipeline into tasks using decorators like `@task`.

2. How does Prefect help in managing workflows?
   Use Prefect flows to manage your ETL steps in a sequential manner.

Task 3: Branching and GitHub Collaboration

Objective:
You will implement Git branching strategies and manage your project using GitHub for collaboration.

Steps:

1. Create a `prod`, `staging`, and `dev` branch:
   What is the purpose of each branch in software development?

2. Push your code to GitHub:
   Push all your branches and files to a new GitHub repository.

3. Collaborate on GitHub:
   Invite a collaborator to work on your repository. How do you merge changes from `dev` into `prod`?

PART C - Submission Instructions

You are expected to submit your assignment by the end of the week. Submission will be done via Google Forms. You are encouraged to put your work online to help build your portfolio and show your learning

For Twitter Submission:
- Tag the official TDI page: @TDataImmersed
- Tag Annie @DabereNnamani
- Tag the project coordinator @The_Jonathaan
- Tag @python
- Tag @JOloganj
- Use the hashtag TDI

For LinkedIn Submission:
- Tag the TDI page: @TheDataImmersed
- Tag Annie @AnneNnamani
- Tag @josephologunja

For this assignment, you will be required to submit your work via GitHub.

Push your code to GitHub:
  - Ensure all code is pushed to GitHub, with the appropriate branching (`dev`, `staging`, `prod`).
  - Submit your GitHub repository link and ensure that your `README.md` file explains how to set up the project, run the workflows, and how collaborators can contribute.

PART D - Correction Class

Correction classes will be held every Saturday from 4 pm to 6 pm Nigerian Time on the TDI official Discord or a Google Meet link will be shared before the class.

Good luck with your assignment! We hope you find it both challenging and rewarding. If you have any questions, feel free to reach out to the mentors or the community on the TDI platform.