mardi 25 mars 2025

Data Engineer: technical test

In this test, the candidate is expected to develop an end-to-end data pipeline that begins with the extraction of synthetic CSV files containing overlapping and complementary data—such as patient demographics, visit details, lab results, physician assignments, and medication prescriptions—and continues through data cleaning, transformation, and merging. The candidate should demonstrate proficiency in handling missing values, normalizing data formats, and joining tables based on common keys like patient_id and visit_id. After transforming the data into a consistent and relational format, the candidate must design and deploy a robust PostgreSQL schema with appropriate constraints and indexes, and replicate the process on a Supabase instance. Finally, they should create and execute SQL queries that answer complex business questions, and integrate these queries within Python scripts to filter, aggregate, and visualize the data, ensuring the entire process is automated and well-documented. This comprehensive task assesses their skills in ETL processes, database management, SQL querying, and Python-based data manipulation in a realistic, production-oriented scenario.

Objective

Build an end-to-end pipeline to ingest, clean, transform, and load synthetic clinical study data into both a **standalone** PostgreSQL database and a Supabase instance. Then, use Python to query and filter the data from these databases.

Key expertises that we want to evaluate:

- **ETL Processes:** Extract, clean, transform, and merge multiple data sources.
- **Database Management:** Design table schemas, deploy and manage PostgreSQL and Supabase databases.
- **SQL Proficiency:** Write complex SQL queries to filter and aggregate data.
- **Python Integration:** Create robust and production-ready Python scripts for data handling, database connectivity, and querying.

Data Description

You will work with synthetic tabular data that includes the following columns (sample structure):

- **patient_id:** Unique identifier for each patient.
- **visit_id:** Unique identifier for each visit (if applicable).
- **visit_date:** Date of the clinical visit.
- **diagnosis:** Diagnosis code or description.
- **medication:** Medication details if prescribed.
- **other_fields:** Additional clinical or demographic details (e.g., age, gender, etc.).

Task Parts

Part 1: Data Extraction, Cleaning, and Transformation

1. Data Extraction:
   - Write a Python script that reads the provided CSV files.
   - Load the data into memory using libraries such as `pandas`.
2. Data Cleaning:
   - Handle missing or null values appropriately.
   - Detect and remove duplicate records.
   - Normalize inconsistent formatting (e.g., date formats, text case).
3. Data Transformation & Merging:
   - Merge datasets on common keys (e.g., `patient_id`).
   - Create new derived fields if necessary (e.g., visit frequency, age group classification).
4. Documentation:
   - Clearly comment your code and document your approach in a README file.

Part 2: Database Setup and Deployment

1. PostgreSQL Database:
   - Design a relational schema that accommodates the cleaned data.
   - Write DDL statements (SQL scripts) to create the necessary tables with appropriate data types, constraints, and indexes.
   - Provide a Python pipeline that connects to the PostgreSQL database (using libraries like `psycopg2` or `SQLAlchemy`) and loads the transformed data.
2. Supabase Database:
   - Create a local instance of Supabase.
   - Repeat the table creation and data loading process for Supabase.
   - Document any differences in configuration or setup between PostgreSQL and Supabase.
   - Make a script migrating data from PostgreSQL to Supabase.
3. Automation:
   - Include a script or instructions to automate the data loading process (e.g., a scheduler working once per day).

Part 3: Querying and Filtering with Python

1. SQL Queries:
    - Write several SQL queries that answer typical business questions, such as:
        - Retrieving all visits for a given patient.
        - Filtering patients based on diagnosis or visit date ranges.
        - Aggregating data (e.g., number of visits per month, average visits per patient).
2. Python Integration:
    - Develop Python scripts that connect to both the PostgreSQL and Supabase databases.
    - Execute the SQL queries through Python and fetch the results.
    - Demonstrate how to manipulate and display query results using `pandas` or other libraries.
    - Implement error handling and logging to ensure the scripts are robust.
    - Write testing pipeline (using pytest)

Deliverables

- **Code Repository:** A Git repository containing:
    - Python scripts for the ETL process.
    - SQL scripts for table creation and sample queries.
    - Documentation (README) explaining your design choices, setup instructions, and how to run the code.
- **Deployment Instructions:** Step-by-step instructions on setting up the PostgreSQL and Supabase databases.
- **Testing and Validation:** Example outputs of your queries, either as console prints or exported CSV files demonstrating successful data extraction and query results.