RevStore
(Data & Analytics P0 Project)

Version 1

Table of Contents

Application Overview 3
Core Functional Requirements 3
Standard Functional Scope 5
Definition of Done 5
Submission 6
Non-Functional Expectations 6
Source Data Location 6

Application Overview

The RevStore is a data engineering project aiming to provide transformed data to the Analytics team to derive valuable insights, so that the stakeholders can make informed decisions.

The project includes processing transaction log data and customer data present in JSON format, making transformations so that the data structure is adhering to the business logic. Finally, when data is extracted and transformed it will be loaded to tables in the MySQL database.
The project aims to create a robust ETL pipeline which can be used to load similar JSON data into the MySQLdatabase.

Core Functional Requirements

1. Data Reading:
- Implement Python scripts to read JSON data files for both customer data and transaction logs.

2. Exploratory Data Analysis (EDA):
- Perform EDA to understand the structure, relationships, and patterns within the data.
- Identify key data attributes, missing values, and outliers.

3. Database Schema Design:
- Design a normalized database schema based on the insights gained from EDA.
- Ensure the schema adheres to database normalization principles to avoid redundancy and ensure data integrity.

4. Data Mapping and Transformation:
- Determine which fields from the JSON data will be used in each table.
- Transform the data as needed to fit the database schema, including type conversions and handling of missing values.

5. Data Loading:
- Implement Python scripts to load the transformed data into the MySQL database.
- Ensure data is inserted correctly into the corresponding tables, adhering to the normalized schema.

6. Testing Scripts:
- Create test scripts to fetch and validate records from the tables to ensure data integrity and correctness.

7. Version Control:
- Use Git and GitHub for version control throughout the project.
- Ensure that commits are well-documented and follow best practices for version control.

Standard Functional Scope

1. Framework and API Usage:
- Use appropriate Python libraries and APIs for database operations, such as `mysql-connector-python` or `SQLAlchemy`.
- Ensure that API calls and database interactions follow best practices.

2. Routing and Design Patterns:
- Configure routing centrally in the application for easy management.
- Follow design patterns and best practices, such as the Model-View-Controller (MVC) pattern, for clean and maintainable code.

3. Validation and Error Handling:
- Validate input data types and formats before processing.
- Display user-friendly messages for functional-related errors (e.g., data validation errors) instead of system or SQL error codes.
- Handle exceptions and errors gracefully to ensure robust operation.

4. Logging:
- Implement a logging framework to capture relevant application events and errors.
- Configure log levels using configuration files to allow easy changes without modifying the code.
- Ensure logs are directed to a specified log file for review.

5. Testing:
- Write comprehensive test cases using appropriate testing frameworks (e.g., `unittest` or `pytest`).
- Aim for code coverage of at least 80% to ensure reliability and functionality.

6. Security:
- Protect against SQL injection threats by using parameterized queries.
- Apply CORS restrictions if applicable to secure API endpoints.
- Store secrets and credentials as environment variables using secure credential storage

methods.

7. Coding Standards:
- Follow industry coding standards and conventions for readability and maintainability.
- Develop modular code to enhance reusability and ease of maintenance.
- Manage resource objects (e.g., database connections) properly to prevent resource leaks.
- Apply design patterns and application layering principles (e.g., Business Service, DAO Layer) where applicable.

Definition of Done

- **Data Reading:**
    - JSON data is successfully read and parsed into a usable format.

- **EDA Completion:**
    - EDA is performed, and insights are documented, guiding schema design.

- **Schema Design:**
    - A normalized database schema is designed and reviewed for correctness and completeness.

- **Data Transformation:**
    - Data is mapped and transformed according to the schema requirements.

- **Data Loading:**
    - Transformed data is loaded into MySQL, and data integrity is verified.

- **Testing Scripts:**
    - Test scripts are created to fetch and validate records from the tables, ensuring data accuracy.

- **Functionality:**
    - All core functionalities (data reading, EDA, schema design, transformation, and loading) are implemented and tested.

- **Validation & Error Handling:**
    - Validation checks and error handling mechanisms are in place and functioning as expected.

- Logging:

- Logging is set up, configured, and outputs to the specified log file.

- Testing:

- Test cases are written, and code coverage meets the target of 80%.

- Security:

- SQL injection, CORS restrictions, and secure credential storage are implemented and verified.

- **Version Control:**

- Git and GitHub are used for version control, with well-documented commits and adherence to version control best practices.

- **Coding Standards:**

- Code adheres to industry standards and conventions, with modular development and proper resource management.

- Documentation:

- Documentation for code, schema, and processes is complete and clear.

Submission

1. A test script which fetches 5 records from each table.
2. Sharing the associates' code repo for technical review with:
   - Architecture
   - Data Models
   - ETL documentation

Non-Functional Expectations

- Application development should use version control systems (e.g., Git) to manage the project codebase and facilitate collaboration.
- Application development is supposed to follow the Scrum process.

Source Data Location

Link to the JSON files --- click here.