RevStore Project Documentation

Table of Contents

Application Overview

The RevStore project is designed to build a robust ETL (Extract, Transform, Load) pipeline for processing and transforming JSON data. The primary goal is to deliver clean and structured data to the Analytics team, enabling stakeholders to make data-driven decisions. The application will handle both transaction log data and customer data in JSON format, perform necessary transformations, and load the data into a MySQL database. The project aims to provide a scalable and reusable solution for similar data processing needs in the future.

Core Functional Requirements

1. **Data Reading**:
    - Implement Python scripts to read JSON files containing customer data and transaction logs.
2. **Exploratory Data Analysis (EDA)**:
    - Conduct EDA to understand data structure, relationships, and patterns.
    - Identify key attributes, missing values, and outliers.
3. **Database Schema Design**:
    - Design a normalized schema based on EDA insights.
    - Ensure adherence to normalization principles to prevent redundancy and maintain data integrity.
4. **Data Mapping and Transformation**:
    - Map JSON fields to database tables.
    - Transform data to fit the schema, including type conversions and handling of missing values.
5. **Data Loading**:
    - Develop Python scripts to load transformed data into MySQL tables.

- Ensure data is inserted correctly according to the normalized schema.

6. **Testing Scripts**:
    - Create scripts to validate data integrity and correctness by fetching records from tables.
7. **Version Control**:
    - Utilize Git and GitHub for version control.
    - Ensure commits are well-documented and follow best practices.

Standard Functional Scope

1. **Framework and API Usage**:
    - Use Python libraries like mysql-connector-python or SQLAlchemy for database operations.
    - Follow best practices for API calls and database interactions.
2. **Routing and Design Patterns**:
    - Centralize routing configuration for management ease.
    - Apply design patterns such as MVC for maintainable and clean code.
3. **Validation and Error Handling**:
    - Validate data types and formats before processing.
    - Provide user-friendly error messages and handle exceptions gracefully.
4. **Logging**:
    - Implement a logging framework to capture events and errors.
    - Configure log levels and direct logs to a specific file for review.
5. **Testing**:
    - Develop comprehensive test cases using frameworks like unittest or pytest.
    - Target at least 80% code coverage for reliability.
6. **Security**:
    - Use parameterized queries to prevent SQL injection.
    - Implement CORS restrictions for API endpoints if applicable.
    - Securely manage secrets and credentials through environment variables.
7. **Coding Standards**:
    - Adhere to industry coding standards and conventions.
    - Ensure modular, reusable, and maintainable code.

    - Manage resources properly to avoid leaks and apply design patterns as needed.

Definition of Done

- **Data Reading**: JSON data is read and parsed correctly.
- **EDA Completion**: EDA performed, and schema design insights are documented.
- **Schema Design**: Database schema is designed, reviewed, and approved.
- **Data Transformation**: Data is transformed according to schema requirements.
- **Data Loading**: Data is loaded into MySQL and integrity is validated.
- **Testing Scripts**: Test scripts are created and validate record accuracy.
- **Functionality**: All core functionalities are implemented and tested.
- **Validation & Error Handling**: Validation and error handling mechanisms are in place.
- **Logging**: Logging is configured and outputs to the specified file.
- **Testing**: Test cases are complete, and code coverage is at least 80%.
- **Security**: Security measures (SQL injection prevention, CORS, credential storage) are implemented.
- **Version Control**: Git and GitHub are used for version control with documented commits.
- **Coding Standards**: Code adheres to standards, is modular, and manages resources effectively.
- **Documentation**: Documentation for code, schema, and processes is complete.

Submission

1. Provide a test script to fetch 5 records from each table.
2. Share the code repository for technical review, including:
    - Architecture
    - Data Models
    - ETL Documentation

Non-Functional Expectations

- **Version Control**: Use version control systems (e.g., Git) for code management and collaboration.
- **Development Process**: Follow the Scrum process for application development.

Source Data Location

Link to JSON files