**WarmUp**

1. Python

Please fix or improve the python code.

1. SQL

There’s a mismatch on the query, please fix it.

**Main Test**

**Task 1: Create a Simple Odoo Module**

* **Scenario**: Build a Library Management module that allows users to add LibraryBook with fields for Title, Author, and Price. Add a constraint to ensure the price is not negative.

**Sub-Tasks:**

1. **Version Control Requirement**:
   * Use **Git** to manage the module development, including at least 3 commits: (1) initial setup, (2) model and view creation, (3) adding the constraint.
   * **Pre-requisite**: Push the code to a **GitHub** or **GitLab** repository with a proper commit history.
2. **Unit Test Requirement**:
   * Write unit tests for the LibraryBook model that test the constraint (no negative prices).
   * **Pre-requisite**: Use Odoo’s built-in testing framework to write and run tests.

**Task 2: Extend the LibraryBook Model**

* **Scenario**: Extend the LibraryBook model by adding a Category field and creating a list view. Implement a button that calculates the total number of books per category.

**Sub-Tasks:**

1. **Field Validation Requirement**:
   * Add a validation rule to ensure the Category field is not left blank.
   * **Pre-requisite**: Implement custom validation using Python.
2. **Action Button Logic**:
   * Implement the button logic to display a dynamic message on the form view showing the total number of books in each category (e.g., using message\_post() to post a message on the form).
   * **Pre-requisite**: Use Odoo’s ORM methods to compute totals and trigger the form message.
3. **Performance Requirement**:
   * Optimize the button logic so that it scales efficiently with thousands of book records.
   * **Pre-requisite**: Implement optimized query logic to minimize database load.
4. **Testing Requirement**:
   * Write unit tests to verify the business logic, including the button functionality and validation rules.
   * **Pre-requisite**: Use Odoo’s testing framework to write tests and run them in the test environment.

**Task 3: API Integration & Performance Optimization**

* **Scenario 1**: Integrate the OpenLibrary API to fetch additional book details (ISBN, Publication Year) based on the book title. Store this data in the LibraryBook model.
* **Scenario 2**: Optimize the performance of the LibraryBook list view by refactoring inefficient queries.

**Sub-Tasks (Enrichment):**

1. **API Caching Requirement**:
   * Implement a caching mechanism for API calls to avoid repeated requests for the same book title.
   * **Pre-requisite**: Use Odoo’s caching framework or Python libraries like requests\_cache.
2. **Security Requirement**:
   * Ensure the API integration is secure by handling potential exceptions (e.g., API timeouts or invalid responses) and sanitizing inputs.
   * **Pre-requisite**: Use Odoo’s built-in security mechanisms and Python try-except blocks to handle errors gracefully.
3. **Advanced ORM Optimization**:
   * Optimize the database queries in the list view by using Odoo's **prefetching** and **batch processing** capabilities to reduce the number of queries executed.
   * **Pre-requisite**: Use Odoo’s ORM methods for efficient query handling and ensure it can handle large data sets.
4. **CI/CD Integration**:
   * Integrate the module with a CI/CD pipeline using **Jenkins**. Set up automated testing and deployment for the module.
   * **Pre-requisite**: Create a Jenkins pipeline to run unit tests and deploy the module to a test instance of Odoo.