**Week 2: PL/SQL Assignments Overview**

**1. Control Structures**

* **Scenario 1**: Apply a 1% discount to loan interest rates for customers over 60 years old.
  + **PL/SQL Block**: Use a loop to iterate through all customers, check their age, and update the interest rate.
* **Scenario 2**: Promote customers to VIP status if their balance exceeds $10,000.
  + **PL/SQL Block**: Iterate through customers and set IsVIP to TRUE for eligible customers.
* **Scenario 3**: Send reminders to customers with loans due within 30 days.
  + **PL/SQL Block**: Fetch loans due within the next 30 days and print reminder messages.

**2. Error Handling**

* **Scenario 1**: Handle errors during fund transfers between accounts.
  + **Stored Procedure**: SafeTransferFunds logs errors and rolls back the transaction if an error occurs.
* **Scenario 2**: Manage errors when updating employee salaries.
  + **Stored Procedure**: UpdateSalary handles non-existent employee IDs by logging errors.
* **Scenario 3**: Ensure data integrity when adding a new customer.
  + **Stored Procedure**: AddNewCustomer prevents duplicate entries and logs errors.

**3. Stored Procedures**

* **Scenario 1**: Process monthly interest for savings accounts.
  + **Stored Procedure**: ProcessMonthlyInterest calculates and updates the balance with 1% interest.
* **Scenario 2**: Implement a bonus scheme for employees.
  + **Stored Procedure**: UpdateEmployeeBonus adds a bonus percentage to employees' salaries.
* **Scenario 3**: Enable customers to transfer funds between accounts.
  + **Stored Procedure**: TransferFunds checks sufficient balance before transferring.

**4. Functions**

* **Scenario 1**: Calculate the age of customers.
  + **Function**: CalculateAge returns the customer's age based on their date of birth.
* **Scenario 2**: Compute the monthly installment for a loan.
  + **Function**: CalculateMonthlyInstallment returns the monthly installment based on loan details.
* **Scenario 3**: Check if a customer has sufficient balance.
  + **Function**: HasSufficientBalance returns a boolean indicating if the balance is sufficient.

**Week 2: Spring Core and Maven Assignments**

1. **Configuring a Basic Spring Application**
   * **Scenario**: Develop a web application for library management using Spring Framework.
   * **Steps**: Begin by setting up a Maven project, integrating essential Spring dependencies, and configuring the applicationContext.xml. Define the BookService and BookRepository beans within the context, and subsequently run the application to ensure everything is functioning as expected.
2. **Implementing Dependency Injection**
   * **Scenario**: Efficiently manage the dependencies between BookService and BookRepository.
   * **Steps**: Modify the applicationContext.xml to wire the repository into the service. Conduct tests to validate the injection and ensure seamless communication between components.
3. **Implementing Logging with Spring AOP**
   * **Scenario**: Integrate logging to monitor method execution times, enhancing transparency and performance analysis.
   * **Steps**: Add the Spring AOP dependency to your project, create a LoggingAspect class, enable AspectJ support in the configuration, and test the logging functionality to ensure accurate tracking of method execution.
4. **Creating and Configuring a Maven Project**
   * **Scenario**: Establish a Maven project tailored for the library application.
   * **Steps**: Set up the Maven project structure, incorporate the necessary Spring dependencies, and configure relevant Maven plugins to streamline project management.
5. **Configuring the Spring IoC Container**
   * **Scenario**: Centralize the configuration of beans and dependencies using the Spring IoC container.
   * **Steps**: Define and manage beans within the applicationContext.xml, ensuring all dependencies are well-configured and the application behaves as intended during testing.
6. **Configuring Beans with Annotations**
   * **Scenario**: Optimize bean configuration using Spring annotations for a more streamlined setup.
   * **Steps**: Enable component scanning within the Spring configuration, annotate your classes appropriately, and perform tests to verify that the annotations are correctly managing the bean lifecycle.
7. **Implementing Constructor and Setter Injection**
   * **Scenario**: Leverage both constructor and setter injection techniques to manage dependencies within your Spring application.
   * **Steps**: Update applicationContext.xml to accommodate these injection methods, and validate the setup through testing.
8. **Implementing Basic AOP with Spring**
   * **Scenario**: Introduce basic AOP (Aspect-Oriented Programming) features to modularize cross-cutting concerns like logging.
   * **Steps**: Develop a LoggingAspect, define the necessary advice methods, configure these within Spring, and test to ensure the AOP functionality is integrated correctly.
9. **Creating a Spring Boot Application**
   * **Scenario**: Develop a Spring Boot application to streamline the library management system.
   * **Steps**: Utilize Spring Initializr to bootstrap your project, add essential dependencies, configure application.properties, define your entity models, create a RESTful controller, and execute the application to ensure everything operates smoothly.