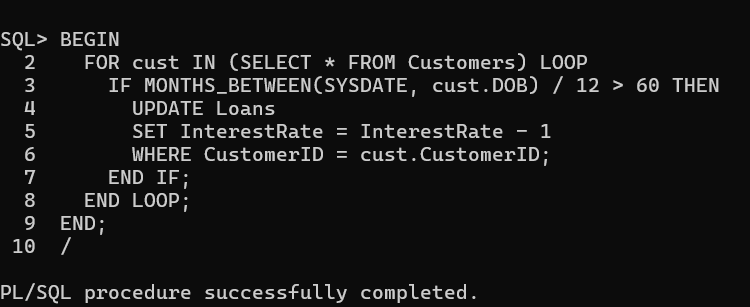
**Exercise 1: Control Structures**

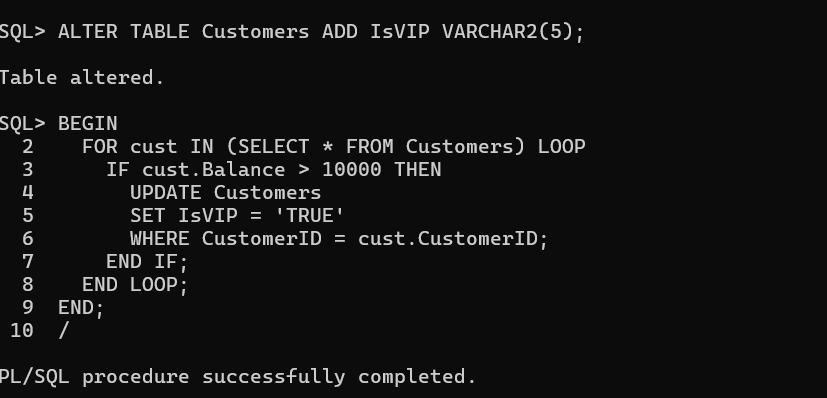
**Scenario 1:** The bank wants to apply a discount to loan interest rates for customers above 60 years old.

Question: Write a PL/SQL block that loops through all customers, checks their age, and if they are above 60, apply a 1% discount to their current loan interest rates.

****

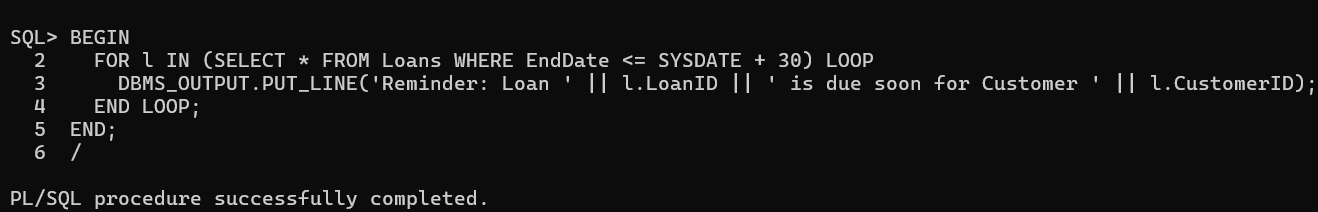
**Scenario 2**: A customer can be promoted to VIP status based on their balance.

Question: Write a PL/SQL block that iterates through all customers and sets a flag Is VIP to TRUE for those with a balance over $10,000.



**Scenario 3:** The bank wants to send reminders to customers whose loans are due within the next 30 days.

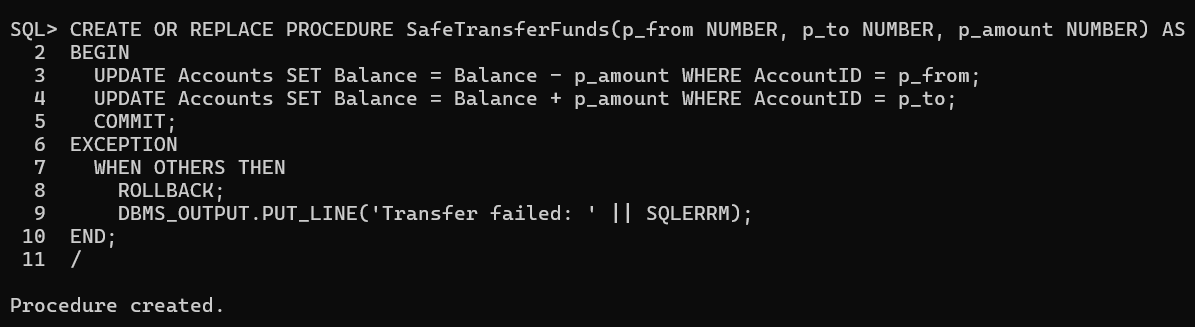
Question: Write a PL/SQL block that fetches all loans due in the next 30 days and prints a reminder message for each customer.



**Exercise 2: Error Handling**

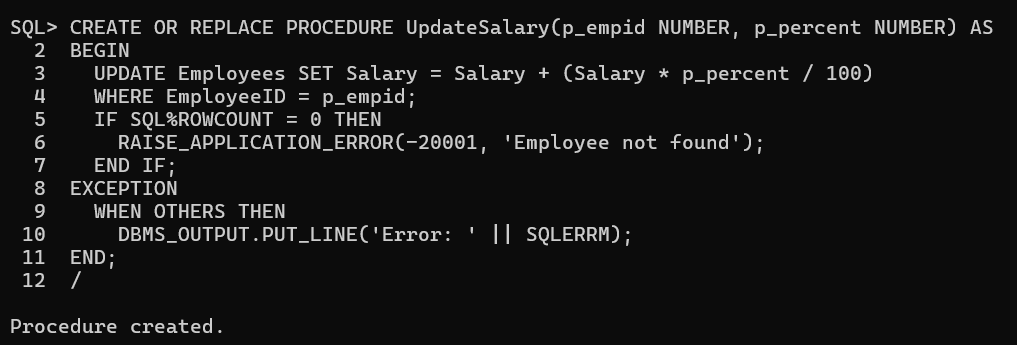
**Scenario 1:** Handle exceptions during fund transfers between accounts.

Question: Write a stored procedure SafeTransferFunds that transfers funds between two accounts. Ensure that if any error occurs (e.g., insufficient funds), an appropriate error message is logged and the transaction is rolled back.

****

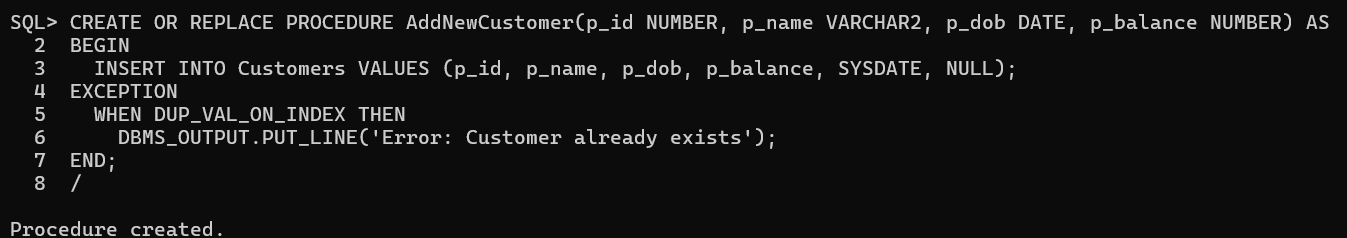
**Scenario 2:** Manage errors when updating employee salaries.

Question: Write a stored procedure UpdateSalary that increases the salary of an employee by a given percentage. If the employee ID does not exist, handle the exception and log an error message.

****

**Scenario 3:** Ensure data integrity when adding a new customer.

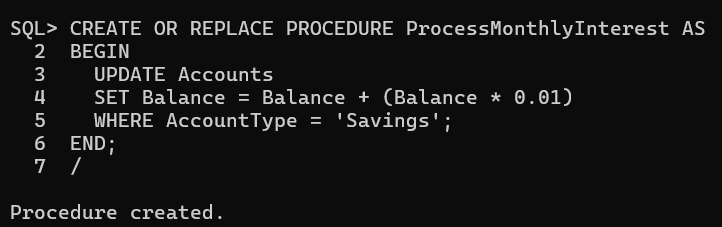
Question: Write a stored procedure AddNewCustomer that inserts a new customer into the Customers table. If a customer with the same ID already exists, handle the exception by logging an error and preventing the insertion.



**Exercise 3: Stored Procedures**

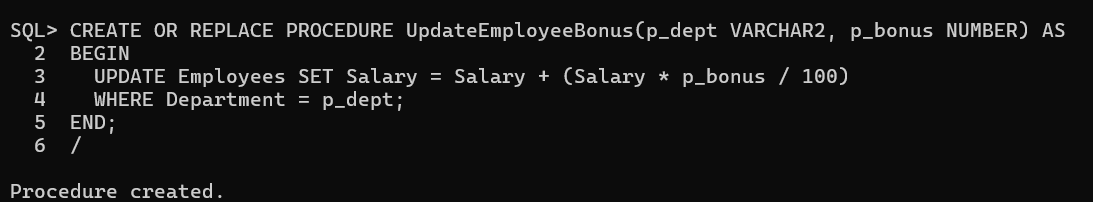
**Scenario 1:** The bank needs to process monthly interest for all savings accounts.

Question: Write a stored procedure ProcessMonthlyInterest that calculates and updates the balance of all savings accounts by applying an interest rate of 1% to the current balance**.**

****

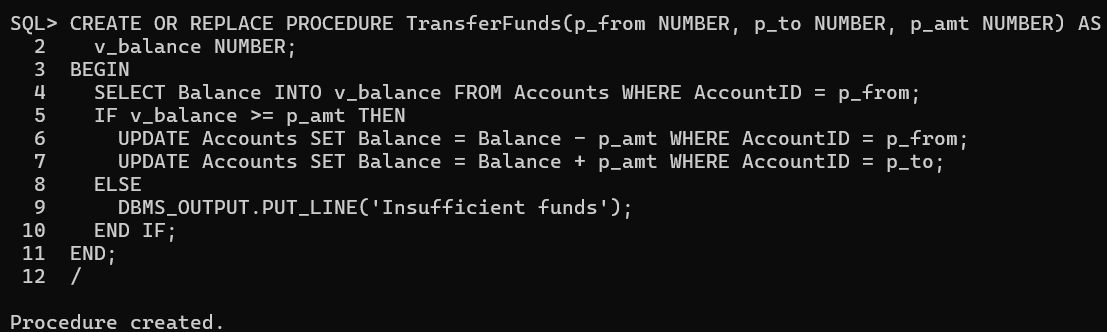
**Scenario 2:** The bank wants to implement a bonus scheme for employees based on their performance.

Question: Write a stored procedure UpdateEmployeeBonus that updates the salary of employees in a given department by adding a bonus percentage passed as a parameter.

****

**Scenario 3**: Customers should be able to transfer funds between their accounts.

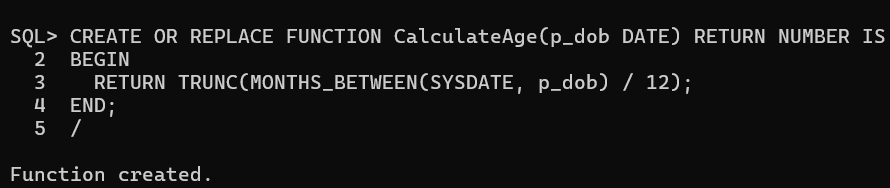
Question: Write a stored procedure TransferFunds that transfers a specified amount from one account to another, checking that the source account has sufficient balance before making the transfer.



**Exercise 4: Functions**

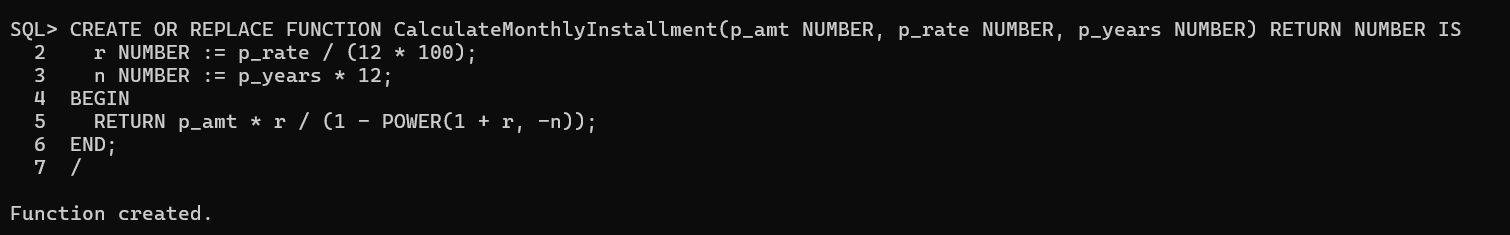
**Scenario 1**: Calculate the age of customers for eligibility checks.

Question: Write a function CalculateAge that takes a customer's date of birth as input and returns their age in years.



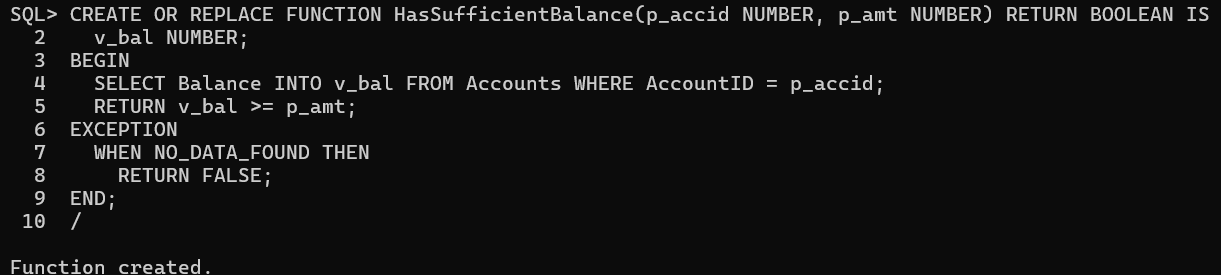
**Scenario 2:** The bank needs to compute the monthly installment for a loan.

Question: Write a function CalculateMonthlyInstallment that takes the loan amount, interest rate, and loan duration in years as input and returns the monthly installment amount.



**Scenario 3:** Check if a customer has sufficient balance before making a transaction.

Question: Write a function HasSufficientBalance that takes an account ID and an amount as input and returns a boolean indicating whether the account has at least the specified amount.



**Exercise 5: Triggers**

**Scenario 1:** Automatically update the last modified date when a customer's record is updated.

Question: Write a trigger UpdateCustomerLastModified that updates the LastModified column of the Customers table to the current date whenever a customer's record is updated.

Solution:

CREATE OR REPLACE TRIGGER UpdateCustomerLastModified

BEFORE UPDATE ON Customers

FOR EACH ROW

BEGIN

:NEW.LastModified := SYSDATE;

END;

/

**Scenario 2:** Maintain an audit log for all transactions.

Question: Write a trigger LogTransaction that inserts a record into an AuditLog table whenever a transaction is inserted into the Transactions table.

Solution:

CREATE OR REPLACE TRIGGER LogTransaction

AFTER INSERT ON Transactions

FOR EACH ROW

BEGIN

INSERT INTO AuditLog (AccountID, Action, ActionDate)

VALUES (:NEW.AccountID, 'Transaction', SYSDATE);

END;

/

**Scenario 3:** Enforce business rules on deposits and withdrawals.

Question: Write a trigger CheckTransactionRules that ensures withdrawals do not exceed the balance and deposits are positive before inserting a record into the Transactions table.

Solution:

CREATE OR REPLACE TRIGGER CheckTransactionRules

BEFORE INSERT ON Transactions

FOR EACH ROW

DECLARE

v\_balance NUMBER;

BEGIN

SELECT Balance INTO v\_balance FROM Accounts WHERE AccountID = :NEW.AccountID;

IF :NEW.TransactionType = 'Withdrawal' AND :NEW.Amount > v\_balance THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Withdrawal amount exceeds balance.');

ELSIF :NEW.TransactionType = 'Deposit' AND :NEW.Amount <= 0 THEN

RAISE\_APPLICATION\_ERROR(-20002, 'Deposit amount must be positive.');

END IF;

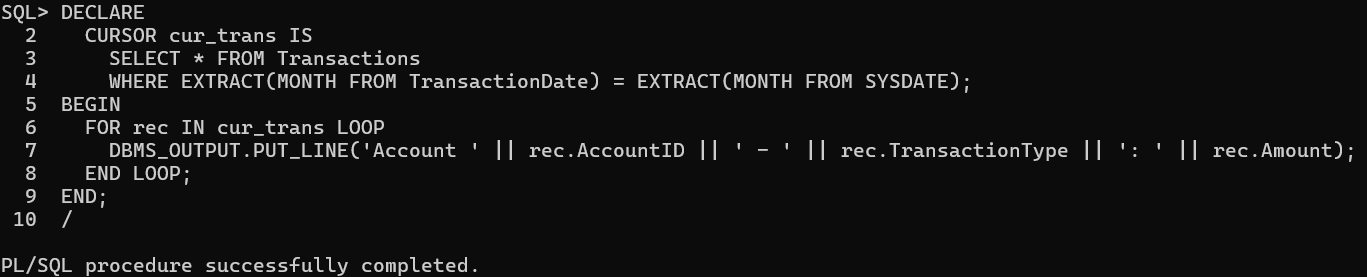
END;

/

**Exercise 6: Cursors**

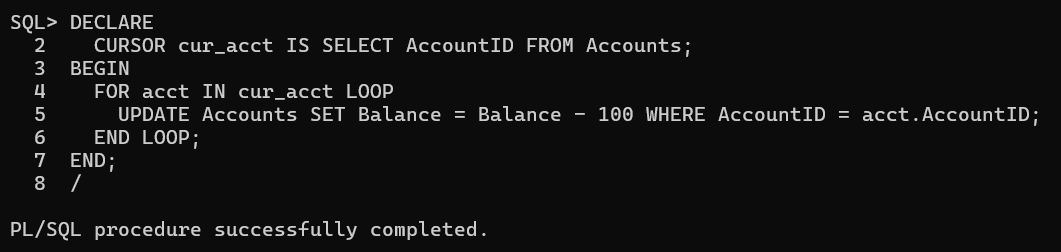
**Scenario 1:** Generate monthly statements for all customers.

Question**:** Write a PL/SQL block using an explicit cursor GenerateMonthlyStatements that retrieves all transactions for the current month and prints a statement for each customer.



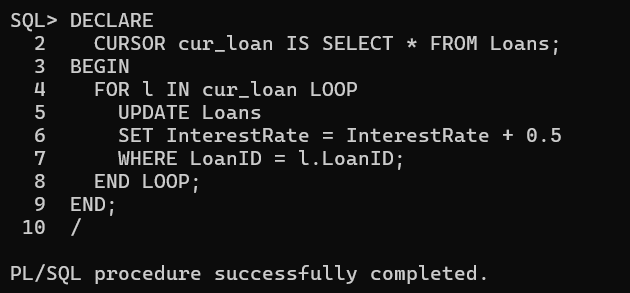
**Scenario 2:** Apply annual fee to all accounts.

Question**:** Write a PL/SQL block using an explicit cursor ApplyAnnualFee that deducts an annual maintenance fee from the balance of all accounts.



**Scenario 3:** Update the interest rate for all loans based on a new policy.

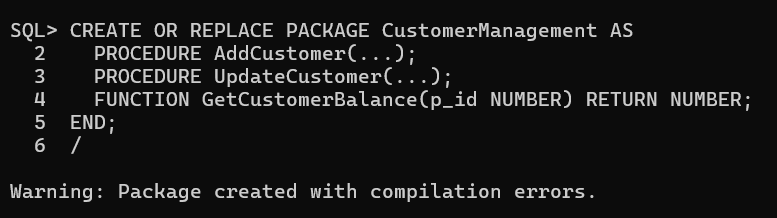
Question**:** Write a PL/SQL block using an explicit cursor UpdateLoanInterestRates that fetches all loans and updates their interest rates based on the new policy.



**Exercise 7: Packages**

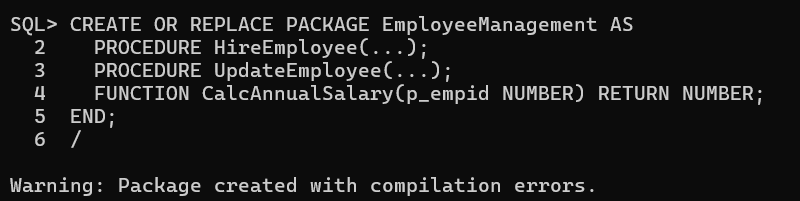
**Scenario 1:** Group all customer-related procedures and functions into a package.

Question**:** Create a package CustomerManagement with procedures for adding a new customer, updating customer details, and a function to get customer balance.



**Scenario 2:** Create a package to manage employee data.

Question: Write a package EmployeeManagement with procedures to hire new employees, update employee details, and a function to calculate annual salary.



**Scenario 3:** Group all account-related operations into a package.

Question: Create a package AccountOperations with procedures for opening a new account, closing an account, and a function to get the total balance of a customer across all accounts.

