**PL/SQL programming**

**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

**Code:**

DELIMITER $$

CREATE PROCEDURE ApplyInterestDiscountVerbose()

BEGIN

DECLARE done INT DEFAULT 0;

DECLARE cust\_id INT;

DECLARE cust\_name VARCHAR(100);

DECLARE cust\_age INT;

DECLARE loan\_id INT;

DECLARE old\_interest DECIMAL(5,2);

DECLARE new\_interest DECIMAL(5,2);

DECLARE cur CURSOR FOR

SELECT c.CustomerID, c.Name, TIMESTAMPDIFF(YEAR, c.DOB, CURDATE()) AS Age

FROM Customers c;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;

OPEN cur;

read\_loop: LOOP

FETCH cur INTO cust\_id, cust\_name, cust\_age;

IF done THEN

LEAVE read\_loop;

END IF;

IF cust\_age > 60 THEN

-- Direct inner SELECT without cursor

SELECT LoanID, InterestRate INTO loan\_id, old\_interest

FROM Loans

WHERE CustomerID = cust\_id

LIMIT 1;

IF loan\_id IS NOT NULL THEN

UPDATE Loans

SET InterestRate = InterestRate - 1

WHERE LoanID = loan\_id;

SELECT InterestRate INTO new\_interest FROM Loans WHERE LoanID = loan\_id;

SELECT CONCAT(

'Updated Customer ID: ', cust\_id,

', Name: ', cust\_name,

', Old Interest Rate: ', old\_interest,

', New Interest Rate: ', new\_interest

) AS Message;

END IF;

END IF;

END LOOP;

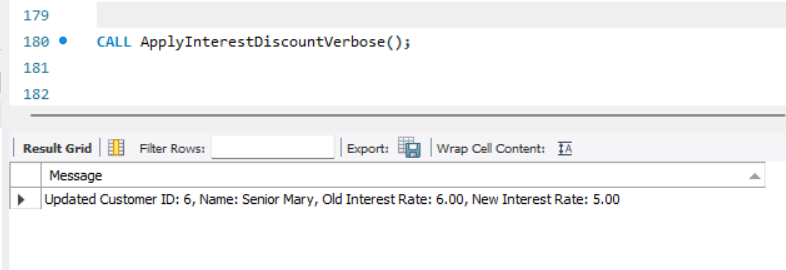
CLOSE cur;

END$$

DELIMITER ;

CALL ApplyInterestDiscountVerbose();

**Output:**

****

**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 IsVIP to TRUE for those with a balance over $10,000.

**Code:**

DELIMITER $$

CREATE PROCEDURE PromoteVIPCustomers()

BEGIN

DECLARE done INT DEFAULT 0;

DECLARE cust\_id INT;

DECLARE cust\_name VARCHAR(100);

DECLARE cust\_balance DECIMAL(10,2);

DECLARE cur CURSOR FOR

SELECT CustomerID, Name, Balance FROM Customers WHERE Balance > 10000;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;

OPEN cur;

read\_loop: LOOP

FETCH cur INTO cust\_id, cust\_name, cust\_balance;

IF done THEN

LEAVE read\_loop;

END IF;

UPDATE Customers

SET IsVIP = 'TRUE'

WHERE CustomerID = cust\_id;

SELECT CONCAT(

'Promoted to VIP -> Customer ID: ', cust\_id,

', Name: ', cust\_name,

', Balance: ', cust\_balance

) AS Message;

END LOOP;

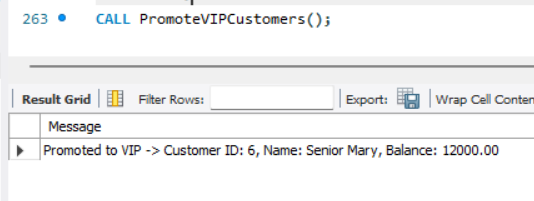
CLOSE cur;

END$$

DELIMITER ;

CALL PromoteVIPCustomers();

**Output:**

****

**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.

**Code:**

DELIMITER $$

CREATE PROCEDURE SendLoanReminders()

BEGIN

DECLARE done INT DEFAULT 0;

DECLARE cust\_id INT;

DECLARE cust\_name VARCHAR(100);

DECLARE loan\_id INT;

DECLARE due\_date DATE;

DECLARE days\_left INT;

DECLARE cur CURSOR FOR

SELECT l.LoanID, l.CustomerID, l.EndDate, c.Name

FROM Loans l

JOIN Customers c ON l.CustomerID = c.CustomerID

WHERE l.EndDate BETWEEN CURDATE() AND DATE\_ADD(CURDATE(), INTERVAL 30 DAY);

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;

OPEN cur;

read\_loop: LOOP

FETCH cur INTO loan\_id, cust\_id, due\_date, cust\_name;

IF done THEN

LEAVE read\_loop;

END IF;

SET days\_left = DATEDIFF(due\_date, CURDATE());

-- Display reminder message

SELECT CONCAT(

'Reminder: Loan ID ', loan\_id,

' for Customer ID ', cust\_id,

' (', cust\_name, ') is due on ',

DATE\_FORMAT(due\_date, '%Y-%m-%d'),

' (', days\_left, ' days left).'

) AS Message;

END LOOP;

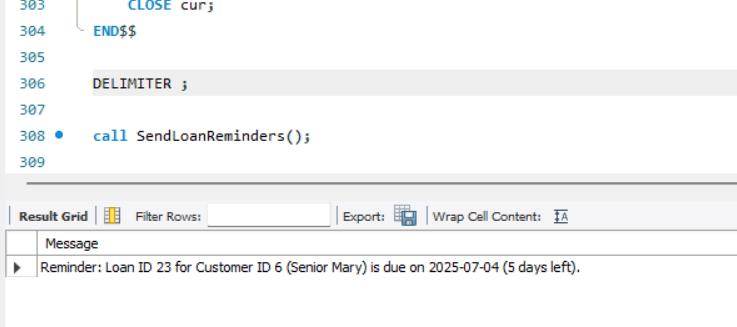
CLOSE cur;

END$$

DELIMITER ;

call SendLoanReminders();

**Output:**



**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.

**Code:**

DELIMITER $$

CREATE PROCEDURE ProcessMonthlyInterest()

BEGIN

DECLARE done INT DEFAULT 0;

DECLARE acc\_id INT;

DECLARE acc\_balance DECIMAL(10,2);

DECLARE new\_balance DECIMAL(10,2);

DECLARE cur CURSOR FOR

SELECT AccountID, Balance FROM Accounts WHERE AccountType = 'Savings';

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;

OPEN cur;

read\_loop: LOOP

FETCH cur INTO acc\_id, acc\_balance;

IF done THEN

LEAVE read\_loop;

END IF;

SET new\_balance = acc\_balance \* 1.01;

UPDATE Accounts SET Balance = new\_balance, LastModified = CURDATE() WHERE AccountID = acc\_id;

-- Display the updated row

SELECT CONCAT(

'Interest Applied -> AccountID: ', acc\_id,

', Old Balance: ', acc\_balance,

', New Balance: ', new\_balance

) AS Message;

END LOOP;

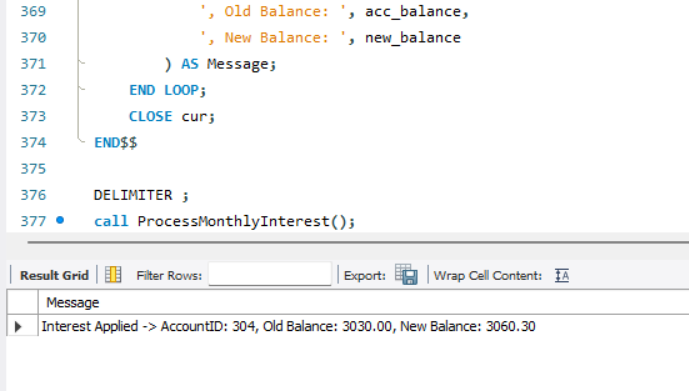
CLOSE cur;

END$$

DELIMITER ;

call ProcessMonthlyInterest();

**Output:**



**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.

**Code:**

DELIMITER $$

CREATE PROCEDURE UpdateEmployeeBonus(IN dept\_name VARCHAR(50), IN bonus\_percent DECIMAL(5,2))

BEGIN

DECLARE done INT DEFAULT 0;

DECLARE emp\_id INT;

DECLARE emp\_name VARCHAR(100);

DECLARE old\_salary DECIMAL(10,2);

DECLARE new\_salary DECIMAL(10,2);

DECLARE cur CURSOR FOR

SELECT EmployeeID, Name, Salary FROM Employees WHERE Department = dept\_name;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;

OPEN cur;

read\_loop: LOOP

FETCH cur INTO emp\_id, emp\_name, old\_salary;

IF done THEN

LEAVE read\_loop;

END IF;

SET new\_salary = old\_salary \* (1 + bonus\_percent / 100);

UPDATE Employees SET Salary = new\_salary WHERE EmployeeID = emp\_id;

SELECT CONCAT(

'Bonus Applied: EmployeeID: ', emp\_id,

', Name: ', emp\_name,

', Old Salary: ', old\_salary,

', New Salary: ', new\_salary

) AS Message;

END LOOP;

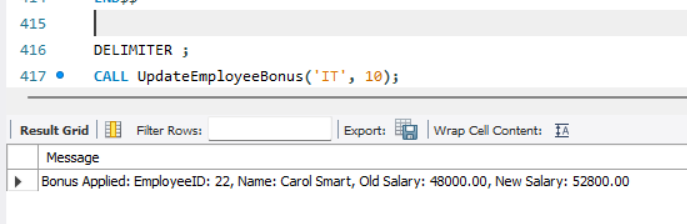
CLOSE cur;

END$$

DELIMITER ;

CALL UpdateEmployeeBonus('IT', 10);

**Output:**

****

**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.

**Code:**

DELIMITER $$

CREATE PROCEDURE TransferFunds(

IN from\_account\_id INT,

IN to\_account\_id INT,

IN transfer\_amount DECIMAL(10,2)

)

BEGIN

DECLARE from\_balance DECIMAL(10,2);

DECLARE to\_balance DECIMAL(10,2);

-- Fetch current balances

SELECT Balance INTO from\_balance FROM Accounts WHERE AccountID = from\_account\_id;

SELECT Balance INTO to\_balance FROM Accounts WHERE AccountID = to\_account\_id;

-- Check sufficient balance

IF from\_balance >= transfer\_amount THEN

-- Update balances

UPDATE Accounts SET Balance = Balance - transfer\_amount, LastModified = CURDATE() WHERE AccountID = from\_account\_id;

UPDATE Accounts SET Balance = Balance + transfer\_amount, LastModified = CURDATE() WHERE AccountID = to\_account\_id;

SELECT CONCAT(

'Transfer Successful: ', transfer\_amount,

' transferred from AccountID ', from\_account\_id,

' to AccountID ', to\_account\_id,

'. New From Balance: ', from\_balance - transfer\_amount,

', New To Balance: ', to\_balance + transfer\_amount

) AS Message;

ELSE

SELECT CONCAT(

'Transfer Failed: Insufficient funds in AccountID ', from\_account\_id,

'. Available: ', from\_balance,

', Required: ', transfer\_amount

) AS Message;

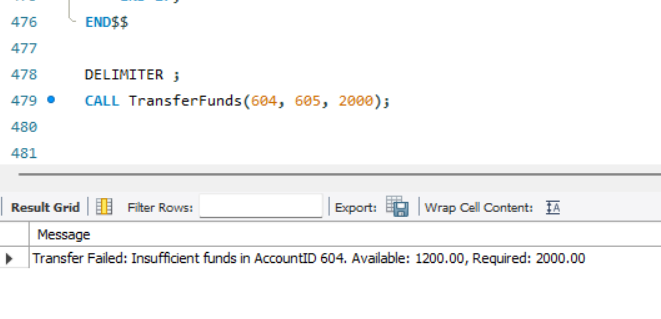
END IF;

END$$

DELIMITER ;

CALL TransferFunds(604, 605, 2000);

**Output:**

****