**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.

**Answer:**

**PROGRAM:**

DELIMITER $$

CREATE PROCEDURE ApplySeniorDiscount()

BEGIN

DECLARE done INT DEFAULT FALSE;

DECLARE v\_customer\_id INT;

DECLARE v\_loan\_id INT;

DECLARE v\_interest\_rate DECIMAL(5,2);

DECLARE cur CURSOR FOR

SELECT c.CustomerID, l.LoanID, l.InterestRate

FROM Customers c

JOIN Loans l ON c.CustomerID = l.CustomerID

WHERE TIMESTAMPDIFF(YEAR, c.DOB, CURDATE()) > 60;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

OPEN cur;

read\_loop: LOOP

FETCH cur INTO v\_customer\_id, v\_loan\_id, v\_interest\_rate;

IF done THEN

LEAVE read\_loop;

END IF;

UPDATE Loans

SET InterestRate = InterestRate - 1.00

WHERE LoanID = v\_loan\_id;

END LOOP;

CLOSE cur;

END$$

DELIMITER ;

CALL ApplySeniorDiscount();

**OUTPUT :**

Before PROCEDURE CALL:

Table : Customers

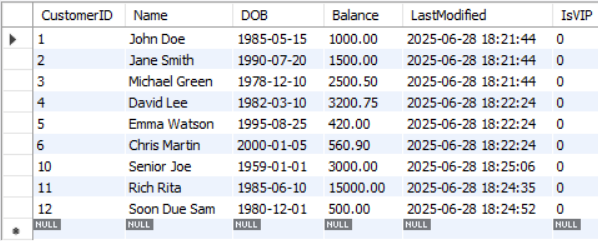
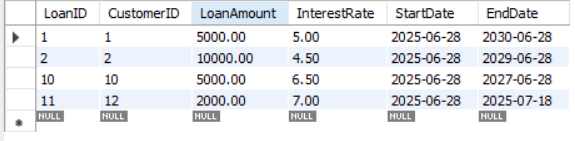


Table : Loans



AFTER PROCEDURE CALL:

Table : Customers

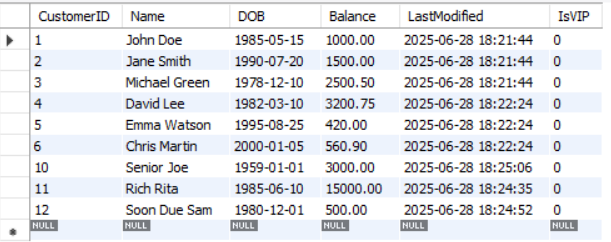
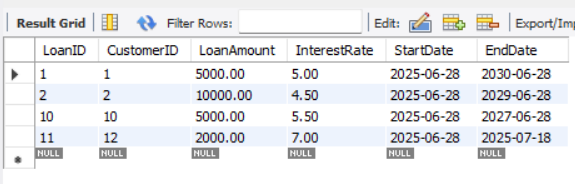


Table : Loans



**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.

**Answer:**

**PROGRAM:**

DELIMITER $$

CREATE PROCEDURE MarkVIPCustomers()

BEGIN

UPDATE Customers

SET IsVIP = TRUE

WHERE Balance > 10000;

END$$

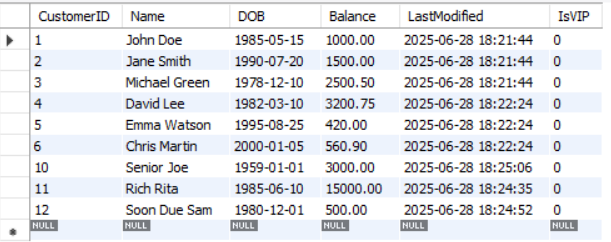
DELIMITER ;

CALL MarkVIPCustomers();

**OUTPUT:**

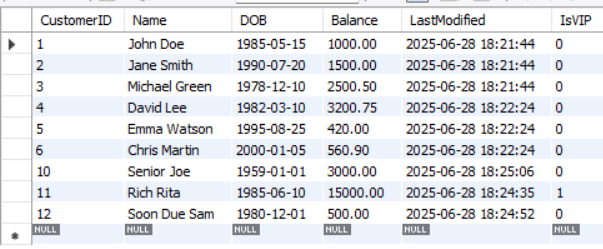
BEFORE CALLING THE PROCEDURE:

Table : Customers



AFTER CALLING THE PROCEDURE:

Table : Customers



**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.

**Answer:**

**PROGRAM:**

DELIMITER $$

CREATE PROCEDURE SendLoanReminders()

BEGIN

SELECT

CONCAT('Reminder: Loan ID ', l.LoanID,

' for customer "', c.Name,

'" is due on ', DATE\_FORMAT(l.EndDate, '%d-%b-%Y')) AS Reminder

FROM Loans l

JOIN Customers c ON c.CustomerID = l.CustomerID

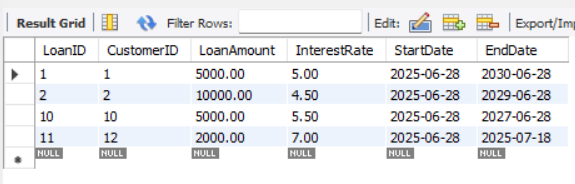
WHERE l.EndDate BETWEEN CURDATE() AND DATE\_ADD(CURDATE(), INTERVAL 30 DAY);

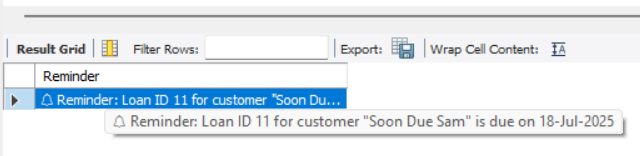
END$$

DELIMITER ;

**OUTPUT:**

Table : Loans



****

**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.

**Answer:**

**PROGRAM:**

DELIMITER $$

CREATE PROCEDURE ProcessMonthlyInterest()

BEGIN

UPDATE Accounts

SET Balance = Balance \* 1.01

WHERE AccountType = 'Savings';

END$$

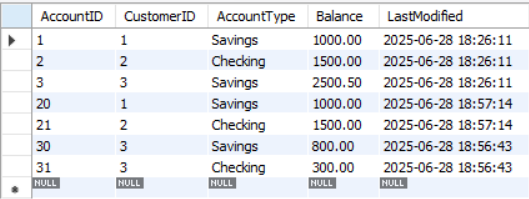
DELIMITER ;

CALL ProcessMonthlyInterest();

**OUTPUT:**

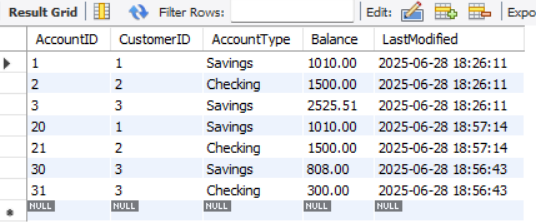
BEFORE CALLING THE PROCEDURE:

Table : Accounts



AFTER CALLING THE PROCEDURE:

Table : Accounts



**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.

**Answer:**

**PROGRAM:**

DELIMITER $$

CREATE PROCEDURE UpdateEmployeeBonus(

IN deptName VARCHAR(50),

IN bonusPercent DECIMAL(5,2)

)

BEGIN

UPDATE Employees

SET Salary = Salary + (Salary \* bonusPercent / 100)

WHERE Department = deptName;

END$$

DELIMITER ;

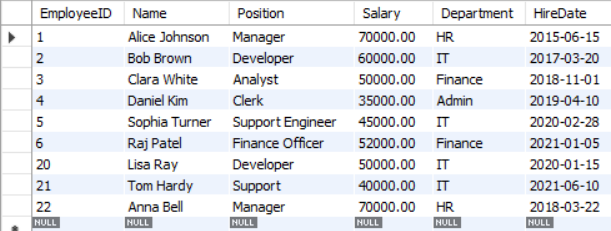
-- for example, I am adding 10 % bonus to employees from IT Department

CALL UpdateEmployeeBonus('IT', 10);

**OUTPUT:**

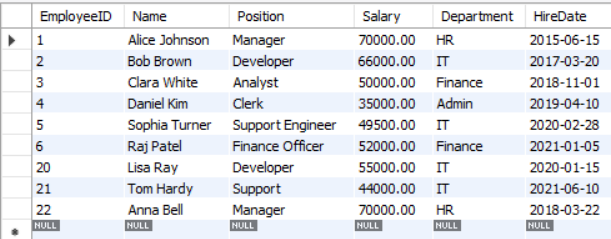
BEFORE CALLING THE PROCEDURE

Table : Employees



AFTER CALLING THE PROCEDURE

Table : Employees



**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.

**Answer:**

**PROGRAM:**

DELIMITER $$

CREATE PROCEDURE TransferFunds(

IN fromAccountID INT,

IN toAccountID INT,

IN transferAmount DECIMAL(10,2)

)

BEGIN

DECLARE fromBalance DECIMAL(10,2);

-- Check source account balance

SELECT Balance INTO fromBalance

FROM Accounts

WHERE AccountID = fromAccountID;

IF fromBalance >= transferAmount THEN

-- Deduct from source

UPDATE Accounts

SET Balance = Balance - transferAmount

WHERE AccountID = fromAccountID;

-- Add to destination

UPDATE Accounts

SET Balance = Balance + transferAmount

WHERE AccountID = toAccountID;

ELSE

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Insufficient funds in source account';

END IF;

END$$

DELIMITER ;

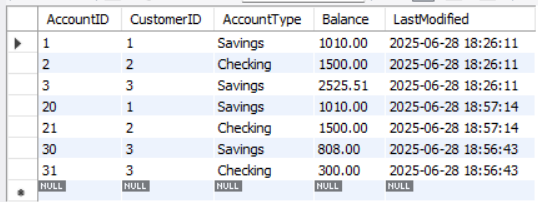
-- for example, transferring funds from savings (30) to checking (31) account

CALL TransferFunds(30, 31, 200.00);

**OUTPUT:**

BEFORE CALLING THE PROCEDURE

Table : Accounts



AFTER CALLING THE PROCEDURE

Table : Accounts

