**MODULE 3: 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.

**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.

**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

**SOLUTION:**

**CODE:**

CREATE DATABASE BankDB1;

use BankDB1;

CREATE TABLE Customers (

CustomerID INT PRIMARY KEY,

Name VARCHAR(100),

DOB DATE,

Balance DECIMAL(10,2),

LastModified DATE,

IsVIP BOOLEAN DEFAULT FALSE

);

CREATE TABLE Loans (

LoanID INT PRIMARY KEY,

CustomerID INT,

LoanAmount DECIMAL(10,2),

InterestRate DECIMAL(5,2),

StartDate DATE,

EndDate DATE,

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

INSERT INTO Customers VALUES

(1, 'Sankha', '1950-05-20', 15000, CURDATE(), FALSE),

(2, 'Luffy', '1985-08-15', 8000, CURDATE(), FALSE),

(3, 'Naruto', '1960-04-10', 12000, CURDATE(), FALSE);

INSERT INTO Loans VALUES

(101, 1, 100000, 8.5, '2020-01-01', '2025-07-10'),

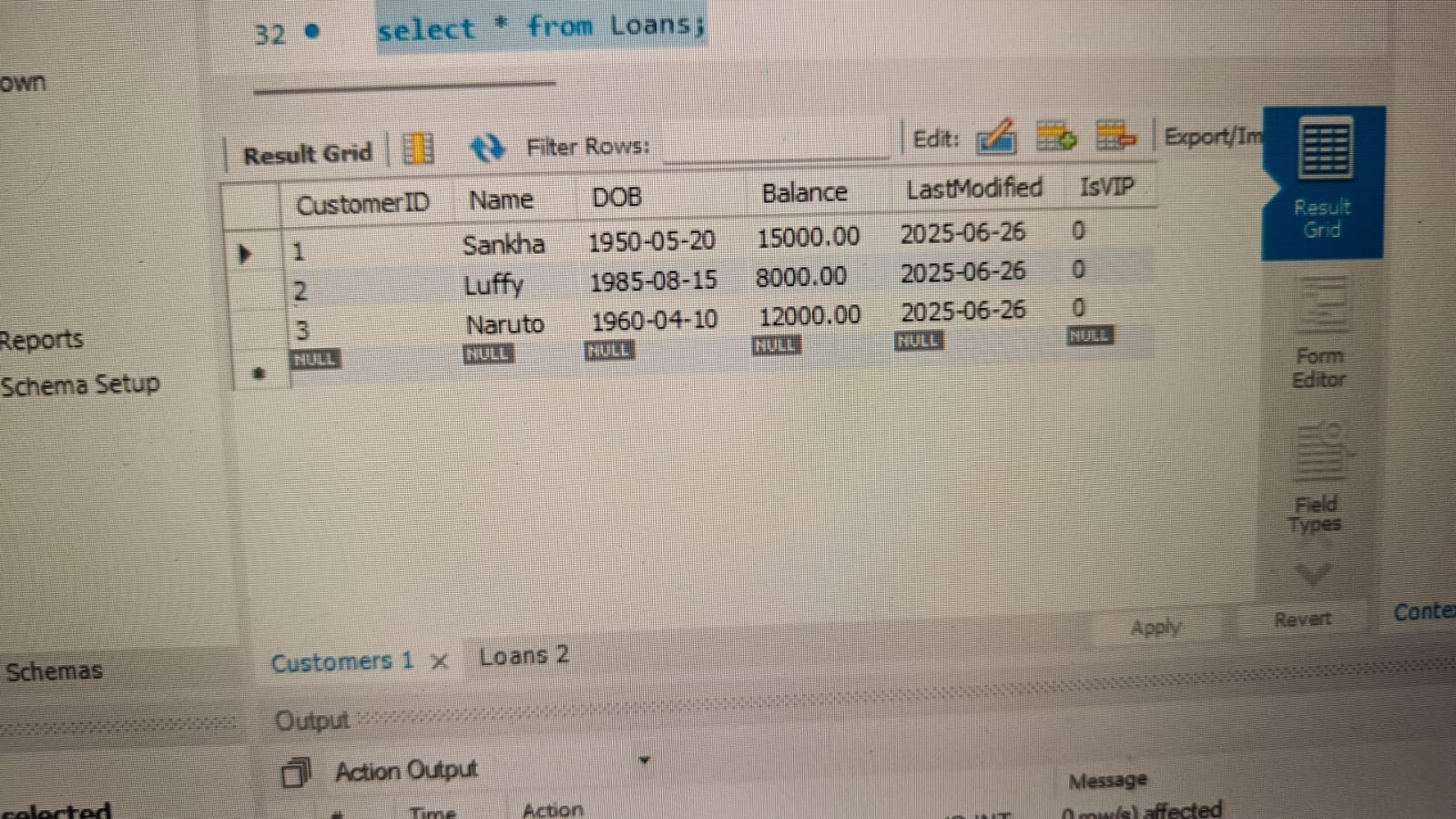
(102, 2, 50000, 7.5, '2021-03-01', '2025-08-01'),

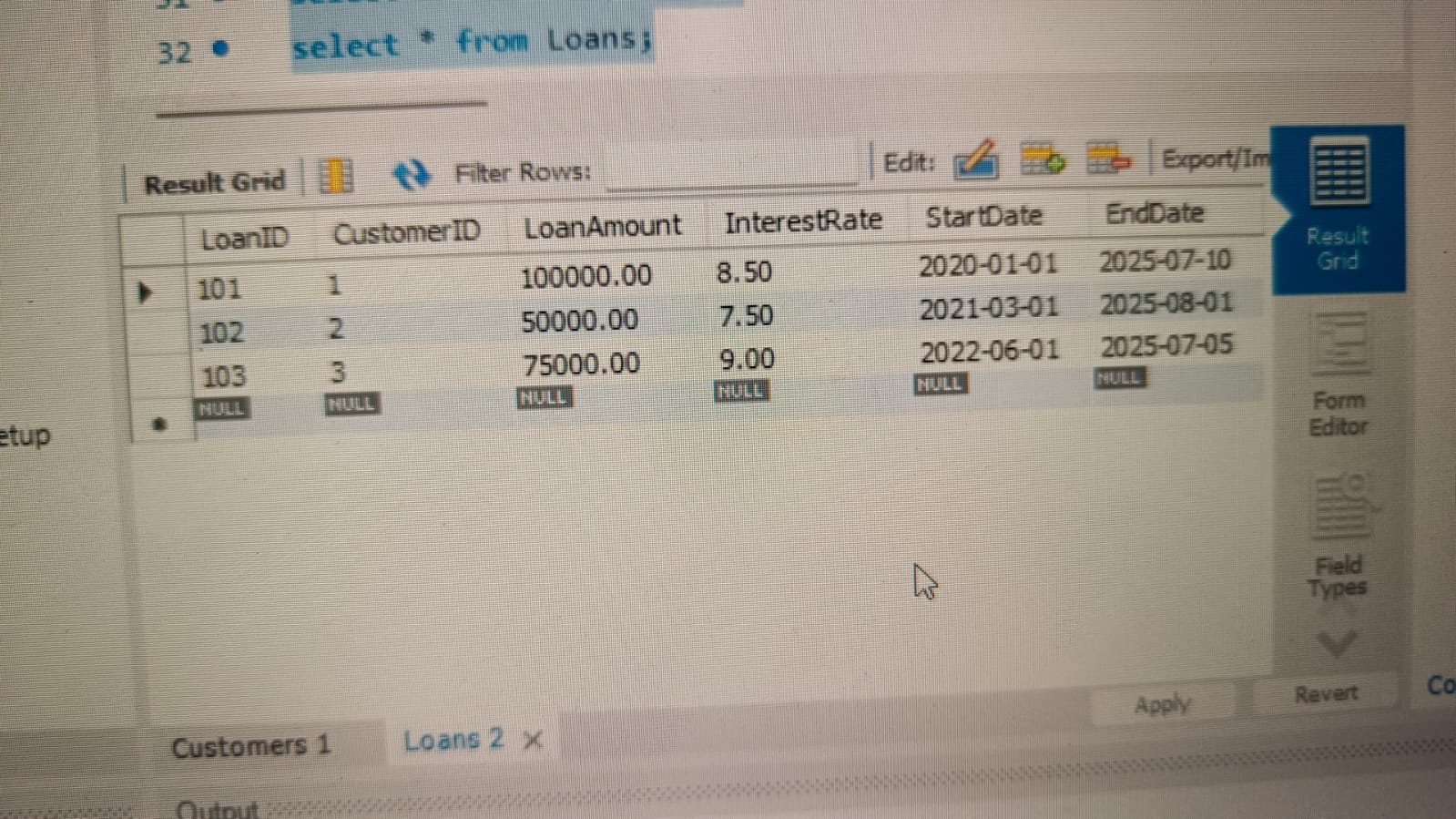
(103, 3, 75000, 9.0, '2022-06-01', '2025-07-05');

SELECT \* from Customers;

SELECT \* from Loans;

**OUTPUTS:**

****

****

**SCENARIO 1:**

DELIMITER //

CREATE PROCEDURE ApplySeniorDiscount()

BEGIN

DECLARE done INT DEFAULT FALSE;

DECLARE cust\_id INT;

DECLARE age INT;

DECLARE cust\_cursor CURSOR FOR

SELECT CustomerID, TIMESTAMPDIFF(YEAR, DOB, CURDATE()) AS Age FROM Customers;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

OPEN cust\_cursor;

read\_loop: LOOP

FETCH cust\_cursor INTO cust\_id, age;

IF done THEN

LEAVE read\_loop;

END IF;

IF age > 60 THEN

UPDATE Loans SET InterestRate = InterestRate - 1 WHERE CustomerID = cust\_id;

END IF;

END LOOP;

CLOSE cust\_cursor;

END //

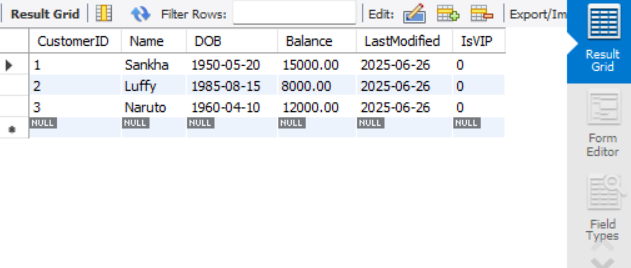
DELIMITER ;

CALL ApplySeniorDiscount();

SELECT \* from Customers;

SELECT \* from Loans;

**OUTPUTS:**

****

**A screenshot of a computer

AI-generated content may be incorrect.**

**SCENARIO 2:**

CREATE PROCEDURE PromoteToVIP()

BEGIN

DECLARE done INT DEFAULT FALSE;

DECLARE cust\_id INT;

DECLARE bal DECIMAL(10,2);

DECLARE cust\_cursor CURSOR FOR

SELECT CustomerID, Balance FROM Customers;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

OPEN cust\_cursor;

read\_loop: LOOP

FETCH cust\_cursor INTO cust\_id, bal;

IF done THEN

LEAVE read\_loop;

END IF;

IF bal > 10000 THEN

UPDATE Customers SET IsVIP = TRUE WHERE CustomerID = cust\_id;

END IF;

END LOOP;

CLOSE cust\_cursor;

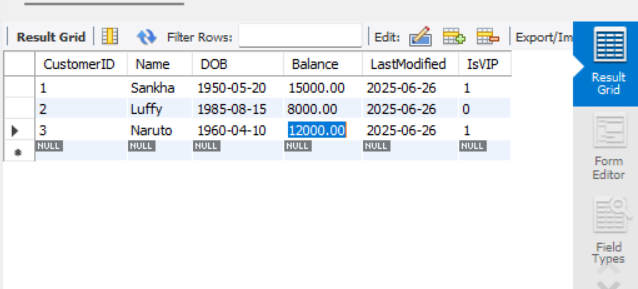
END //

DELIMITER ;

CALL PromoteToVIP();

SELECT \* from Customers;

**OUTPUT:**

****

**SCENARIO 3:**

DELIMITER //

CREATE PROCEDURE SendLoanReminders()

BEGIN

DECLARE done INT DEFAULT FALSE;

DECLARE cust\_id INT;

DECLARE loan\_due DATE;

DECLARE cust\_name VARCHAR(100);

DECLARE loan\_cursor CURSOR FOR

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

FROM Customers c

JOIN Loans l ON c.CustomerID = l.CustomerID

WHERE l.EndDate BETWEEN CURDATE() AND CURDATE() + INTERVAL 30 DAY;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

OPEN loan\_cursor;

read\_loop: LOOP

FETCH loan\_cursor INTO cust\_id, cust\_name, loan\_due;

IF done THEN

LEAVE read\_loop;

END IF;

SELECT CONCAT('Reminder: Mr./Mrs. ', cust\_name, ', your loan is due on ', loan\_due) AS Reminder;

END LOOP;

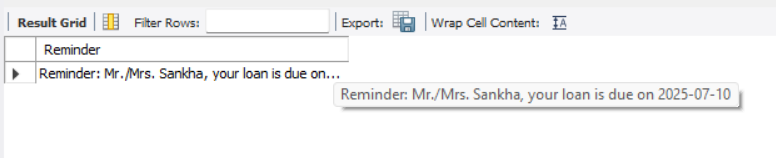
CLOSE loan\_cursor;

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.**

**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.**

**SOLUTION:**

**CODE:**

CREATE DATABASE BankDB2;

USE BankDB2;

CREATE TABLE IF NOT EXISTS Customers (

CustomerID INT PRIMARY KEY,

Name VARCHAR(100),

DOB DATE,

Balance DECIMAL(10,2),

LastModified DATE,

IsVIP BOOLEAN DEFAULT FALSE

);

CREATE TABLE IF NOT EXISTS Accounts (

AccountID INT PRIMARY KEY,

CustomerID INT,

AccountType VARCHAR(20),

Balance DECIMAL(10,2),

LastModified DATE,

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

CREATE TABLE IF NOT EXISTS Employees (

EmployeeID INT PRIMARY KEY,

Name VARCHAR(100),

Position VARCHAR(50),

Salary DECIMAL(10,2),

Department VARCHAR(50),

HireDate DATE

);

INSERT INTO Customers VALUES

(1, 'Sankha', '1950-05-20', 15000, CURDATE(), FALSE),

(2, 'Spandan', '1995-04-18', 8000, CURDATE(), FALSE),

(3, 'Kaori', '1988-12-25', 12000, CURDATE(), FALSE);

INSERT INTO Accounts VALUES

(201, 1, 'Savings', 10000.00, CURDATE()),

(202, 1, 'Savings', 5000.00, CURDATE()),

(203, 2, 'Savings', 7000.00, CURDATE()),

(204, 3, 'Savings', 6000.00, CURDATE());

INSERT INTO Employees VALUES

(1, 'Itachi', 'Manager', 50000.00, 'IT', '2021-01-01'),

(2, 'Sanji', 'Clerk', 30000.00, 'HR', '2022-03-01'),

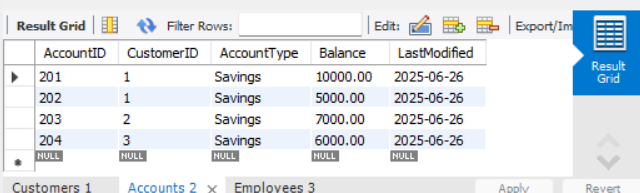
(3, 'Levi', 'Developer', 40000.00, 'IT', '2020-11-11');

SELECT \* from Customers;

SELECT \* from Accounts;

SELECT \* from Employees;

**OUTPUTS:A screenshot of a computer

AI-generated content may be incorrect. A screenshot of a computer

AI-generated content may be incorrect.**

**SCENARIO 1:**

DELIMITER //

CREATE PROCEDURE ProcessMonthlyInterest()

BEGIN

DECLARE done INT DEFAULT FALSE;

DECLARE acc\_id INT;

DECLARE bal DECIMAL(10,2);

DECLARE cur CURSOR FOR

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

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

OPEN cur;

read\_loop: LOOP

FETCH cur INTO acc\_id, bal;

IF done THEN

LEAVE read\_loop;

END IF;

UPDATE Accounts

SET Balance = bal + (bal \* 0.01),

LastModified = CURDATE()

WHERE AccountID = acc\_id;

END LOOP;

CLOSE cur;

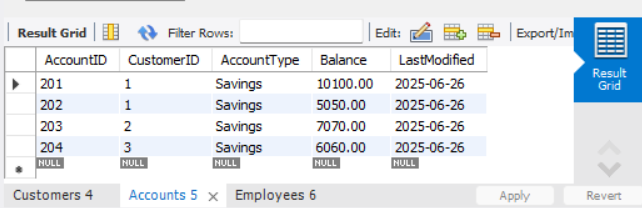
END //

DELIMITER ;

CALL ProcessMonthlyInterest();

SELECT \* from Accounts;

**OUTPUT:**

****

**SCENARIO 2:**

DELIMITER //

CREATE PROCEDURE UpdateEmployeeBonus(

IN dept\_name VARCHAR(50),

IN bonus\_percent DECIMAL(5,2)

)

BEGIN

UPDATE Employees

SET Salary = Salary + (Salary \* bonus\_percent / 100)

WHERE Department = dept\_name;

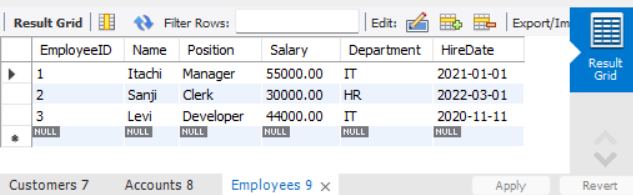
END //

DELIMITER ;

SET SQL\_SAFE\_UPDATES = 0;

CALL UpdateEmployeeBonus('IT', 10);

**OUTPUT:**

****

**SCENARIO 3:**

DELIMITER //

CREATE PROCEDURE TransferFunds(

IN fromAccount INT,

IN toAccount INT,

IN amount DECIMAL(10,2)

)

BEGIN

DECLARE fromBalance DECIMAL(10,2);

SELECT Balance INTO fromBalance FROM Accounts WHERE AccountID = fromAccount;

IF fromBalance >= amount THEN

UPDATE Accounts

SET Balance = Balance - amount,

LastModified = CURDATE()

WHERE AccountID = fromAccount;

UPDATE Accounts

SET Balance = Balance + amount,

LastModified = CURDATE()

WHERE AccountID = toAccount;

ELSE

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Insufficient balance for transfer';

END IF;

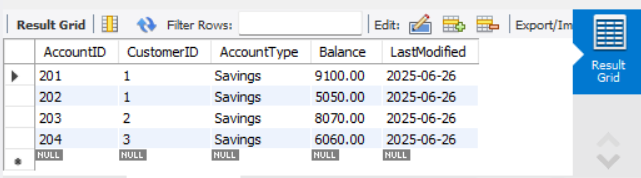
END //

DELIMITER ;

CALL TransferFunds(201, 203, 1000);

SELECT \* from Accounts;

**OUTPUT:**

****