**Exercise 1: Control Structures**

**apply\_senior\_discount\_procedure.sql**

DELIMITER //

-- This procedure applies a 1% interest discount to loans for customers over 60 years old.

CREATE PROCEDURE ApplySeniorLoanDiscount()

BEGINDECLARE done INT DEFAULT FALSE;

DECLARE v\_customer\_id INT;

DECLARE v\_dob DATE;

DECLARE v\_age INT;

DECLARE cur CURSOR FOR

SELECT CustomerID, DOB FROM Customers;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

OPEN cur;

read\_loop: LOOP

FETCH cur INTO v\_customer\_id, v\_dob;

IF done THEN

LEAVE read\_loop;

END IF;

SET v\_age = TIMESTAMPDIFF(YEAR, v\_dob, CURDATE());

IF v\_age > 60 THEN

UPDATE Loans

SET InterestRate = InterestRate - 1

WHERE CustomerID = v\_customer\_id;

END IF;

END LOOP;

CLOSE cur;

END;

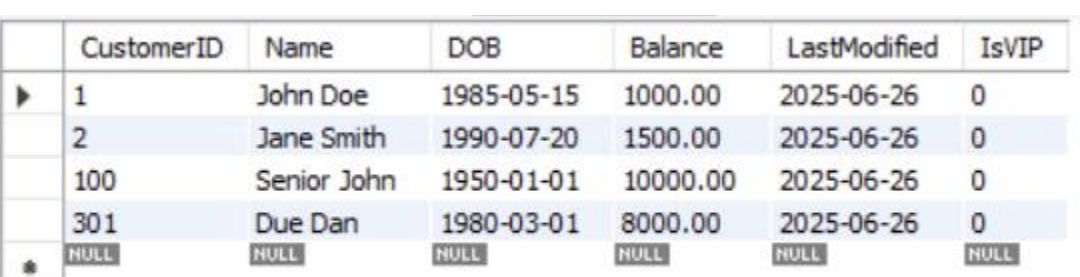
//DELIMITER ;

CALL ApplySeniorLoanDiscount();

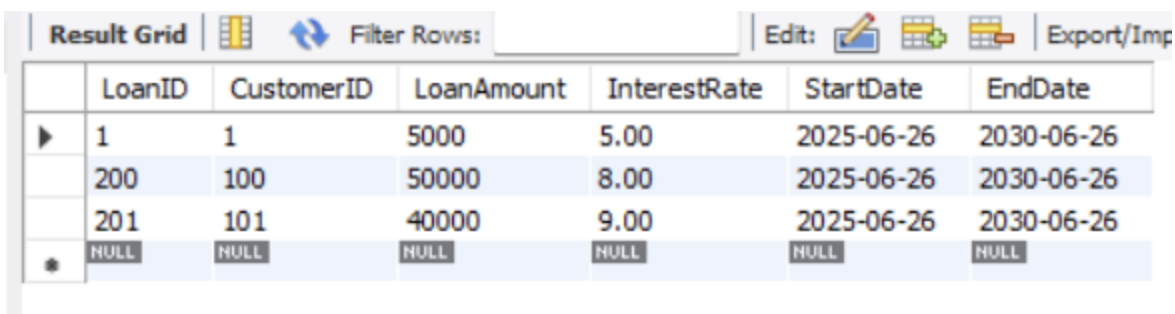
**Output**:

Scenario1:

1. A Customer name “Senior Jhon” with CustomerID 100 have the age greater than 60

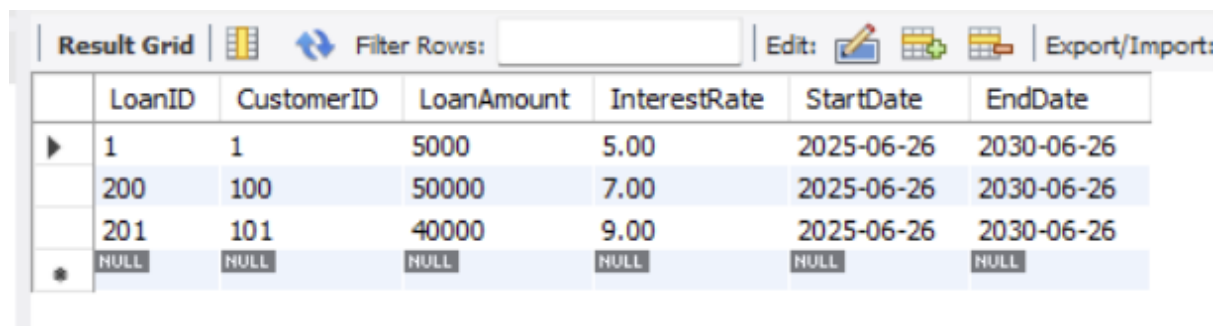


2. The Interest rate with CustomerID 100 is 8% before calling the procedure



3. The Interest rate after calling the procedure became 7% because the discount of 1%

applied.



**lone\_due\_reminder\_procedure.sql**

DELIMITER //

CREATE PROCEDURE SendLoanDueReminders()

BEGIN

DECLARE done INT DEFAULT FALSE;

DECLARE v\_loan\_id INT;

DECLARE v\_customer\_id INT;

DECLARE v\_end\_date DATE;

DECLARE v\_customer\_name VARCHAR(100);

-- Cursor to select loans due within next 30 days

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 = TRUE;

OPEN cur;

read\_loop: LOOP

FETCH cur INTO v\_loan\_id, v\_customer\_id, v\_end\_date, v\_customer\_name;

IF done THEN

LEAVE read\_loop;

END IF;

-- Print reminder message

SELECT CONCAT('Reminder: Dear ', v\_customer\_name,

', your loan ID ', v\_loan\_id,

' is due on ', v\_end\_date) AS ReminderMessage;

END LOOP;

CLOSE cur;

END;

//

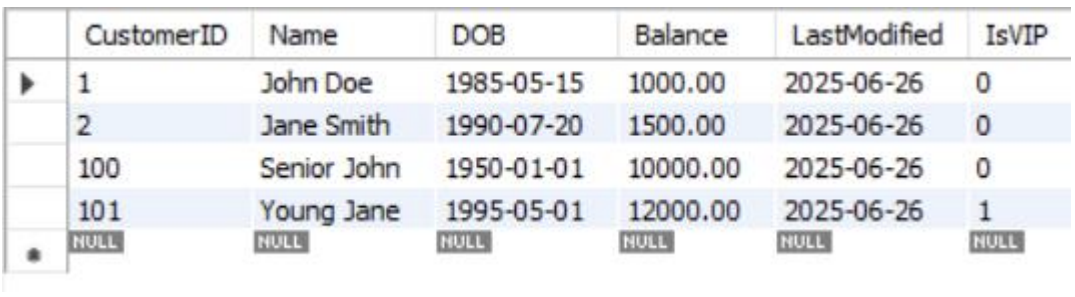
DELIMITER ;

CALL SendLoanDueReminders();

**Output:**

Scenario 2:

Customer is promoted to VIP status by updating the value of IsVIP to 1 if balance is over $10,000 using procedure in SQL. Here CustomerID 101 have balance 12,000 thus, It will get VIP status.



**promote\_vip\_customers\_procedure.sql**

ALTER TABLE Customers ADD COLUMN IsVIP BOOLEAN DEFAULT FALSE;

DELIMITER //

CREATE PROCEDURE PromoteVIPCustomers()

BEGIN

DECLARE done INT DEFAULT FALSE;

DECLARE v\_customer\_id INT;

DECLARE v\_balance DECIMAL(10,2);

DECLARE cur CURSOR FOR

SELECT CustomerID, Balance FROM Customers;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

OPEN cur;

read\_loop: LOOP

FETCH cur INTO v\_customer\_id, v\_balance;

IF done THEN

LEAVE read\_loop;

END IF;

IF v\_balance > 10000 THEN

UPDATE Customers

SET IsVIP = TRUE

WHERE CustomerID = v\_customer\_id;

END IF;

END LOOP;

CLOSE cur;

END;

//

DELIMITER ;

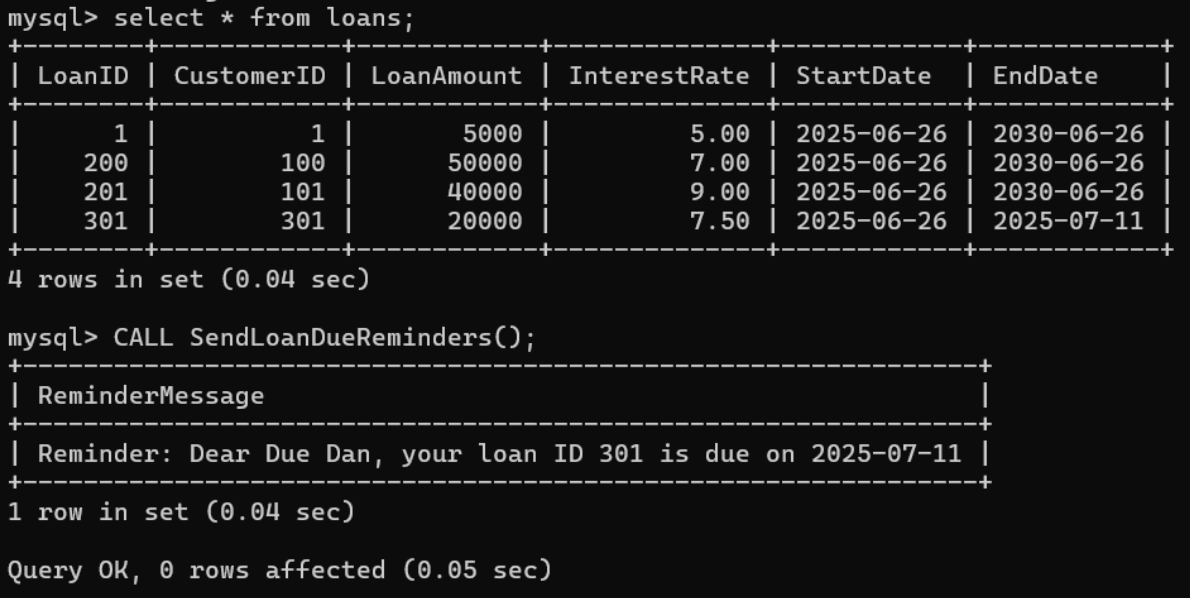
-- calling the procedure to promote customers

CALL PromoteVIPCustomers();

**Output:**

Scenario 3

A procedure is implemented that fetches all loans due in the next 30 days and prints a reminder message for each customer. Here CustomerID 301 will get a message.



**Exercise 3: Stored Procedures**

**process\_monthly\_interest\_procedure.sql**

DELIMITER //

CREATE PROCEDURE ProcessMonthlyInterest()

BEGIN

DECLARE done INT DEFAULT FALSE;

DECLARE v\_account\_id INT;

DECLARE v\_balance DECIMAL(10,2);

-- Cursor to select all savings accounts

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 v\_account\_id, v\_balance;

IF done THEN

LEAVE read\_loop;

END IF;

-- Update account balance with 1% interest

UPDATE Accounts

SET Balance = v\_balance + (v\_balance \* 0.01)

WHERE AccountID = v\_account\_id;

END LOOP;

CLOSE cur;

END;

//

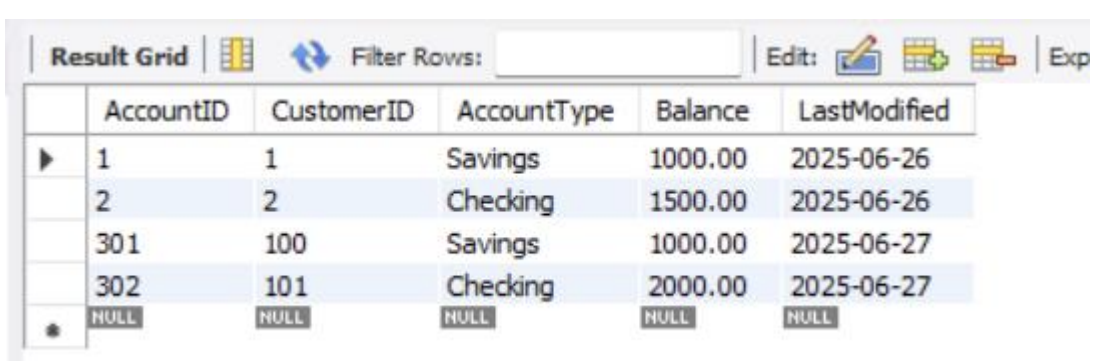
DELIMITER ;

CALL ProcessMonthlyInterest();

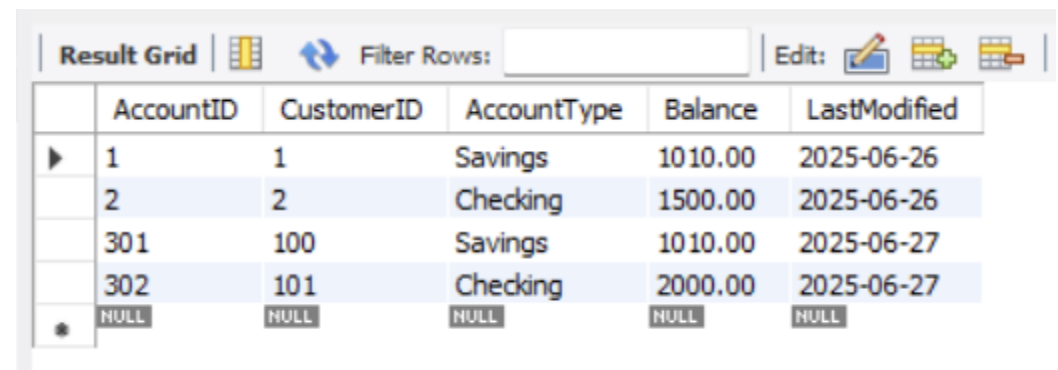
**Output:**

Scenario 1:

Here are Savings accounts before getting update of 1% of interest rate in balance.



Here are Savings accounts after getting update of 1% of interest rate in balance using stored procedure in SQL.



**transfer\_funds\_procedure.sql**

DELIMITER //

CREATE PROCEDURE TransferFunds(

IN from\_account\_id INT,

IN to\_account\_id INT,

IN amount DECIMAL(10,2)

)

BEGIN

DECLARE from\_balance DECIMAL(10,2);

-- check if the amount is positive

IF amount <= 0 THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Transfer amount must be greater than zero';

END IF;

-- check if source account exists and fetch its balance

SELECT Balance INTO from\_balance

FROM Accounts

WHERE AccountID = from\_account\_id;

-- if source account has insufficient balance, throw an error

IF from\_balance < amount THEN

SIGNAL SQLSTATE '45000'

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

END IF;

-- deduct amount from source account

UPDATE Accounts

SET Balance = Balance - amount

WHERE AccountID = from\_account\_id;

-- add amount to destination account

UPDATE Accounts

SET Balance = Balance + amount

WHERE AccountID = to\_account\_id;

END;

//

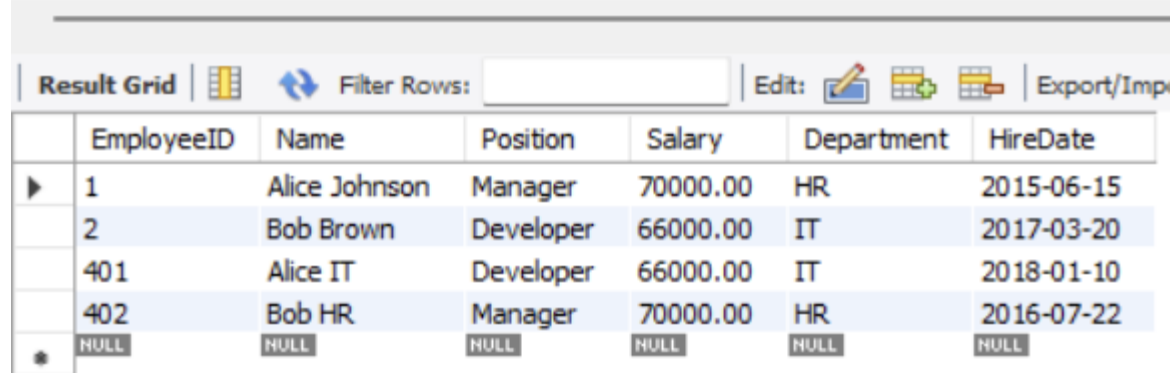
DELIMITER ;

Scenario 2:

Here is the employee salary before getting bonus of employee table



Below is the employee salary after getting bonus, Here the bonus is applied to IT department and percentage is 10%.



**update\_employee\_bonus\_procedure.sql**

DELIMITER //

CREATE PROCEDURE UpdateEmployeeBonus(

IN dept\_name VARCHAR(50),

IN bonus\_percent DECIMAL(5,2) -- e.g. 5.00 for 5%

)

BEGIN

-- Update salaries by applying the bonus percentage

UPDATE Employees

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

WHERE Department = dept\_name;

END;

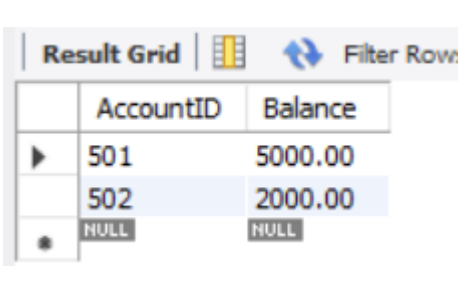
//

DELIMITER ;

CALL UpdateEmployeeBonus('IT', 10.00);

Scenario 3 :

Here is the AccountID and Balance of customer before fund transfer.



Here is the updated entries after the fund transfer of 1000.00 between AccountIDs 501 and 502. Here the procedure is used in SQL.

