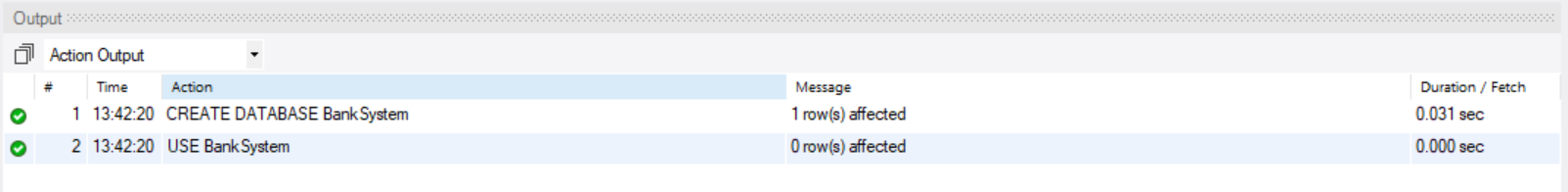
WEEK-2

**PL/SQL programming**

CREATE DATABASE BankSystem;

USE BankSystem;



CREATE TABLE Customers (

CustomerID INT PRIMARY KEY,

Name VARCHAR(100),

DOB DATE,

Balance DECIMAL(10,2),

LastModified DATETIME

);

CREATE TABLE Accounts (

AccountID INT PRIMARY KEY,

CustomerID INT,

AccountType VARCHAR(20),

Balance DECIMAL(10,2),

LastModified DATETIME,

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

CREATE TABLE Transactions (

TransactionID INT PRIMARY KEY,

AccountID INT,

TransactionDate DATE,

Amount DECIMAL(10,2),

TransactionType VARCHAR(10),

FOREIGN KEY (AccountID) REFERENCES Accounts(AccountID)

);

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)

);

CREATE TABLE Employees (

EmployeeID INT PRIMARY KEY,

Name VARCHAR(100),

Position VARCHAR(50),

Salary DECIMAL(10,2),

Department VARCHAR(50),

HireDate DATE

);

-- Insert Customers

INSERT INTO Customers (CustomerID, Name, DOB, Balance, LastModified)

VALUES

(1, 'John Doe', '1985-05-15', 1000.00, NOW()),

(2, 'Jane Smith', '1990-07-20', 1500.00, NOW());

-- Insert Accounts

INSERT INTO Accounts (AccountID, CustomerID, AccountType, Balance, LastModified)

VALUES

(1, 1, 'Savings', 1000.00, NOW()),

(2, 2, 'Checking', 1500.00, NOW());

-- Insert Transactions

INSERT INTO Transactions (TransactionID, AccountID, TransactionDate, Amount, TransactionType)

VALUES

(1, 1, NOW(), 200.00, 'Deposit'),

(2, 2, NOW(), 300.00, 'Withdrawal');

-- Insert Loans

INSERT INTO Loans (LoanID, CustomerID, LoanAmount, InterestRate, StartDate, EndDate)

VALUES

(1, 1, 5000.00, 5.00, CURDATE(), DATE\_ADD(CURDATE(), INTERVAL 60 MONTH));

-- Insert Employees

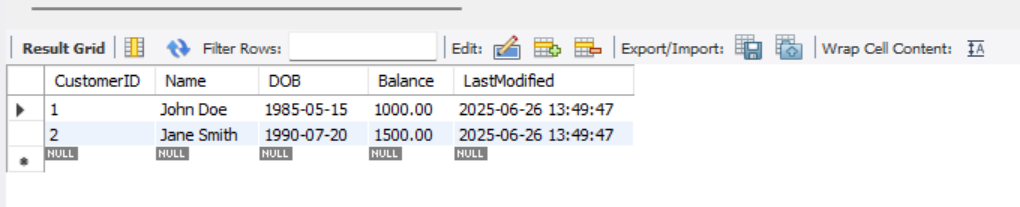
INSERT INTO Employees (EmployeeID, Name, Position, Salary, Department, HireDate)

VALUES

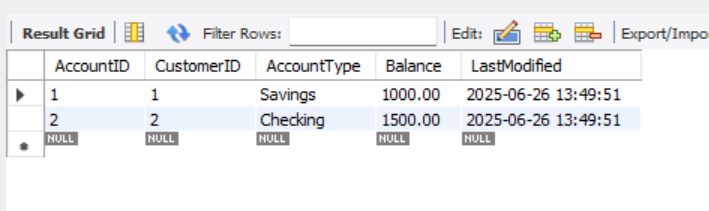
(1, 'Alice Johnson', 'Manager', 70000.00, 'HR', '2015-06-15'),

(2, 'Bob Brown', 'Developer', 60000.00, 'IT', '2017-03-20');

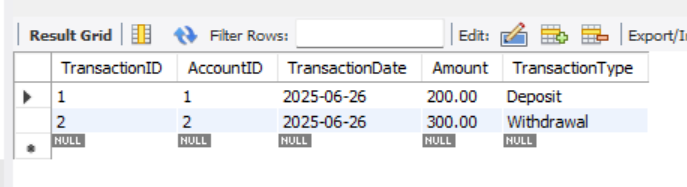
select \* from customers;



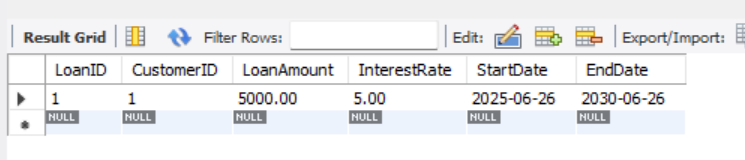
Select \* from Accounts;



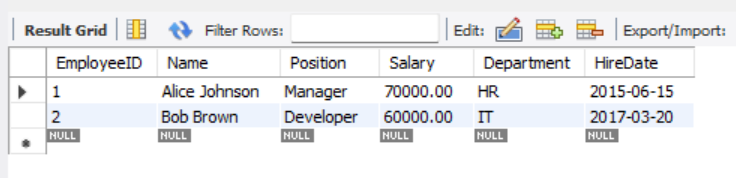
Select \* from Transactions;



Select \* from Loans;



Select \* from Employees;



**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.

DELIMITER $$

CREATE PROCEDURE ApplyInterestDiscount()

BEGIN

DECLARE done INT DEFAULT 0;

DECLARE c\_id INT;

DECLARE age INT;

-- Cursor to get CustomerIDs and their DOBs

DECLARE cur CURSOR FOR

SELECT CustomerID FROM Customers;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;

OPEN cur;

read\_loop: LOOP

FETCH cur INTO c\_id;

IF done THEN

LEAVE read\_loop;

END IF;

-- Calculate age

SET age = YEAR(CURDATE()) - YEAR((SELECT DOB FROM Customers WHERE CustomerID = c\_id));

IF age > 60 THEN

UPDATE Loans

SET InterestRate = InterestRate - 1

WHERE CustomerID = c\_id;

END IF;

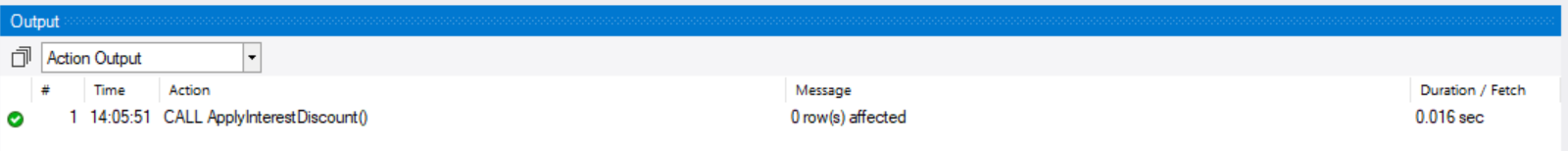
END LOOP;

CLOSE cur;

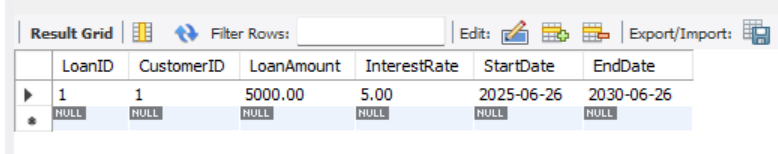
END $$

DELIMITER ;

CALL ApplyInterestDiscount();



Select \* from 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.

**Solution:**

ALTER TABLE Customers ADD COLUMN IsVIP BOOLEAN DEFAULT FALSE;

DELIMITER $$

CREATE PROCEDURE PromoteVIPCustomers()

BEGIN

DECLARE done INT DEFAULT 0;

DECLARE cid INT;

DECLARE bal DECIMAL(10,2);

DECLARE cur CURSOR FOR SELECT CustomerID, Balance FROM Customers;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;

OPEN cur;

vip\_loop: LOOP

FETCH cur INTO cid, bal;

IF done THEN

LEAVE vip\_loop;

END IF;

IF bal > 10000 THEN

UPDATE Customers

SET IsVIP = TRUE

WHERE CustomerID = cid;

END IF;

END LOOP;

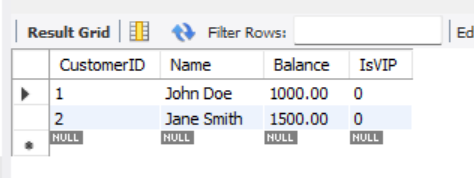
CLOSE cur;

END $$

DELIMITER ;

CALL PromoteVIPCustomers();

SELECT CustomerID, Name, Balance, IsVIP FROM 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.

**Solution:**

DELIMITER $$

CREATE PROCEDURE SendLoanReminders()

BEGIN

DECLARE done INT DEFAULT 0;

DECLARE cid INT;

DECLARE cname VARCHAR(100);

DECLARE loanid INT;

DECLARE due\_date DATE;

DECLARE cur CURSOR FOR

SELECT L.LoanID, C.CustomerID, C.Name, L.EndDate

FROM Loans L

JOIN Customers C ON C.CustomerID = L.CustomerID

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

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;

OPEN cur;

reminder\_loop: LOOP

FETCH cur INTO loanid, cid, cname, due\_date;

IF done THEN

LEAVE reminder\_loop;

END IF;

-- Print reminder message

SELECT CONCAT('Reminder: Dear ', cname, ', your loan (Loan ID: ', loanid, ') is due on ', due\_date) AS ReminderMessage;

END LOOP;

CLOSE cur;

END $$

DELIMITER ;

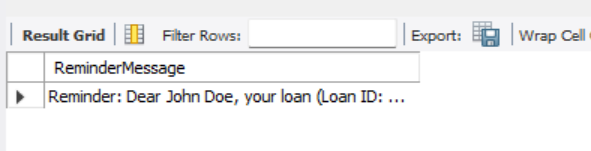
CALL SendLoanReminders();

UPDATE Loans

SET EndDate = DATE\_ADD(CURDATE(), INTERVAL 10 DAY)

WHERE LoanID = 1;

CALL SendLoanReminders();



**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.

**Solution:**

DELIMITER $$

CREATE PROCEDURE ProcessMonthlyInterest()

BEGIN

DECLARE done INT DEFAULT 0;

DECLARE acc\_id INT;

DECLARE current\_balance DECIMAL(10,2);

DECLARE interest 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;

interest\_loop: LOOP

FETCH cur INTO acc\_id, current\_balance;

IF done THEN

LEAVE interest\_loop;

END IF;

SET interest = current\_balance \* 0.01;

UPDATE Accounts

SET Balance = Balance + interest

WHERE AccountID = acc\_id;

END LOOP;

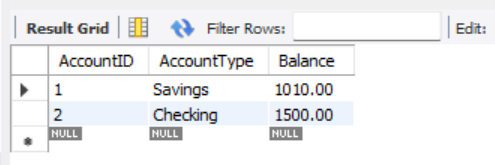
CLOSE cur;

END $$

DELIMITER ;

CALL ProcessMonthlyInterest();

SELECT AccountID, AccountType, Balance FROM 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

**Solution:**

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 EmployeeID IN (

SELECT e.EmployeeID FROM (

SELECT EmployeeID FROM Employees WHERE Department = dept\_name

) AS e

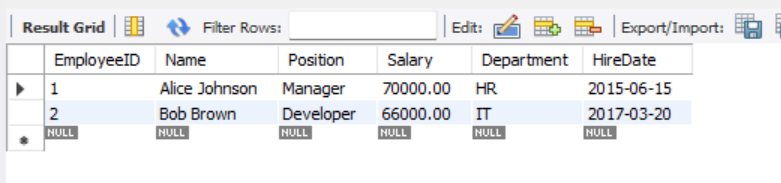
);

END $$

DELIMITER ;

CALL UpdateEmployeeBonus('IT', 10);

SELECT \* FROM 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.

**Solution:**

DELIMITER $$

CREATE PROCEDURE TransferFunds(

IN from\_account INT,

IN to\_account INT,

IN amount DECIMAL(10,2)

)

BEGIN

DECLARE from\_balance DECIMAL(10,2);

-- Get current balance of source account

SELECT Balance INTO from\_balance

FROM Accounts

WHERE AccountID = from\_account;

-- Check for sufficient balance

IF from\_balance >= amount THEN

-- Deduct from source

UPDATE Accounts

SET Balance = Balance - amount

WHERE AccountID = from\_account;

-- Add to destination

UPDATE Accounts

SET Balance = Balance + amount

WHERE AccountID = to\_account;

ELSE

-- Show insufficient balance message

SELECT 'Transfer failed: insufficient balance.' AS Message;

END IF;

END $$

DELIMITER ;

CALL TransferFunds(1, 2, 200);

SELECT AccountID, Balance FROM Accounts;

