**WEEK -2 PL/SQL HANDSON**

**Exercise 1: Control Structures**

**DATABASE**

CREATE Database cts1;

use cts1;

CREATE TABLE customers (

customer\_id INT PRIMARY KEY,

name VARCHAR(100),

age INT,

balance DECIMAL(10,2),

loan\_interest\_rate DECIMAL(5,2),

IsVIP BOOLEAN DEFAULT FALSE,

loan\_due\_date DATE

);

INSERT INTO customers (customer\_id, name, age, balance, loan\_interest\_rate, IsVIP, loan\_due\_date)

VALUES

(1, 'Alice', 65, 12000.00, 9.5, FALSE, '2025-07-01'),

(2, 'Bob', 58, 9500.00, 10.0, FALSE, '2025-06-30'),

(3, 'Charlie', 61, 20000.00, 8.5, FALSE, '2025-07-10'),

(4, 'David', 45, 8000.00, 11.0, FALSE, '2025-08-05'),

(5, 'Eve', 62, 10500.00, 9.0, FALSE, '2025-06-20');

**Scenario-1:**

DELIMITER $$

CREATE PROCEDURE apply\_interest\_discount()

BEGIN

DECLARE done INT DEFAULT FALSE;

DECLARE cust\_id INT;

DECLARE current\_interest DECIMAL(5,2);

DECLARE cur CURSOR FOR SELECT customer\_id, loan\_interest\_rate FROM customers WHERE age > 60;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

OPEN cur;

read\_loop: LOOP

FETCH cur INTO cust\_id, current\_interest;

IF done THEN

LEAVE read\_loop;

END IF;

UPDATE customers

SET loan\_interest\_rate = current\_interest - 1

WHERE customer\_id = cust\_id;

END LOOP;

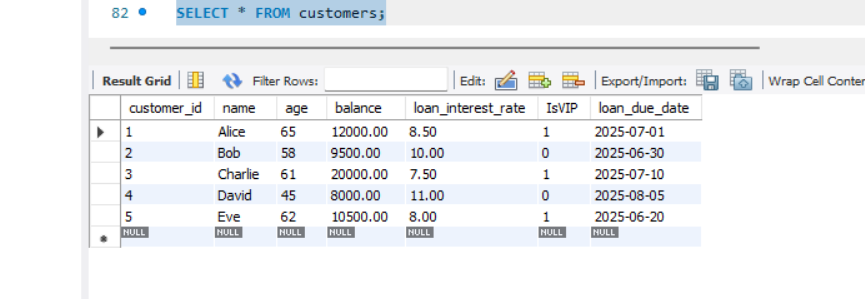
CLOSE cur;

END$$

DELIMITER ;

CALL apply\_interest\_discount();

SELECT \* FROM customers;

****

**SCENARIO-2:**

DELIMITER $$

CREATE PROCEDURE promote\_vip\_customers()

BEGIN

DECLARE done INT DEFAULT FALSE;

DECLARE cust\_id INT;

DECLARE cust\_balance DECIMAL(10,2);

DECLARE cur CURSOR FOR SELECT customer\_id, balance FROM customers WHERE balance > 10000;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

OPEN cur;

vip\_loop: LOOP

FETCH cur INTO cust\_id, cust\_balance;

IF done THEN

LEAVE vip\_loop;

END IF;

UPDATE customers

SET IsVIP = TRUE

WHERE customer\_id = cust\_id;

END LOOP;

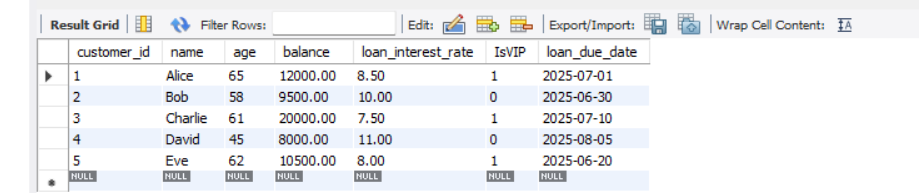
CLOSE cur;

END$$

DELIMITER ;

CALL promote\_vip\_customers();

SELECT \* FROM customers;



**SCENARIO-3:**

DELIMITER $$

CREATE PROCEDURE send\_loan\_reminders()

BEGIN

DECLARE done INT DEFAULT FALSE;

DECLARE cust\_name VARCHAR(100);

DECLARE due\_date DATE;

DECLARE cur CURSOR FOR

SELECT name, loan\_due\_date FROM customers

WHERE loan\_due\_date BETWEEN CURDATE() AND DATE\_ADD(CURDATE(), INTERVAL 30 DAY);

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

OPEN cur;

reminder\_loop: LOOP

FETCH cur INTO cust\_name, due\_date;

IF done THEN

LEAVE reminder\_loop;

END IF;

SELECT CONCAT('Reminder: ', cust\_name, ', your loan is due on ', due\_date) AS Reminder;

END LOOP;

CLOSE cur;

END$$

DELIMITER ;

CALL send\_loan\_reminders();

**A screenshot of a computer

AI-generated content may be incorrect.**

**Exercise 3: Stored Procedures**

**Database :**

CREATE DATABASE BankDB;

use BankDB;

CREATE TABLE Accounts (

AccountID INT PRIMARY KEY,

AccountType ENUM('Savings', 'Current') NOT NULL,

Balance DECIMAL(12,2) NOT NULL CHECK (Balance >= 0)

);

CREATE TABLE Employees (

EmployeeID INT PRIMARY KEY,

Name VARCHAR(50) NOT NULL,

Department VARCHAR(30) NOT NULL,

Salary DECIMAL(10,2) NOT NULL CHECK (Salary >= 0)

);

CREATE TABLE Customers (

CustomerID INT PRIMARY KEY,

Name VARCHAR(50),

Age INT,

Balance DECIMAL(12,2),

IsVIP BOOLEAN DEFAULT FALSE

);

INSERT INTO Accounts VALUES (101, 'Savings', 10000.00);

INSERT INTO Accounts VALUES (102, 'Current', 5000.00);

INSERT INTO Accounts VALUES (103, 'Savings', 15000.00);

INSERT INTO Employees VALUES (1, 'John', 'HR', 50000.00);

INSERT INTO Employees VALUES (2, 'Alice', 'IT', 60000.00);

INSERT INTO Employees VALUES (3, 'Bob', 'IT', 55000.00);

INSERT INTO Customers VALUES (201, 'Sam', 65, 8000.00, FALSE);

INSERT INTO Customers VALUES (202, 'Lily', 58, 12000.00, FALSE);

INSERT INTO Customers VALUES (203, 'David', 70, 15000.00, FALSE);

**Scenario-1:**

DELIMITER $$

CREATE PROCEDURE ProcessMonthlyInterest()

BEGIN

UPDATE Accounts

SET Balance = Balance + (Balance \* 0.01)

WHERE AccountType = 'Savings';

END$$

DELIMITER ;

CALL ProcessMonthlyInterest();

SELECT \* FROM Accounts WHERE AccountType = 'Savings';

**OUTPUT**

A screenshot of a computer

AI-generated content may be incorrect.

**SCENARIO-2**

DELIMITER $$

CREATE PROCEDURE UpdateEmployeeBonus(

IN deptName VARCHAR(30),

IN bonusPercent DECIMAL(5,2)

)

BEGIN

UPDATE Employees

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

WHERE Department = deptName;

END$$

DELIMITER ;

CALL UpdateEmployeeBonus('IT', 10.0);

SELECT \* FROM Employees WHERE Department = 'IT';

**OUTPUT:**

A screenshot of a computer

AI-generated content may be incorrect.

**SCENARIO-3:**

DELIMITER $$

CREATE PROCEDURE TransferFunds(

IN fromAcc INT,

IN toAcc INT,

IN amount DECIMAL(12,2)

)

BEGIN

DECLARE fromBalance DECIMAL(12,2);

-- Get balance of source account

SELECT Balance INTO fromBalance

FROM Accounts

WHERE AccountID = fromAcc;

-- Check sufficient balance

IF fromBalance < amount THEN

SIGNAL SQLSTATE '45000'

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

ELSE

-- Deduct from source

UPDATE Accounts

SET Balance = Balance - amount

WHERE AccountID = fromAcc;

-- Add to destination

UPDATE Accounts

SET Balance = Balance + amount

WHERE AccountID = toAcc;

END IF;

END$$

DELIMITER ;

CALL TransferFunds(101, 102, 2000.00);

SELECT \* FROM Accounts WHERE AccountID IN (101, 102);

**OUTPUT**

A screenshot of a computer

AI-generated content may be incorrect.