**Week 2 Module**

**PL/SQL Programming**

**Exercise 1: Control Structures**

**Query:**

CREATE TABLE Customers (

    CustomerID NUMBER PRIMARY KEY,

    Name VARCHAR2(50),

    Age NUMBER,

    Balance NUMBER,

    IsVIP VARCHAR2(5)

);

CREATE TABLE Loans (

    LoanID NUMBER PRIMARY KEY,

    CustomerID NUMBER,

    InterestRate NUMBER(5,2),

    DueDate DATE,

    FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

INSERT INTO Customers VALUES (1, 'Alice', 65, 15000, 'FALSE');

INSERT INTO Customers VALUES (2, 'Bob', 45, 8000, 'FALSE');

INSERT INTO Customers VALUES (3, 'Charlie', 70, 20000, 'FALSE');

INSERT INTO Customers VALUES (4, 'David', 30, 12000, 'FALSE');

INSERT INTO Loans VALUES (101, 1, 7.5, SYSDATE + 10);

INSERT INTO Loans VALUES (102, 2, 8.0, SYSDATE + 40);

INSERT INTO Loans VALUES (103, 3, 6.5, SYSDATE + 5);

INSERT INTO Loans VALUES (104, 4, 7.0, SYSDATE + 25);

COMMIT;

BEGIN

  FOR cust IN (

    SELECT c.CustomerID

    FROM Customers c

    JOIN Loans l ON c.CustomerID = l.CustomerID

    WHERE c.Age > 60

  ) LOOP

    UPDATE Loans

    SET InterestRate = InterestRate - 1

    WHERE CustomerID = cust.CustomerID;

  END LOOP;

  COMMIT;

END;

/

BEGIN

  FOR cust IN (

    SELECT CustomerID

    FROM Customers

    WHERE Balance > 10000

  ) LOOP

    UPDATE Customers

    SET IsVIP = 'TRUE'

    WHERE CustomerID = cust.CustomerID;

  END LOOP;

  COMMIT;

END;

/

SET SERVEROUTPUT ON;

BEGIN

  FOR loan\_rec IN (

    SELECT l.CustomerID, l.DueDate, c.Name

    FROM Loans l

    JOIN Customers c ON l.CustomerID = c.CustomerID

    WHERE l.DueDate BETWEEN SYSDATE AND SYSDATE + 30

  ) LOOP

    DBMS\_OUTPUT.PUT\_LINE('Reminder: ' || loan\_rec.Name ||

                         ', your loan is due on ' || TO\_CHAR(loan\_rec.DueDate, 'DD-MON-YYYY'));

  END LOOP;

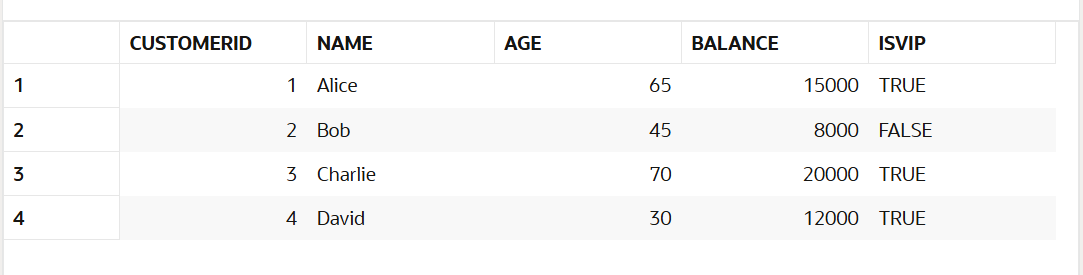
END;

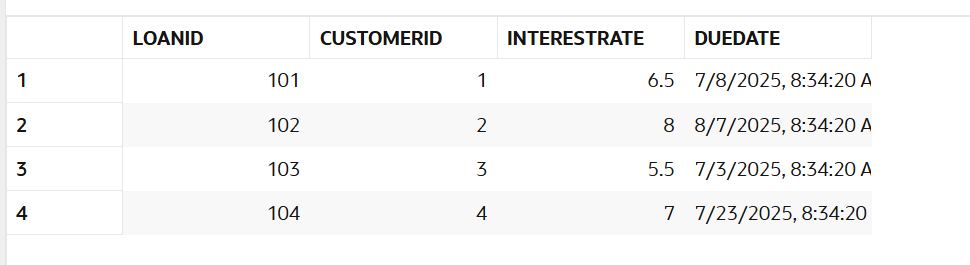
/

SELECT \* FROM Customers;

SELECT \* FROM Loans;

**Ouput:**

****



**Exercise 2: Error Handling**

**Query:**

CREATE TABLE Customers (

    CustomerID NUMBER PRIMARY KEY,

    Name VARCHAR2(50),

    Balance NUMBER

);

CREATE TABLE Employees (

    EmployeeID NUMBER PRIMARY KEY,

    Name VARCHAR2(50),

    Salary NUMBER

);

CREATE TABLE ErrorLogs (

    LogID NUMBER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,

    LogDate DATE DEFAULT SYSDATE,

    Message VARCHAR2(200)

);

INSERT INTO Customers VALUES (1, 'Alice', 1000);

INSERT INTO Customers VALUES (2, 'Bob', 500);

INSERT INTO Customers VALUES (3, 'Charlie', 800);

INSERT INTO Employees VALUES (101, 'John', 30000);

INSERT INTO Employees VALUES (102, 'Sara', 40000);

COMMIT;

CREATE OR REPLACE PROCEDURE SafeTransferFunds(

    p\_from\_id IN NUMBER,

    p\_to\_id IN NUMBER,

    p\_amount IN NUMBER

) IS

    v\_balance NUMBER;

BEGIN

    SELECT Balance INTO v\_balance FROM Customers WHERE CustomerID = p\_from\_id;

    IF v\_balance < p\_amount THEN

        RAISE\_APPLICATION\_ERROR(-20001, 'Insufficient funds in source account.');

    END IF;

    UPDATE Customers SET Balance = Balance - p\_amount WHERE CustomerID = p\_from\_id;

    UPDATE Customers SET Balance = Balance + p\_amount WHERE CustomerID = p\_to\_id;

    COMMIT;

EXCEPTION

    WHEN OTHERS THEN

        ROLLBACK;

        INSERT INTO ErrorLogs (Message)

        VALUES (SUBSTR('Error during transfer: ' || SQLERRM, 1, 200));

        COMMIT;

END;

/

CREATE OR REPLACE PROCEDURE UpdateSalary(

    p\_emp\_id IN NUMBER,

    p\_percent IN NUMBER

) IS

BEGIN

    UPDATE Employees

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

    WHERE EmployeeID = p\_emp\_id;

    IF SQL%ROWCOUNT = 0 THEN

        RAISE\_APPLICATION\_ERROR(-20002, 'Employee not found.');

    END IF;

  COMMIT;

EXCEPTION

    WHEN OTHERS THEN

        ROLLBACK;

        INSERT INTO ErrorLogs (Message)

        VALUES (SUBSTR('Error updating salary: ' || SQLERRM, 1, 200));

        COMMIT;

END;

/

CREATE OR REPLACE PROCEDURE AddNewCustomer(

    p\_id IN NUMBER,

    p\_name IN VARCHAR2,

    p\_balance IN NUMBER

) IS

BEGIN

    INSERT INTO Customers (CustomerID, Name, Balance)

    VALUES (p\_id, p\_name, p\_balance);

    COMMIT;

EXCEPTION

    WHEN DUP\_VAL\_ON\_INDEX THEN

        INSERT INTO ErrorLogs (Message)

        VALUES (SUBSTR('Customer ID already exists: ' || p\_id, 1, 200));

        COMMIT;

    WHEN OTHERS THEN

        INSERT INTO ErrorLogs (Message)

        VALUES (SUBSTR('Error adding customer: ' || SQLERRM, 1, 200));

        COMMIT;

END;

/

BEGIN

    SafeTransferFunds(1, 2, 300);

END;

/

BEGIN

    SafeTransferFunds(2, 3, 9999);

END;

/

BEGIN

    UpdateSalary(101, 10);

END;

/

BEGIN

    UpdateSalary(999, 5);

END;

/

BEGIN

    AddNewCustomer(5, 'David', 700);

END;

/

BEGIN

    AddNewCustomer(1, 'Duplicate Alice', 200);

END;

/

SELECT \* FROM ErrorLogs;