**CONTROL STRUCTURES**

**CODE:**

CREATE TABLE customers (

    customer\_id   NUMBER PRIMARY KEY,

    name          VARCHAR2(100),

    age           NUMBER,

    balance       NUMBER,

    isvip         VARCHAR2(5)

);

CREATE TABLE loans (

    loan\_id       NUMBER PRIMARY KEY,

    customer\_id   NUMBER,

    interest\_rate NUMBER,

    due\_date      DATE,

    FOREIGN KEY (customer\_id) REFERENCES customers(customer\_id)

);

INSERT INTO customers VALUES (1, 'Alice', 65, 12000, 'FALSE');

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

INSERT INTO customers VALUES (3, 'Charlie', 70, 15000, 'FALSE');

INSERT INTO customers VALUES (4, 'Diana', 30, 9500, 'FALSE');

INSERT INTO loans VALUES (101, 1, 9.5, SYSDATE + 20);

INSERT INTO loans VALUES (102, 2, 10.0, SYSDATE + 35);

INSERT INTO loans VALUES (103, 3, 8.5, SYSDATE + 10);

INSERT INTO loans VALUES (104, 4, 9.0, SYSDATE + 5);

COMMIT;

SET SERVEROUTPUT ON;

BEGIN

    FOR rec IN (

        SELECT c.customer\_id, l.loan\_id, l.interest\_rate

        FROM customers c

        JOIN loans l ON c.customer\_id = l.customer\_id

        WHERE c.age > 60

    ) LOOP

        UPDATE loans

        SET interest\_rate = interest\_rate - 1

        WHERE loan\_id = rec.loan\_id;

    END LOOP;

    COMMIT;

END;

/

BEGIN

    FOR rec IN (

        SELECT customer\_id

        FROM customers

        WHERE balance > 10000

    ) LOOP

        UPDATE customers

        SET isvip = 'TRUE'

        WHERE customer\_id = rec.customer\_id;

    END LOOP;

    COMMIT;

END;

/

BEGIN

    FOR rec IN (

        SELECT c.name, l.due\_date

        FROM customers c

        JOIN loans l ON c.customer\_id = l.customer\_id

        WHERE l.due\_date <= SYSDATE + 30

    ) LOOP

        DBMS\_OUTPUT.PUT\_LINE(

            'Reminder: ' || rec.name || ', your loan is due on ' || TO\_CHAR(rec.due\_date, 'DD-MON-YYYY')

        );

    END LOOP;

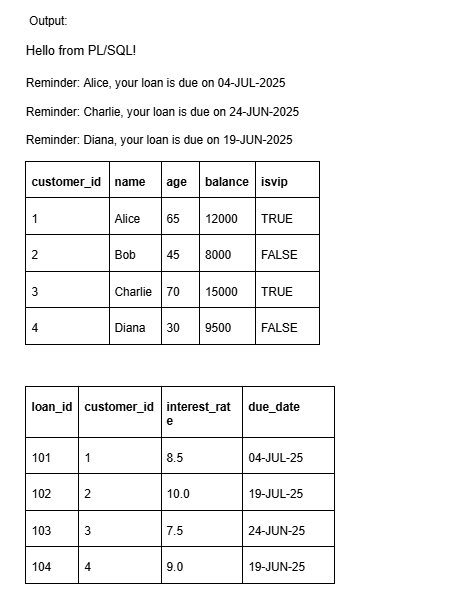
END;

/

SELECT \* FROM loans;

SELECT \* FROM customers;

**OUTPUT:**



**STORED PROCEDURES**

**CODE:**

SET SERVEROUTPUT ON;

-- 1. Savings Accounts Table

CREATE TABLE savings\_accounts (

account\_id NUMBER PRIMARY KEY,

customer\_id NUMBER,

balance NUMBER

);

-- 2. Employees Table

CREATE TABLE employees (

employee\_id NUMBER PRIMARY KEY,

name VARCHAR2(100),

department VARCHAR2(50),

salary NUMBER

);

-- 3. Bank Accounts Table (for transfers)

CREATE TABLE bank\_accounts (

account\_id NUMBER PRIMARY KEY,

customer\_id NUMBER,

balance NUMBER

);

-- Sample Data for savings\_accounts

INSERT INTO savings\_accounts VALUES (1, 101, 5000);

INSERT INTO savings\_accounts VALUES (2, 102, 7500);

-- Sample Data for employees

INSERT INTO employees VALUES (1, 'John', 'HR', 50000);

INSERT INTO employees VALUES (2, 'Sara', 'HR', 52000);

INSERT INTO employees VALUES (3, 'Mike', 'IT', 60000);

-- Sample Data for bank\_accounts

INSERT INTO bank\_accounts VALUES (1, 201, 10000);

INSERT INTO bank\_accounts VALUES (2, 202, 5000);

COMMIT;

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

BEGIN

FOR acc IN (SELECT account\_id, balance FROM savings\_accounts) LOOP

UPDATE savings\_accounts

SET balance = balance + (balance \* 0.01)

WHERE account\_id = acc.account\_id;

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('Monthly interest applied to all savings accounts.');

END;

/

EXEC ProcessMonthlyInterest;

SELECT \* FROM savings\_accounts;

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus(

p\_department IN VARCHAR2,

p\_bonus\_pct IN NUMBER

) IS

BEGIN

FOR emp IN (SELECT employee\_id, salary FROM employees WHERE department = p\_department) LOOP

UPDATE employees

SET salary = salary + (salary \* p\_bonus\_pct / 100)

WHERE employee\_id = emp.employee\_id;

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('Bonus applied to department: ' || p\_department);

END;

/

EXEC UpdateEmployeeBonus('HR', 10);

SELECT \* FROM employees;

CREATE OR REPLACE PROCEDURE TransferFunds(

p\_from\_account IN NUMBER,

p\_to\_account IN NUMBER,

p\_amount IN NUMBER

) IS

v\_balance NUMBER;

BEGIN

-- Get source balance

SELECT balance INTO v\_balance FROM bank\_accounts WHERE account\_id = p\_from\_account;

IF v\_balance < p\_amount THEN

DBMS\_OUTPUT.PUT\_LINE('Insufficient balance. Transfer failed.');

ELSE

-- Deduct from source

UPDATE bank\_accounts

SET balance = balance - p\_amount

WHERE account\_id = p\_from\_account;

-- Add to destination

UPDATE bank\_accounts

SET balance = balance + p\_amount

WHERE account\_id = p\_to\_account;

DBMS\_OUTPUT.PUT\_LINE('Transfer successful: ' || p\_amount || ' transferred.');

END IF;

END;

/

EXEC TransferFunds(1, 2, 3000);

SELECT \* FROM bank\_accounts;

**OUTPUT:**

