**1: Control Structures**

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, 9500, '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);

COMMIT;

**Scenario 1: Interest Discount for Customers > 60**

BEGIN

FOR cust\_rec IN (

SELECT customer\_id

FROM customers

WHERE age > 60

) LOOP

UPDATE loans

SET interest\_rate = interest\_rate - 1

WHERE customer\_id = cust\_rec.customer\_id;

END LOOP;

COMMIT;

END;

/

**Scenario 2: Promote to VIP Based on Balance**

BEGIN

FOR cust\_rec IN (

SELECT customer\_id

FROM customers

WHERE balance > 10000

) LOOP

UPDATE customers

SET isVIP = 'TRUE'

WHERE customer\_id = cust\_rec.customer\_id;

END LOOP;

COMMIT;

END;

/

**Scenario 3: Loan Reminders (Next 30 Days)**

BEGIN

FOR loan\_rec IN (

SELECT l.loan\_id, l.due\_date, c.name

FROM loans l

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

WHERE l.due\_date <= SYSDATE + 30

) LOOP

DBMS\_OUTPUT.PUT\_LINE('Reminder: Loan ID ' || loan\_rec.loan\_id ||

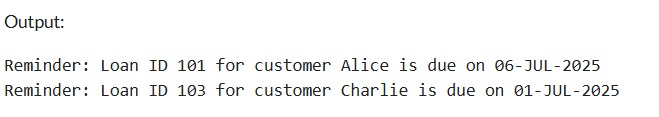
' for customer ' || loan\_rec.name ||

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

END LOOP;

END;

/



1. **Stored Procedures**

CREATE TABLE accounts (

account\_id NUMBER PRIMARY KEY,

customer\_name VARCHAR2(100),

balance NUMBER(10, 2),

account\_type VARCHAR2(20)

);

-- Employees Table (for bonuses)

CREATE TABLE employees (

emp\_id NUMBER PRIMARY KEY,

emp\_name VARCHAR2(100),

department\_id NUMBER,

salary NUMBER(10, 2)

);

-- Accounts

INSERT INTO accounts VALUES (1001, 'Alice', 5000.00, 'SAVINGS');

INSERT INTO accounts VALUES (1002, 'Bob', 2000.00, 'SAVINGS');

INSERT INTO accounts VALUES (1003, 'Charlie', 10000.00, 'CURRENT');

-- Employees

INSERT INTO employees VALUES (1, 'John', 101, 50000.00);

INSERT INTO employees VALUES (2, 'Mary', 102, 60000.00);

INSERT INTO employees VALUES (3, 'Steve', 101, 55000.00);

COMMIT;

-- **SCENARIO 1: ProcessMonthlyInterest**

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

BEGIN

UPDATE accounts

SET balance = balance + (balance \* 0.01)

WHERE account\_type = 'SAVINGS';

COMMIT;

END;

/

-- ✅ Test it

BEGIN

ProcessMonthlyInterest;

END;

/

-- **SCENARIO 2: UpdateEmployeeBonus**

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

p\_dept\_id IN employees.department\_id%TYPE,

p\_bonus\_percent IN NUMBER

) IS

BEGIN

UPDATE employees

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

WHERE department\_id = p\_dept\_id;

COMMIT;

END;

/

**Testing** (10% bonus to dept 101)

BEGIN

UpdateEmployeeBonus(101, 10);

END;

/

**-- SCENARIO 3: TransferFunds**

CREATE OR REPLACE PROCEDURE TransferFunds (

p\_from\_account IN accounts.account\_id%TYPE,

p\_to\_account IN accounts.account\_id%TYPE,

p\_amount IN NUMBER

) IS

v\_balance NUMBER;

BEGIN

-- Check source account balance

SELECT balance INTO v\_balance

FROM accounts

WHERE account\_id = p\_from\_account;

IF v\_balance < p\_amount THEN

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

END IF;

-- Debit source

UPDATE accounts

SET balance = balance - p\_amount

WHERE account\_id = p\_from\_account;

-- Credit destination

UPDATE accounts

SET balance = balance + p\_amount

WHERE account\_id = p\_to\_account;

COMMIT;

END;

/

**Testing (transfer 1000 from 1001 to 1002)**

BEGIN

TransferFunds(1001, 1002, 1000);

END;

/

**Final Check: View Updated Tables**

-- View accounts

SELECT \* FROM accounts;

-- View employees

SELECT \* FROM employees;

