-- Scenario 1: Apply 1% discount to loan interest rates for customers above 60 years old

DECLARE

CURSOR cur\_customers IS

SELECT customer\_id FROM customers WHERE age > 60;

BEGIN

FOR cust IN cur\_customers LOOP

UPDATE loans

SET interest\_rate = interest\_rate - 1

WHERE customer\_id = cust.customer\_id;

END LOOP;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Discount applied to all loans for customers above 60.');

END;

/

-- Scenario 2: Set IsVIP flag to TRUE for customers with balance over $10,000

BEGIN

UPDATE customers

SET isvip = 'TRUE'

WHERE balance > 10000;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('VIP status updated for customers with balance over $10,000.');

END;

/

-- Scenario 3: Print reminders for loans due within the next 30 days

DECLARE

CURSOR cur\_due\_loans IS

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 BETWEEN SYSDATE AND SYSDATE + 30;

BEGIN

FOR rec IN cur\_due\_loans LOOP

DBMS\_OUTPUT.PUT\_LINE('Reminder: ' || rec.name || ', your loan #' || rec.loan\_id ||

' is due on ' || TO\_CHAR(rec.due\_date, 'YYYY-MM-DD') || '.');

END LOOP;

END;

/

-- Create accounts table

CREATE TABLE accounts (

account\_id NUMBER PRIMARY KEY,

account\_type VARCHAR2(20),

balance NUMBER(10,2)

);

-- Create employees table

CREATE TABLE employees (

employee\_id NUMBER PRIMARY KEY,

name VARCHAR2(50),

department\_id NUMBER,

salary NUMBER(10,2)

);

-- Insert sample accounts

INSERT INTO accounts VALUES (101, 'SAVINGS', 1000);

INSERT INTO accounts VALUES (102, 'SAVINGS', 2000);

INSERT INTO accounts VALUES (201, 'CURRENT', 5000);

INSERT INTO accounts VALUES (202, 'CURRENT', 3000);

-- Insert sample employees

INSERT INTO employees VALUES (1, 'Alice', 10, 50000);

INSERT INTO employees VALUES (2, 'Bob', 10, 60000);

INSERT INTO employees VALUES (3, 'Charlie', 20, 45000);

COMMIT;

--------------------------------------------------------------------------------

-- Scenario 1: Monthly interest for savings accounts

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

BEGIN

UPDATE accounts

SET balance = balance \* 1.01

WHERE account\_type = 'SAVINGS';

COMMIT;

END;

/

--------------------------------------------------------------------------------

-- Scenario 2: Employee bonus update

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus(

p\_department\_id IN NUMBER,

p\_bonus\_percent IN NUMBER

) IS

BEGIN

UPDATE employees

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

WHERE department\_id = p\_department\_id;

COMMIT;

END;

/

--------------------------------------------------------------------------------

-- Scenario 3: Fund transfer between accounts

CREATE OR REPLACE PROCEDURE TransferFunds(

p\_from\_account IN NUMBER,

p\_to\_account IN NUMBER,

p\_amount IN NUMBER

) IS

v\_balance NUMBER;

BEGIN

SELECT balance INTO v\_balance FROM accounts WHERE account\_id = p\_from\_account FOR UPDATE;

IF v\_balance < p\_amount THEN

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

END IF;

UPDATE accounts

SET balance = balance - p\_amount

WHERE account\_id = p\_from\_account;

UPDATE accounts

SET balance = balance + p\_amount

WHERE account\_id = p\_to\_account;

COMMIT;

END;

/

--------------------------------------------------------------------------------

-- SHOW BEFORE: Account balances

PROMPT === BEFORE: Accounts ===

SELECT \* FROM accounts;

-- Run Scenario 1

EXEC ProcessMonthlyInterest;

-- SHOW AFTER Scenario 1

PROMPT === AFTER Interest Applied ===

SELECT \* FROM accounts;

--------------------------------------------------------------------------------

-- SHOW BEFORE: Employee salaries

PROMPT === BEFORE: Employees ===

SELECT \* FROM employees;

-- Run Scenario 2

EXEC UpdateEmployeeBonus(10, 10); -- 10% bonus

-- SHOW AFTER Scenario 2

PROMPT === AFTER Bonus Update ===

SELECT \* FROM employees;

--------------------------------------------------------------------------------

-- SHOW BEFORE: Fund Transfer

PROMPT === BEFORE Transfer: Accounts ===

SELECT \* FROM accounts;

-- Run Scenario 3

EXEC TransferFunds(101, 202, 500);

-- SHOW AFTER Transfer

PROMPT === AFTER Transfer: Accounts ===

SELECT \* FROM accounts;