Stored procedures

Scenario - 1

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest

IS

CURSOR savings\_accounts IS

SELECT account\_id, balance

FROM accounts

WHERE account\_type = 'SAVINGS'

FOR UPDATE;

v\_interest\_rate NUMBER := 0.01; -- 1% interest rate

v\_interest\_amount NUMBER;

BEGIN

FOR acc IN savings\_accounts LOOP

-- Calculate interest amount

v\_interest\_amount := acc.balance \* v\_interest\_rate;

-- Update account balance

UPDATE accounts

SET balance = balance + v\_interest\_amount

WHERE CURRENT OF savings\_accounts;

-- Record transaction (optional)

INSERT INTO transactions (account\_id, transaction\_type, amount, transaction\_date)

VALUES (acc.account\_id, 'INTEREST', v\_interest\_amount, SYSDATE);

END LOOP;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Monthly interest processed successfully for all savings accounts.');

EXCEPTION

WHEN OTHERS THEN

ROLLBACK;

DBMS\_OUTPUT.PUT\_LINE('Error processing monthly interest: ' || SQLERRM);

END;

Scenario -2

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus(

p\_department\_id IN NUMBER,

p\_bonus\_percentage IN NUMBER

)

IS

CURSOR emp\_cursor IS

SELECT employee\_id, salary

FROM employees

WHERE department\_id = p\_department\_id

FOR UPDATE;

v\_bonus\_amount NUMBER;

BEGIN

FOR emp IN emp\_cursor LOOP

-- Calculate bonus amount

v\_bonus\_amount := emp.salary \* (p\_bonus\_percentage / 100);

-- Update employee salary with bonus

UPDATE employees

SET salary = salary + v\_bonus\_amount

WHERE CURRENT OF emp\_cursor;

DBMS\_OUTPUT.PUT\_LINE('Employee ID ' || emp.employee\_id ||

' received a bonus of $' || v\_bonus\_amount);

END LOOP;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Bonus update completed for department ' || p\_department\_id);

EXCEPTION

WHEN OTHERS THEN

ROLLBACK;

DBMS\_OUTPUT.PUT\_LINE('Error updating employee bonuses: ' || SQLERRM);

END;

Scenario – 3

CREATE OR REPLACE PROCEDURE TransferFunds(

p\_from\_account IN NUMBER,

p\_to\_account IN NUMBER,

p\_amount IN NUMBER

)

IS

v\_from\_balance NUMBER;

v\_to\_account\_exists NUMBER;

BEGIN

-- Check if source account has sufficient balance

SELECT balance INTO v\_from\_balance

FROM accounts

WHERE account\_id = p\_from\_account

FOR UPDATE;

IF v\_from\_balance < p\_amount THEN

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

END IF;

-- Verify destination account exists

SELECT COUNT(\*) INTO v\_to\_account\_exists

FROM accounts

WHERE account\_id = p\_to\_account;

IF v\_to\_account\_exists = 0 THEN

RAISE\_APPLICATION\_ERROR(-20002, 'Destination account does not exist');

END IF;

-- Perform the transfer

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;

-- Record transactions

INSERT INTO transactions (account\_id, transaction\_type, amount, transaction\_date, related\_account)

VALUES (p\_from\_account, 'TRANSFER\_OUT', p\_amount, SYSDATE, p\_to\_account);

INSERT INTO transactions (account\_id, transaction\_type, amount, transaction\_date, related\_account)

VALUES (p\_to\_account, 'TRANSFER\_IN', p\_amount, SYSDATE, p\_from\_account);

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Successfully transferred $' || p\_amount ||

' from account ' || p\_from\_account ||

' to account ' || p\_to\_account);

EXCEPTION

WHEN OTHERS THEN

ROLLBACK;

DBMS\_OUTPUT.PUT\_LINE('Transfer failed: ' || SQLERRM);

END;