Exercise 3: Stored Procedures

# SET SERVEROUTPUT ON; SET FEEDBACK OFF;

-- Create tables

CREATE TABLE savings\_accounts ( account\_id NUMBER PRIMARY KEY, customer\_id NUMBER,

balance NUMBER(15,2), last\_interest\_date DATE

);

CREATE TABLE employees (

employee\_id NUMBER PRIMARY KEY, first\_name VARCHAR2(50),

last\_name VARCHAR2(50), department VARCHAR2(50), salary NUMBER(10,2), performance\_rating NUMBER(2,1)

);

CREATE TABLE accounts (

account\_id NUMBER PRIMARY KEY, customer\_id NUMBER,

account\_type VARCHAR2(20), balance NUMBER(15,2), status VARCHAR2(20)

);

-- Insert sample data

INSERT INTO savings\_accounts VALUES (101, 1, 5000.00,

# ADD\_MONTHS(SYSDATE, -1));

INSERT INTO savings\_accounts VALUES (102, 2, 12000.00,

# ADD\_MONTHS(SYSDATE, -1));

INSERT INTO savings\_accounts VALUES (103, 3, 7500.00,

# ADD\_MONTHS(SYSDATE, -1));

INSERT INTO employees VALUES (1, 'Sarah', 'Johnson', 'Loans', 55000.00, 4.5);

INSERT INTO employees VALUES (2, 'Michael', 'Chen', 'Customer Service', 48000.00, 3.8);

INSERT INTO employees VALUES (3, 'David', 'Williams', 'Investments', 62000.00, 4.2);

INSERT INTO employees VALUES (4, 'Lisa', 'Brown', 'Loans', 52000.00, 4.0);

INSERT INTO accounts VALUES (1001, 1, 'Checking', 2500.00, 'Active');

INSERT INTO accounts VALUES (1002, 1, 'Savings', 5000.00, 'Active');

INSERT INTO accounts VALUES (2001, 2, 'Checking', 18000.00, 'Active');

INSERT INTO accounts VALUES (2002, 2, 'Savings', 12000.00, 'Active');

INSERT INTO accounts VALUES (3001, 3, 'Checking', 3500.00, 'Active');

# COMMIT;

-- Scenario 1: Process Monthly Interest

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest AS v\_interest\_rate NUMBER := 0.01; -- 1% monthly interest

v\_updated\_count NUMBER := 0; BEGIN

DBMS\_OUTPUT.PUT\_LINE('Processing monthly interest at ' || v\_interest\_rate \* 100 || '%...');

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

SET balance = balance \* (1 + v\_interest\_rate), last\_interest\_date = SYSDATE

WHERE account\_id = acc.account\_id;

v\_updated\_count := v\_updated\_count + 1;

DBMS\_OUTPUT.PUT\_LINE('Account ' || acc.account\_id || ': Added interest. New balance: $' ||

TO\_CHAR(acc.balance \* (1 + v\_interest\_rate), '999,999.99'));

# END LOOP;

DBMS\_OUTPUT.PUT\_LINE('Completed. Updated ' || v\_updated\_count || ' savings accounts.');

# COMMIT; EXCEPTION

WHEN OTHERS THEN ROLLBACK;

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

/

-- Scenario 2: Update Employee Bonus

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus( p\_department IN VARCHAR2,

p\_bonus\_percent IN NUMBER

# ) AS

v\_updated\_count NUMBER := 0; v\_total\_bonus NUMBER := 0;

# BEGIN

DBMS\_OUTPUT.PUT\_LINE('Updating bonuses for ' || p\_department || ' department at ' || p\_bonus\_percent || '%...');

FOR emp IN (SELECT employee\_id, salary, performance\_rating FROM employees

WHERE department = p\_department) LOOP

-- Calculate bonus based on performance (higher rating = higher bonus) DECLARE

v\_performance\_factor NUMBER := emp.performance\_rating / 5;

v\_actual\_bonus\_percent NUMBER := p\_bonus\_percent \* v\_performance\_factor;

v\_bonus\_amount NUMBER := emp.salary \* v\_actual\_bonus\_percent /

100;

# BEGIN

UPDATE employees

SET salary = salary + v\_bonus\_amount WHERE employee\_id = emp.employee\_id;

v\_updated\_count := v\_updated\_count + 1; v\_total\_bonus := v\_total\_bonus + v\_bonus\_amount;

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

': Added $' || TO\_CHAR(v\_bonus\_amount, '999,999.99') || ' (' || ROUND(v\_actual\_bonus\_percent, 2) || '%)');

# END; END LOOP;

DBMS\_OUTPUT.PUT\_LINE('Completed. Updated ' || v\_updated\_count || ' employees.');

DBMS\_OUTPUT.PUT\_LINE('Total bonus paid: $' || TO\_CHAR(v\_total\_bonus, '999,999.99'));

# COMMIT; EXCEPTION

WHEN OTHERS THEN ROLLBACK;

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

/

-- Scenario 3: Transfer Funds

CREATE OR REPLACE PROCEDURE TransferFunds(

p\_from\_account IN NUMBER, p\_to\_account IN NUMBER, p\_amount IN NUMBER, p\_result OUT VARCHAR2

# ) AS

v\_from\_balance NUMBER; v\_to\_balance NUMBER;

v\_from\_status VARCHAR2(20); v\_to\_status VARCHAR2(20);

# BEGIN

-- Check if amount is positive IF p\_amount <= 0 THEN

p\_result := 'Error: Transfer amount must be positive'; RETURN;

# END IF;

-- Get source account info BEGIN

SELECT balance, status INTO v\_from\_balance, v\_from\_status FROM accounts

WHERE account\_id = p\_from\_account; EXCEPTION

# WHEN NO\_DATA\_FOUND THEN

p\_result := 'Error: Source account not found'; RETURN;

# END;

-- Check source account status

IF v\_from\_status != 'Active' THEN

p\_result := 'Error: Source account is not active'; RETURN;

# END IF;

-- Check sufficient balance

IF v\_from\_balance < p\_amount THEN

p\_result := 'Error: Insufficient funds in source account'; RETURN;

# END IF;

-- Get destination account info BEGIN

SELECT balance, status INTO v\_to\_balance, v\_to\_status FROM accounts

WHERE account\_id = p\_to\_account; EXCEPTION

# WHEN NO\_DATA\_FOUND THEN

p\_result := 'Error: Destination account not found'; RETURN;

# END;

-- Check destination account status IF v\_to\_status != 'Active' THEN

p\_result := 'Error: Destination account is not active'; RETURN;

# 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;

# COMMIT;

p\_result := 'Success: Transfer completed. From account new balance: $' || TO\_CHAR(v\_from\_balance - p\_amount, '999,999.99') ||

', To account new balance: $' || TO\_CHAR(v\_to\_balance + p\_amount, '999,999.99');

DBMS\_OUTPUT.PUT\_LINE(p\_result); EXCEPTION

# WHEN OTHERS THEN ROLLBACK;

p\_result := 'Error during transfer: ' || SQLERRM; DBMS\_OUTPUT.PUT\_LINE(p\_result);

END TransferFunds;

/

-- Test the procedures BEGIN

DBMS\_OUTPUT.PUT\_LINE('=== TESTING ProcessMonthlyInterest ==='); ProcessMonthlyInterest();

# DBMS\_OUTPUT.PUT\_LINE(CHR(10) || '=== TESTING

UpdateEmployeeBonus ==='); UpdateEmployeeBonus('Loans', 5);

DBMS\_OUTPUT.PUT\_LINE(CHR(10) || '=== TESTING TransferFunds

===');

# DECLARE

v\_result VARCHAR2(500);

# BEGIN

TransferFunds(1001, 2001, 500, v\_result); END;

# END;

/

-- Show final data BEGIN

# DBMS\_OUTPUT.PUT\_LINE(CHR(10) || '=== FINAL SAVINGS ACCOUNT BALANCES ===');

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

DBMS\_OUTPUT.PUT\_LINE('Account ' || acc.account\_id || ': $' || TO\_CHAR(acc.balance, '999,999.99'));

# END LOOP;

DBMS\_OUTPUT.PUT\_LINE(CHR(10) || '=== FINAL EMPLOYEE SALARIES ===');

FOR emp IN (SELECT employee\_id, first\_name, last\_name, salary FROM employees ORDER BY department, employee\_id) LOOP

DBMS\_OUTPUT.PUT\_LINE(emp.employee\_id || ': ' || emp.first\_name || ' '

|| emp.last\_name ||

' - $' || TO\_CHAR(emp.salary, '999,999.99'));

# END LOOP;

DBMS\_OUTPUT.PUT\_LINE(CHR(10) || '=== FINAL ACCOUNT BALANCES ===');

FOR acc IN (SELECT account\_id, account\_type, balance FROM accounts ORDER BY account\_id) LOOP

DBMS\_OUTPUT.PUT\_LINE('Account ' || acc.account\_id || ' (' || acc.account\_type || '): $' ||

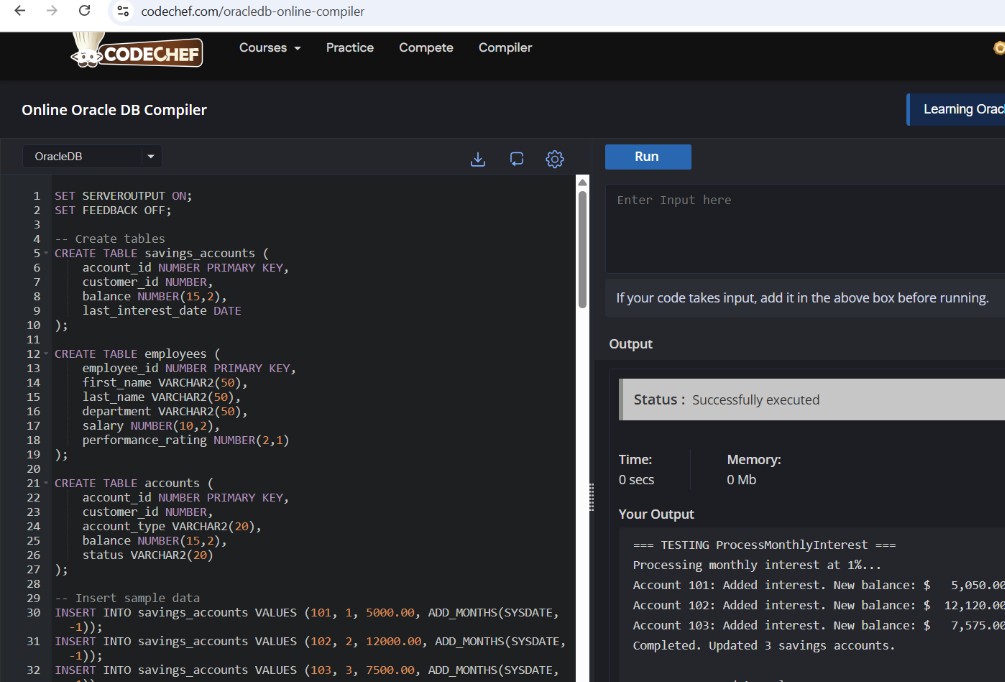
TO\_CHAR(acc.balance, '999,999.99'));

# END LOOP; END;

/

# SET FEEDBACK ON;

**OUTPUT:**

****

