**PL/SQL programming**

**Exercise 3: Stored Procedures**

--Create table accounts

CREATE TABLE accounts (

account\_id NUMBER PRIMARY KEY,

account\_holder VARCHAR2(100),

account\_type VARCHAR2(20),

balance NUMBER(10,2)

);

INSERT INTO accounts (account\_id,account\_holder, account\_type, balance) VALUES (123,'Alice', 'savings', 1000.00);

INSERT INTO accounts (account\_id,account\_holder, account\_type, balance) VALUES (678,'Bob', 'current', 2000.00);

INSERT INTO accounts (account\_id,account\_holder, account\_type, balance) VALUES (890,'Charlie', 'savings', 1500.00);

INSERT INTO accounts (account\_id,account\_holder, account\_type, balance) VALUES (346,'Diana', 'savings', 2500.00);

SELECT \* FROM accounts;

--Create table employees

CREATE TABLE employees (

employee\_id NUMBER PRIMARY KEY,

name VARCHAR2(100),

department\_id NUMBER,

salary NUMBER(10,2)

);

INSERT INTO employees (employee\_id, name, department\_id, salary) VALUES (101, 'Alice', 1, 50000.00);

INSERT INTO employees (employee\_id, name, department\_id, salary) VALUES (102, 'Bob', 2, 60000.00);

INSERT INTO employees (employee\_id, name, department\_id, salary) VALUES (103, 'Charlie', 1, 55000.00);

INSERT INTO employees (employee\_id, name, department\_id, salary) VALUES (104, 'Diana', 3, 70000.00);

SELECT \* FROM employees;

--Scenario 1:

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

BEGIN

UPDATE accounts

SET balance = balance \* 1.01

WHERE account\_type = 'savings';

END;

/

BEGIN

ProcessMonthlyInterest;

END;

/

SELECT \* FROM accounts;

SELECT \* FROM accounts WHERE account\_type = 'savings';

--Scenario 2:

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

dept\_id IN NUMBER,

bonus\_percent IN NUMBER

) IS

BEGIN

UPDATE employees

SET salary = salary + (salary \* bonus\_percent / 100)

WHERE department\_id = dept\_id;

END;

/

BEGIN

UpdateEmployeeBonus(1, 10);

END;

/

SELECT \* FROM employees;

SELECT \* FROM employees WHERE department\_id = 1;

--Scenario 3:

CREATE OR REPLACE PROCEDURE TransferFunds (

p\_source\_account\_id IN NUMBER,

p\_target\_account\_id IN NUMBER,

p\_amount IN NUMBER

) AS

v\_source\_balance NUMBER;

BEGIN

-- Step 1: Get source balance

SELECT balance

INTO v\_source\_balance

FROM accounts

WHERE account\_id = p\_source\_account\_id;

DBMS\_OUTPUT.PUT\_LINE('Source account balance: ' || v\_source\_balance);

-- Step 2: Check sufficient balance

IF v\_source\_balance < p\_amount THEN

DBMS\_OUTPUT.PUT\_LINE('Insufficient balance. Cannot transfer.');

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

END IF;

-- Step 3: Deduct from source account

UPDATE accounts

SET balance = balance - p\_amount

WHERE account\_id = p\_source\_account\_id;

DBMS\_OUTPUT.PUT\_LINE('Deducted ' || p\_amount || ' from account ' || p\_source\_account\_id);

-- Step 4: Add to target account

UPDATE accounts

SET balance = balance + p\_amount

WHERE account\_id = p\_target\_account\_id;

DBMS\_OUTPUT.PUT\_LINE('Added ' || p\_amount || ' to account ' || p\_target\_account\_id);

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Transfer completed successfully.');

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('One or both account IDs do not exist.');

RAISE\_APPLICATION\_ERROR(-20002, 'Source or target account does not exist.');

WHEN OTHERS THEN

ROLLBACK;

DBMS\_OUTPUT.PUT\_LINE('Unexpected error: ' || SQLERRM);

RAISE;

END;

/

SET SERVEROUTPUT ON;

BEGIN

TransferFunds(123, 678, 200);

END;

/

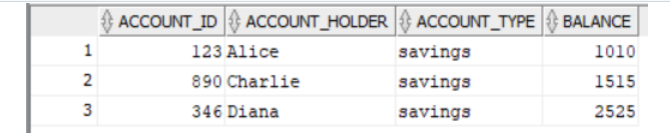
SELECT account\_id, account\_holder, balance

FROM accounts

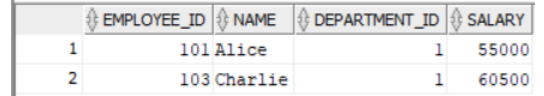
WHERE account\_id IN (123, 678);

**OUTPUT:**

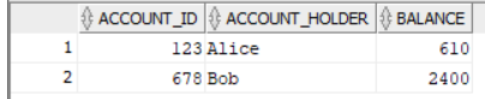
**Scenario 1:**

****

**Scenario 2:**



**Scenario 3:**

****