**Exercise 3: Stored Procedures**

**Scenario 1:** The bank needs to process monthly interest for all savings accounts.

* + **Question:** Write a stored procedure **ProcessMonthlyInterest** that calculates and updates the balance of all savings accounts by applying an interest rate of 1% to the current balance.

**Queries:**

INSERT INTO Accounts (AccountID, CustomerID, AccountType, Balance, LastModified) VALUES (3, 1, 'Savings', 10000, SYSDATE);INSERT INTO Accounts VALUES (2, 102, 2000, 'CURRENT');

INSERT INTO Accounts (AccountID, CustomerID, AccountType, Balance, LastModified) VALUES (4, 2, 'Checking', 8000, SYSDATE);

COMMIT;

**PL/SQL:**

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

BEGIN

UPDATE Accounts

SET Balance = Balance + (Balance \* 0.01),

LastModified = SYSDATE

WHERE AccountType = 'Savings';

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Monthly interest applied to all savings accounts.');

EXCEPTION

WHEN OTHERS THEN

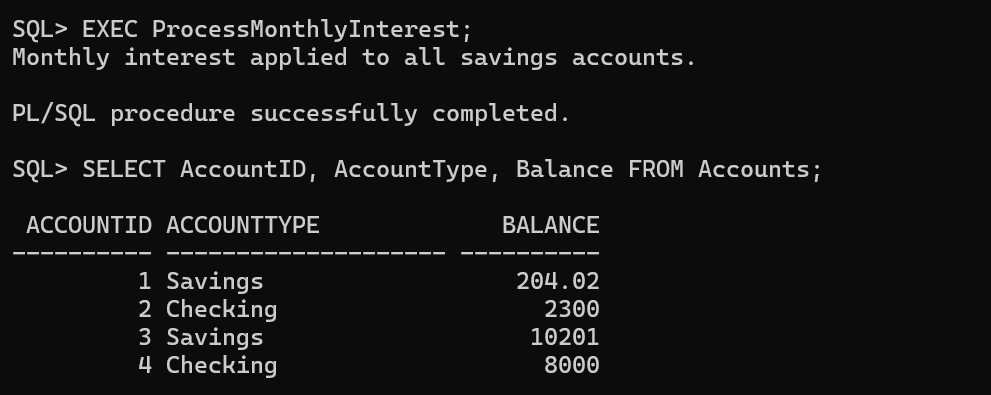
DBMS\_OUTPUT.PUT\_LINE('Error occurred: ' || SQLERRM);

ROLLBACK;

END;

/

**Output:**



**Scenario 2:**

The bank wants to implement a bonus scheme for employees based on their performance.

* + **Question:** Write a stored procedure **UpdateEmployeeBonus** that updates the salary of employees in a given department by adding a bonus percentage passed as a parameter.

**PL/SQL :**

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

p\_department IN Employees.Department%TYPE,

p\_bonus\_percent IN NUMBER

) IS

BEGIN

-- Update salary by applying bonus

UPDATE Employees

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

WHERE Department = p\_department;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Bonus of ' || p\_bonus\_percent || '% applied to department: ' || p\_department);

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Error occurred: ' || SQLERRM);

ROLLBACK;

END;

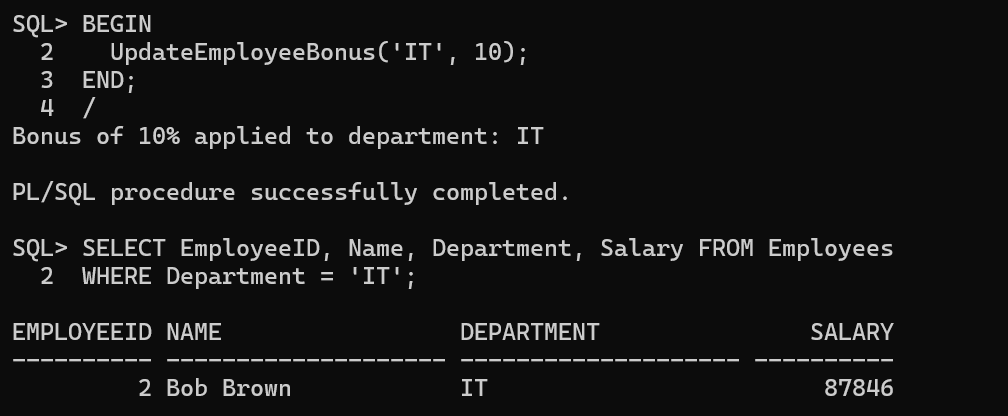
/

BEGIN

UpdateEmployeeBonus('IT', 10);

END;

/

**OutPut:**

**Scenario 3:**

Customers should be able to transfer funds between their accounts.

* + **Question:** Write a stored procedure **TransferFunds** that transfers a specified amount from one account to another, checking that the source account has sufficient balance before making the transfer.

**PL/SQL:**

CREATE OR REPLACE PROCEDURE TransferFunds (

p\_from\_account\_id IN Accounts.AccountID%TYPE,

p\_to\_account\_id IN Accounts.AccountID%TYPE,

p\_amount IN NUMBER

) IS

v\_from\_balance Accounts.Balance%TYPE;

v\_dummy NUMBER;

BEGIN

-- Lock and fetch balance of source account

SELECT Balance

INTO v\_from\_balance

FROM Accounts

WHERE AccountID = p\_from\_account\_id

FOR UPDATE;

-- Ensure destination account exists and lock it

SELECT 1

INTO v\_dummy

FROM Accounts

WHERE AccountID = p\_to\_account\_id

FOR UPDATE;

-- Check for sufficient funds

IF v\_from\_balance < p\_amount THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Insufficient funds in source account (' || p\_from\_account\_id || ')');

END IF;

-- Perform fund transfer

UPDATE Accounts

SET Balance = Balance - p\_amount,

LastModified = SYSDATE

WHERE AccountID = p\_from\_account\_id;

UPDATE Accounts

SET Balance = Balance + p\_amount,

LastModified = SYSDATE

WHERE AccountID = p\_to\_account\_id;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('₹' || p\_amount || ' transferred from Account ' || p\_from\_account\_id || ' to Account ' || p\_to\_account\_id);

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

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

ROLLBACK;

WHEN OTHERS THEN

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

ROLLBACK;

END;

/

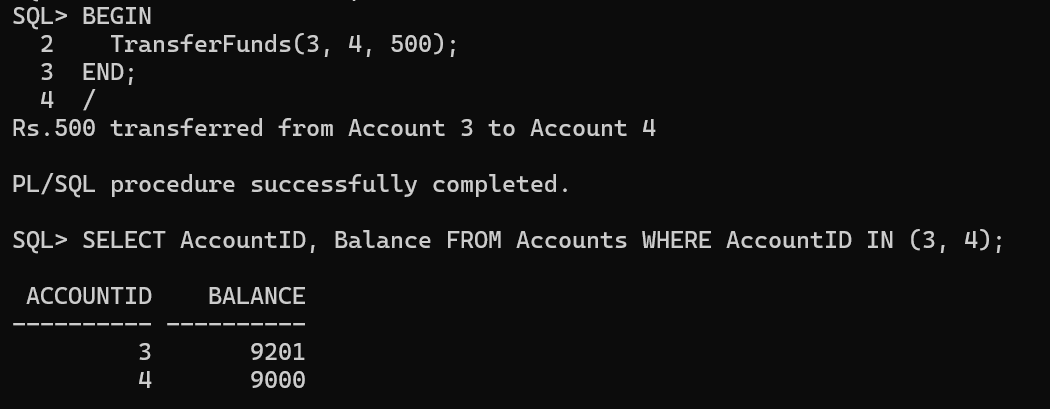
BEGIN

TransferFunds(3, 4, 500);

END;

/

**OutPut:**

****