***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.

**CODE:**

CREATE TABLE Accounts (

account\_id NUMBER PRIMARY KEY,

account\_type VARCHAR2(20),

balance NUMBER

);

-- Insert test data

INSERT INTO Accounts VALUES (101, 'Savings', 10000);

INSERT INTO Accounts VALUES (102, 'Current', 15000);

INSERT INTO Accounts VALUES (103, 'Savings', 20000);

COMMIT;

SELECT \* FROM Accounts;

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

BEGIN

UPDATE Accounts

SET balance = balance + (balance \* 0.01)

WHERE account\_type = 'Savings';

COMMIT;

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

END;

/

-- Call the procedure

BEGIN

ProcessMonthlyInterest;

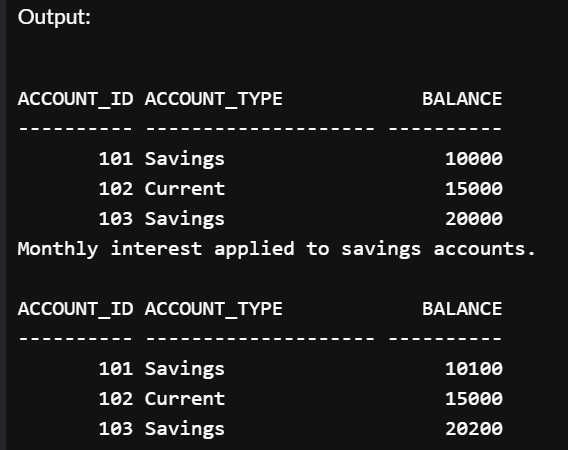
END;

/

SELECT \* FROM Accounts;

* **USING OneCompiler**

***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.

**CODE:**

CREATE TABLE Employees (

emp\_id NUMBER PRIMARY KEY,

emp\_name VARCHAR2(50),

department\_id NUMBER,

salary NUMBER

);

-- Insert sample data

INSERT INTO Employees VALUES (1, 'John', 10, 30000);

INSERT INTO Employees VALUES (2, 'Mary', 20, 40000);

INSERT INTO Employees VALUES (3, 'Alex', 10, 50000);

COMMIT;

SELECT \* FROM Employees;

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

p\_dept\_id IN NUMBER,

p\_bonus\_pct IN NUMBER

) IS

BEGIN

UPDATE Employees

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

WHERE department\_id = p\_dept\_id;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Bonus applied to department ID ' || p\_dept\_id);

END;

/

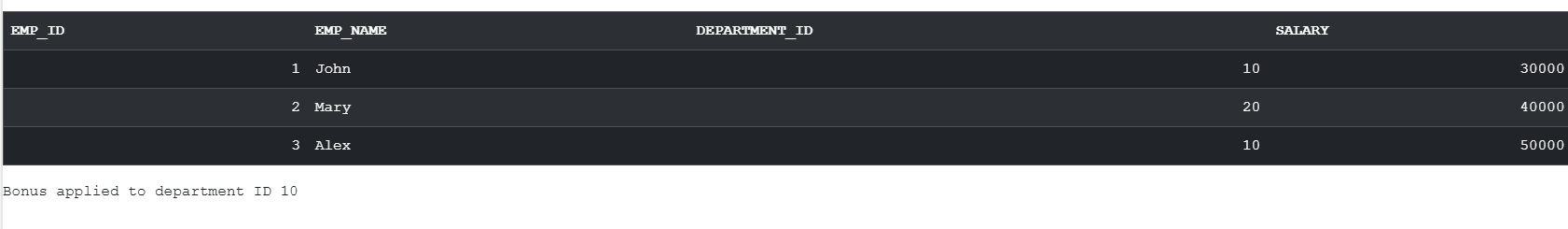
BEGIN

UpdateEmployeeBonus(10, 10); -- 10% bonus to dept 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.

**CODE:**

-- Step 1: Create table

CREATE TABLE Accounts (

account\_id NUMBER PRIMARY KEY,

account\_type VARCHAR2(20),

balance NUMBER

);

-- Step 2: Insert data

INSERT INTO Accounts VALUES (104, 'Savings', 8000);

INSERT INTO Accounts VALUES (105, 'Savings', 3000);

COMMIT;

-- Step 3: Create procedure

CREATE OR REPLACE PROCEDURE TransferFunds (

p\_from\_acc\_id IN NUMBER,

p\_to\_acc\_id IN NUMBER,

p\_amount IN NUMBER

) IS

v\_balance NUMBER;

BEGIN

-- Check balance

SELECT balance INTO v\_balance

FROM Accounts

WHERE account\_id = p\_from\_acc\_id;

IF v\_balance < p\_amount THEN

DBMS\_OUTPUT.PUT\_LINE('❌ Transfer failed: Insufficient balance.');

RETURN;

END IF;

-- Perform transfer

UPDATE Accounts

SET balance = balance - p\_amount

WHERE account\_id = p\_from\_acc\_id;

UPDATE Accounts

SET balance = balance + p\_amount

WHERE account\_id = p\_to\_acc\_id;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Transfer of ' || p\_amount || ' completed.');

END;

/

-- Step 4: Call the procedure

BEGIN

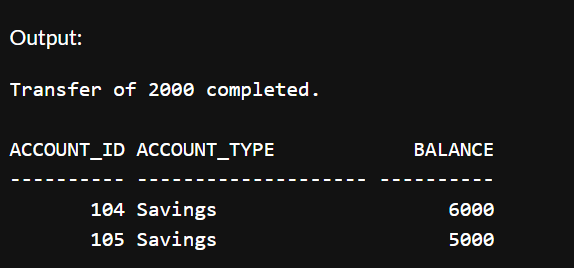
TransferFunds(104, 105, 2000);

END;

/

SELECT \* FROM Accounts;

***OUTPUT:***

******