**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:**

**-- Drop existing table**

BEGIN

EXECUTE IMMEDIATE 'DROP TABLE SavingsAccounts';

EXCEPTION

WHEN OTHERS THEN

IF SQLCODE != -942 THEN

RAISE;

END IF;

END;

/

**-- Create the table**

CREATE TABLE SavingsAccounts (

AccountID NUMBER PRIMARY KEY,

CustomerName VARCHAR2(100),

Balance NUMBER(10, 2)

);

/

**-- Insert sample data**

BEGIN

INSERT INTO SavingsAccounts VALUES (1, 'Ravi Kumar', 10000.00);

INSERT INTO SavingsAccounts VALUES (2, 'Divya Nair', 8000.00);

INSERT INTO SavingsAccounts VALUES (3, 'Sundar Iyer', 12000.00);

COMMIT;

EXCEPTION

WHEN DUP\_VAL\_ON\_INDEX THEN

NULL; -- Ignore duplicate insert if AccountID already exists

END;

/

**-- Create the stored procedure**

CREATE OR REPLACE PROCEDURE ProcessMonthlyInterest IS

BEGIN

FOR acc IN (SELECT AccountID, Balance FROM SavingsAccounts) LOOP

UPDATE SavingsAccounts

SET Balance = Balance + (Balance \* 0.01)

WHERE AccountID = acc.AccountID;

END LOOP;

COMMIT;

END;

/

**-- Run the procedure**

BEGIN

ProcessMonthlyInterest;

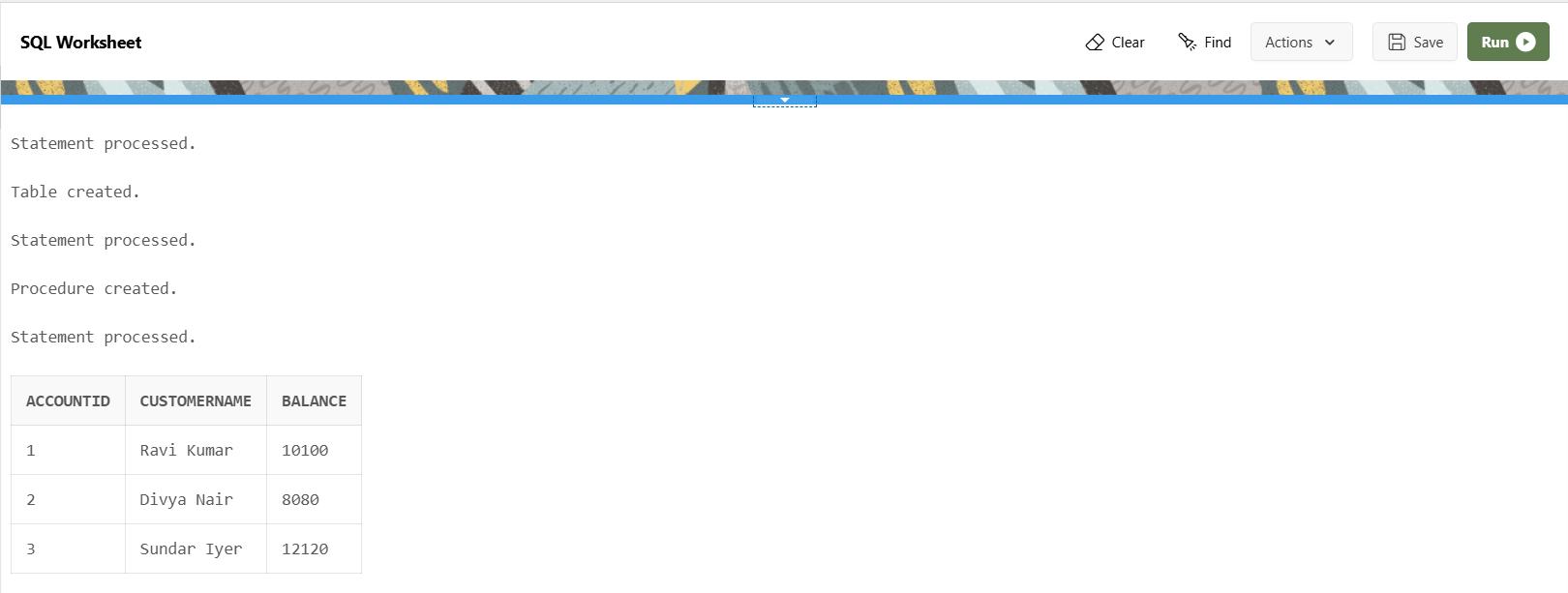
END;

/

**-- View updated balances**

SELECT \* FROM SavingsAccounts;

**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:**

**-- Drop existing table**

BEGIN

EXECUTE IMMEDIATE 'DROP TABLE Employees';

EXCEPTION

WHEN OTHERS THEN

IF SQLCODE != -942 THEN

RAISE;

END IF;

END;

/

**-- Create Employees table**

CREATE TABLE Employees (

EmpID NUMBER PRIMARY KEY,

EmpName VARCHAR2(100),

Department VARCHAR2(50),

Salary NUMBER(10, 2)

);

/

**-- Insert sample data**

BEGIN

INSERT INTO Employees VALUES (1, 'Ananya Rao', 'IT', 50000);

INSERT INTO Employees VALUES (2, 'Rahul Verma', 'Finance', 60000);

INSERT INTO Employees VALUES (3, 'Divya Sharma', 'IT', 55000);

INSERT INTO Employees VALUES (4, 'Kiran Patel', 'HR', 48000);

COMMIT;

EXCEPTION

WHEN DUP\_VAL\_ON\_INDEX THEN

NULL;

END;

/

**-- Create or replace the procedure**

CREATE OR REPLACE PROCEDURE UpdateEmployeeBonus (

p\_Department IN VARCHAR2,

p\_BonusPercent IN NUMBER

) IS

BEGIN

UPDATE Employees

SET Salary = Salary + (Salary \* p\_BonusPercent / 100)

WHERE Department = p\_Department;

COMMIT;

END;

/

**-- Run the procedure**

BEGIN

UpdateEmployeeBonus('IT', 10);

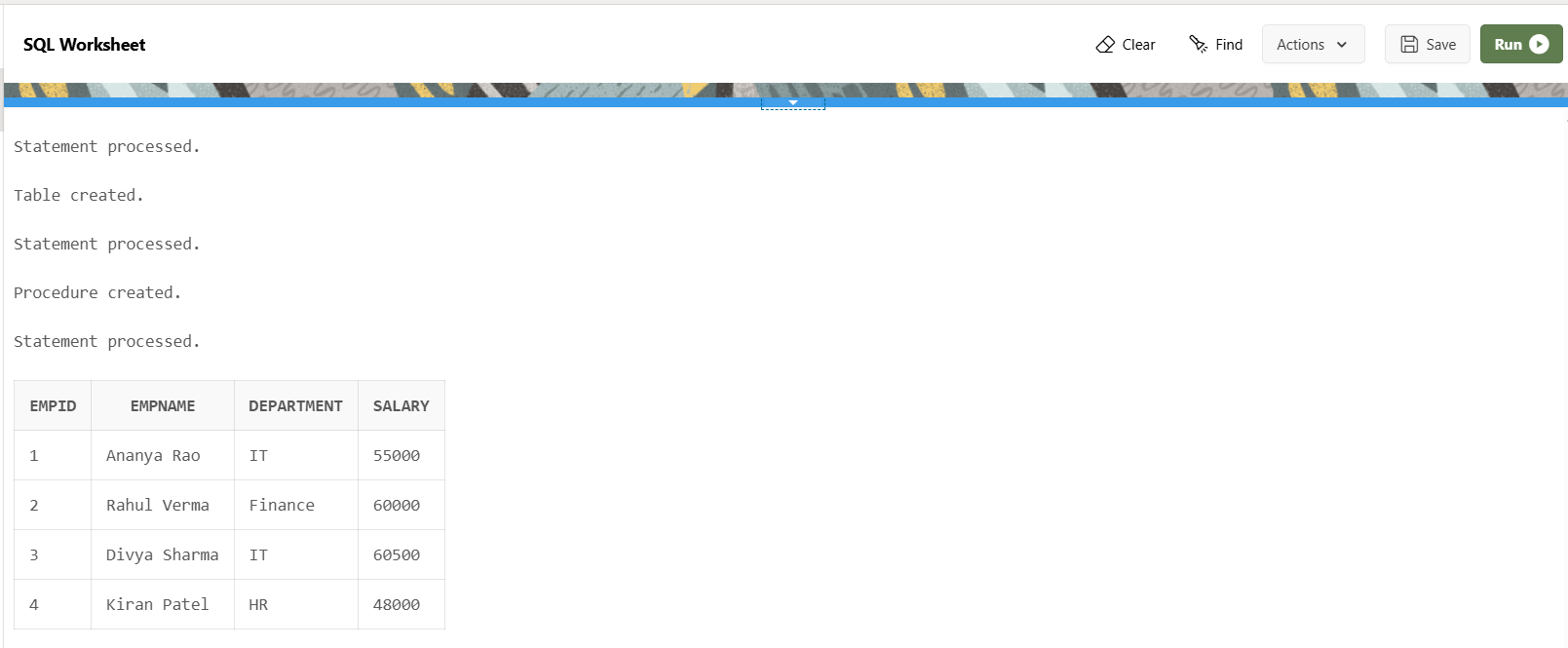
END;

/

**-- View updated salaries**

SELECT \* FROM Employees;

**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:**

**-- Drop the Accounts table if it already exists**

BEGIN

EXECUTE IMMEDIATE 'DROP TABLE Accounts';

EXCEPTION

WHEN OTHERS THEN

IF SQLCODE != -942 THEN

RAISE;

END IF;

END;

/

**-- Create the Accounts table**

CREATE TABLE Accounts (

AccountID NUMBER PRIMARY KEY,

CustomerName VARCHAR2(100),

Balance NUMBER(10, 2)

);

/

**-- Insert sample data**

BEGIN

INSERT INTO Accounts VALUES (101, 'Ravi Kumar', 5000);

INSERT INTO Accounts VALUES (102, 'Divya Nair', 3000);

INSERT INTO Accounts VALUES (103, 'Anil Singh', 2000);

COMMIT;

END;

/

**-- Create the TransferFunds stored procedure**

CREATE OR REPLACE PROCEDURE TransferFunds (

p\_FromAccountID IN NUMBER,

p\_ToAccountID IN NUMBER,

p\_Amount IN NUMBER

) IS

v\_FromBalance NUMBER;

BEGIN

**-- Check if source account exists and has sufficient balance**

SELECT Balance INTO v\_FromBalance

FROM Accounts

WHERE AccountID = p\_FromAccountID;

IF v\_FromBalance >= p\_Amount THEN

**-- Debit from source account**

UPDATE Accounts

SET Balance = Balance - p\_Amount

WHERE AccountID = p\_FromAccountID;

**-- Credit to destination account**

UPDATE Accounts

SET Balance = Balance + p\_Amount

WHERE AccountID = p\_ToAccountID;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('✅ Transfer successful: ₹' || p\_Amount ||

' from Account ' || p\_FromAccountID ||

' to Account ' || p\_ToAccountID);

ELSE

DBMS\_OUTPUT.PUT\_LINE('❌ Transfer failed: Insufficient balance in Account ' || p\_FromAccountID);

END IF;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('❌ Transfer failed: One of the account IDs does not exist.');

WHEN OTHERS THEN

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

END;

/

**-- Transfer ₹1000 from Account 101 to 102**

BEGIN

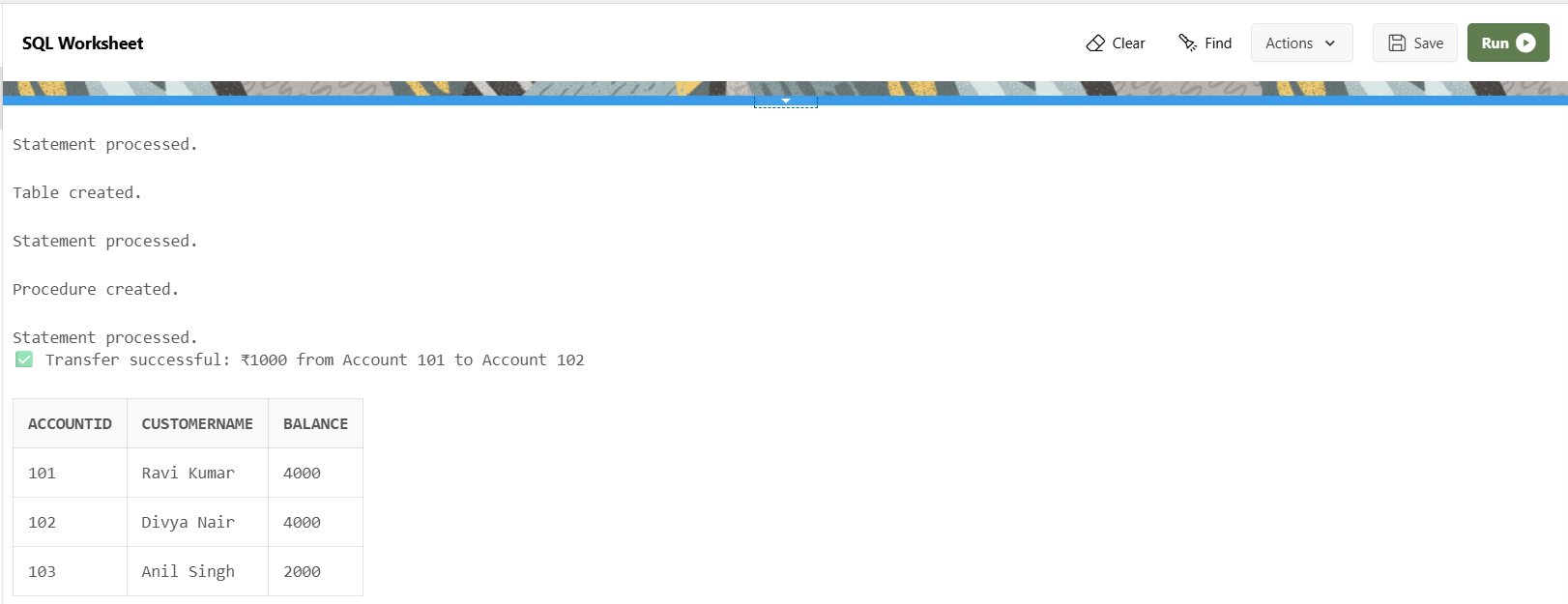
TransferFunds(101, 102, 1000);

END;

/

SELECT \* FROM Accounts;

**Output:**

****