**Exercise 2: Error Handling**

**Scenario 1:**

**Question:** Write a stored procedure **SafeTransferFunds** that transfers funds between two accounts. Ensure that if any error occurs (e.g., insufficient funds), an appropriate error message is logged and the transaction is rolled back.

CREATE OR REPLACE PROCEDURE SafeTransferFunds (

  p\_from\_account\_id IN NUMBER,

  p\_to\_account\_id   IN NUMBER,

  p\_amount          IN NUMBER

)

IS

  v\_from\_balance NUMBER;

BEGIN

  SELECT Balance INTO v\_from\_balance FROM Accounts WHERE AccountID = p\_from\_account\_id FOR UPDATE;

IF v\_from\_balance < p\_amount THEN

    RAISE\_APPLICATION\_ERROR(-20001, 'Insufficient funds');

  END IF;

UPDATE Accounts SET Balance = Balance - p\_amount WHERE AccountID = p\_from\_account\_id;

  UPDATE Accounts SET Balance = Balance + p\_amount WHERE AccountID = p\_to\_account\_id;

COMMIT;

  DBMS\_OUTPUT.PUT\_LINE('Transfer successful.');

EXCEPTION

  WHEN OTHERS THEN

    ROLLBACK;

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

END;

/

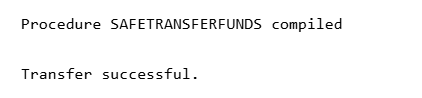
BEGIN

    SafeTransferFunds(1, 2, 500);

    END;

    /

**Output**



**Scenario 2:**

**Question:** Write a stored procedure **UpdateSalary** that increases the salary of an employee by a given percentage. If the employee ID does not exist, handle the exception and log an error message.

CREATE OR REPLACE PROCEDURE UpdateSalary (

  p\_emp\_id   IN NUMBER,

  p\_percent  IN NUMBER

)

IS

BEGIN

  UPDATE Employees

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

  WHERE EmployeeID = p\_emp\_id;

  IF SQL%ROWCOUNT = 0 THEN

    RAISE\_APPLICATION\_ERROR(-20002, 'Employee ID not found.');

  END IF;

  COMMIT;

  DBMS\_OUTPUT.PUT\_LINE('Salary updated.');

EXCEPTION

  WHEN OTHERS THEN

    ROLLBACK;

    DBMS\_OUTPUT.PUT\_LINE('Salary update failed: ' || SQLERRM);

END;

/

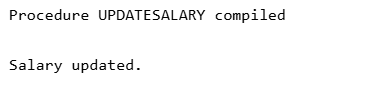
BEGIN

    UpdateSalary(1, 10);

END;

/

**Output**



**Scenario 3:**

**Question:** Write a stored procedure **AddNewCustomer** that inserts a new customer into the Customers table. If a customer with the same ID already exists, handle the exception by logging an error and preventing the insertion.

CREATE OR REPLACE PROCEDURE AddNewCustomer (

  p\_customer\_id IN NUMBER,

  p\_name        IN VARCHAR2,

  p\_dob         IN DATE,

  p\_balance     IN NUMBER

)

IS

BEGIN

  INSERT INTO Customers (CustomerID, Name, DOB, Balance, LastModified)

  VALUES (p\_customer\_id, p\_name, p\_dob, p\_balance, SYSDATE);

  COMMIT;

  DBMS\_OUTPUT.PUT\_LINE('Customer added.');

EXCEPTION

  WHEN DUP\_VAL\_ON\_INDEX THEN

    DBMS\_OUTPUT.PUT\_LINE('Customer ID already exists.');

  WHEN OTHERS THEN

    ROLLBACK;

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

END;

/

BEGIN

    AddNewCustomer(3, 'Samuel Jackson', TO\_DATE('1975-01-10','YYYY-MM-DD'), 4000);

    END;

    /

**Output**

