Exercise 2: Error Handling

Scenario 1: Handle exceptions during fund transfers between accounts.

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 IN NUMBER,
    p_to_account IN NUMBER,
    p_amount IN NUMBER
) AS
BEGIN
    BEGIN
        DECLARE
            v_balance NUMBER;
        BEGIN
            SELECT Balance INTO v_balance
            FROM Accounts
            WHERE AccountID = p_from_account;

            IF v_balance < p_amount THEN
                RAISE_APPLICATION_ERROR(-20001, 'Insufficient funds in account '
|| p_from_account);
            END IF;
        END;
        UPDATE Accounts
        SET Balance = Balance - p_amount
        WHERE AccountID = p_from_account;

        UPDATE Accounts
        SET Balance = Balance + p_amount
        WHERE AccountID = p_to_account;

        COMMIT;

    EXCEPTION
        WHEN OTHERS THEN
            ROLLBACK;
            DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
```

Scenario 2: Manage errors when updating employee salaries.

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_employee_id IN NUMBER,
    p_percentage IN NUMBER
) AS
BEGIN
    BEGIN
        UPDATE Employees
        SET Salary = Salary * (1 + p_percentage / 100)
        WHERE EmployeeID = p_employee_id;

        IF SQL%ROWCOUNT = 0 THEN
            RAISE_APPLICATION_ERROR(-20002, 'Employee ID ' || p_employee_id || '
does not exist');
        END IF;

        COMMIT;

    EXCEPTION
        WHEN OTHERS THEN
            ROLLBACK;
            DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
    END;
END UpdateSalary;
```

Scenario 3: Ensure data integrity when adding a new customer.

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
) AS
BEGIN
    BEGIN
        INSERT INTO Customers (CustomerID, Name, DOB, Balance, LastModified)
        VALUES (p_customer_id, p_name, p_dob, p_balance, SYSDATE);

        COMMIT;

    EXCEPTION
        WHEN DUP_VAL_ON_INDEX THEN
            DBMS_OUTPUT.PUT_LINE('Error: Customer ID ' || p_customer_id || '
already exists');
        WHEN OTHERS THEN
            DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
            ROLLBACK;
    END;
END AddNewCustomer;
```