**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.

**CODE**

CREATE OR REPLACE PROCEDURE SafeTransferFunds (

    from\_account\_id IN NUMBER,

    to\_account\_id IN NUMBER,

    amount IN NUMBER

)

IS

    insufficient\_funds EXCEPTION;

    funds\_balance NUMBER;

BEGIN

    -- Check balance of the from\_account

    SELECT balance INTO funds\_balance FROM Accounts WHERE account\_id = from\_account\_id FOR UPDATE;

    IF funds\_balance < amount THEN

        RAISE insufficient\_funds;

    ELSE

        -- Deduct the amount from the from\_account

        UPDATE Accounts SET balance = balance - amount WHERE account\_id = from\_account\_id;

        -- Add the amount to the to\_account

        UPDATE Accounts SET balance = balance + amount WHERE account\_id = to\_account\_id;

    END IF;

    COMMIT;

EXCEPTION

    WHEN insufficient\_funds THEN

        -- Log the error

        INSERT INTO ErrorLogs (message, log\_time) VALUES ('Insufficient funds for transfer', SYSDATE);

        ROLLBACK;

    WHEN OTHERS THEN

        -- Log any other errors

        INSERT INTO ErrorLogs (message, log\_time) VALUES (SQLERRM, SYSDATE);

        ROLLBACK;

END SafeTransferFunds;

**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

**CODE**

CREATE OR REPLACE PROCEDURE UpdateSalary (

    employee\_id IN NUMBER,

    percentage IN NUMBER

)

IS

    employee\_not\_found EXCEPTION;

    v\_count NUMBER;

BEGIN

    -- Check if the employee exists

    SELECT COUNT(\*) INTO v\_count FROM Employees WHERE id = employee\_id FOR UPDATE;

    IF v\_count = 0 THEN

        RAISE employee\_not\_found;

    ELSE

        -- Update the employee's salary

        UPDATE Employees

        SET salary = salary + (salary \* (percentage / 100))

        WHERE id = employee\_id;

    END IF;

    COMMIT;

EXCEPTION

    WHEN employee\_not\_found THEN

        -- Log the error

        INSERT INTO ErrorLogs (message, log\_time) VALUES ('Employee ID not found', SYSDATE);

        ROLLBACK;

    WHEN OTHERS THEN

        -- Log any other errors

        INSERT INTO ErrorLogs (message, log\_time) VALUES (SQLERRM, SYSDATE);

        ROLLBACK;

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.

**CODE**

CREATE OR REPLACE PROCEDURE AddNewCustomer (

    customer\_id IN NUMBER,

    customer\_name IN VARCHAR2,

    customer\_email IN VARCHAR2

)

IS

    customer\_exists EXCEPTION;

    v\_count NUMBER;

BEGIN

    -- Check if the customer already exists

    SELECT COUNT(\*) INTO v\_count FROM Customers WHERE id = customer\_id;

    IF v\_count > 0 THEN

        RAISE customer\_exists;

    ELSE

        -- Insert the new customer

        INSERT INTO Customers (id, name, email) VALUES (customer\_id, customer\_name, customer\_email);

    END IF;

    COMMIT;

EXCEPTION

    WHEN customer\_exists THEN

        -- Log the error

        INSERT INTO ErrorLogs (message, log\_time) VALUES ('Customer ID already exists', SYSDATE);

        ROLLBACK;

    WHEN OTHERS THEN

        -- Log any other errors

        INSERT INTO ErrorLogs (message, log\_time) VALUES (SQLERRM, SYSDATE);

        ROLLBACK;

END AddNewCustomer;