**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.

**PL/SQL Procedure:**

DELIMITER //

CREATE PROCEDURE SafeTransferFunds (

IN p\_FromAccountID INT,

IN p\_ToAccountID INT,

IN p\_Amount DECIMAL(10, 2)

)

BEGIN

DECLARE v\_FromBalance DECIMAL(10, 2);

DECLARE v\_ToBalance DECIMAL(10, 2);

DECLARE EXIT HANDLER FOR SQLEXCEPTION

BEGIN

ROLLBACK;

SELECT 'Error occurred, transaction rolled back' AS ErrorMessage;

END;

START TRANSACTION;

SELECT Balance INTO v\_FromBalance FROM Accounts WHERE AccountID = p\_FromAccountID FOR UPDATE;

IF v\_FromBalance < p\_Amount THEN

ROLLBACK;

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Insufficient funds in the source account.';

END IF;

SELECT Balance INTO v\_ToBalance FROM Accounts WHERE AccountID = p\_ToAccountID FOR UPDATE;

UPDATE Accounts SET Balance = Balance - p\_Amount WHERE AccountID = p\_FromAccountID;

UPDATE Accounts SET Balance = Balance + p\_Amount WHERE AccountID = p\_ToAccountID;

COMMIT;

END;

//

DELIMITER ;

**Execute each procedure using CALL statements:**

CALL SafeTransferFunds(1, 2, 200);

**To see the results of your operations, you can query the tables directly:**

SELECT \* FROM Accounts;

**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.

**PL/SQL Procedure:**

DELIMITER //

CREATE PROCEDURE UpdateSalary (

IN p\_EmployeeID INT,

IN p\_Percentage DECIMAL(5, 2)

)

BEGIN

DECLARE v\_Salary DECIMAL(10, 2);

DECLARE EXIT HANDLER FOR SQLEXCEPTION

BEGIN

SELECT 'Error occurred, operation aborted' AS ErrorMessage;

END;

START TRANSACTION;

SELECT Salary INTO v\_Salary FROM Employees WHERE EmployeeID = p\_EmployeeID;

IF v\_Salary IS NULL THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Employee ID does not exist.';

ELSE

UPDATE Employees SET Salary = Salary \* (1 + p\_Percentage / 100) WHERE EmployeeID = p\_EmployeeID;

END IF;

COMMIT;

END;

//

DELIMITER ;

**Execute each procedure using CALL statements:**

CALL UpdateSalary(1, 10);

**To see the results of your operations, you can query the tables directly:**

SELECT \* FROM Employees;

**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

**PL/SQL Procedure:**

DELIMITER //

CREATE PROCEDURE AddNewCustomer (

IN p\_CustomerID INT,

IN p\_Name VARCHAR(100),

IN p\_DOB DATE,

IN p\_Balance DECIMAL(10, 2)

)

BEGIN

DECLARE EXIT HANDLER FOR SQLEXCEPTION

BEGIN

SELECT 'Error occurred, operation aborted' AS ErrorMessage;

END;

START TRANSACTION;

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

VALUES (p\_CustomerID, p\_Name, p\_DOB, p\_Balance, CURDATE());

COMMIT;

END;

//

DELIMITER ;

**Execute each procedure using CALL statements:**

CALL AddNewCustomer(3, 'Tom Jackson', '1980-12-15', 3000);

**To see the results of your operations, you can query the tables directly:**

SELECT \* FROM Customers;