**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.

**Answer:**

DELIMITER //

CREATE PROCEDURE SafeTransferFunds(

IN from\_account INT,

IN to\_account INT,

IN amount DECIMAL(10, 2)

)

BEGIN

DECLARE sender\_balance DECIMAL(10, 2);

DECLARE EXIT HANDLER FOR SQLEXCEPTION

BEGIN

ROLLBACK;

INSERT INTO ErrorLog (ErrorMessage) VALUES (CONCAT('Error during funds transfer from account ', from\_account, ' to account ', to\_account, ' of amount ', amount));

END;

START TRANSACTION;

SELECT Balance INTO sender\_balance FROM Accounts WHERE AccountID = from\_account;

IF sender\_balance < amount THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Insufficient funds';

END IF;

UPDATE Accounts SET Balance = Balance - amount WHERE AccountID = from\_account;

UPDATE Accounts SET Balance = Balance + amount WHERE AccountID = to\_account;

COMMIT;

END //

DELIMITER ;

**Call the procedure**

CALL SafeTransferFunds(1, 2, 500);

**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.

**Answer:**

DELIMITER //

CREATE PROCEDURE UpdateSalary(

IN emp\_id INT,

IN percentage DECIMAL(5, 2)

)

BEGIN

DECLARE EXIT HANDLER FOR SQLEXCEPTION

BEGIN

ROLLBACK;

INSERT INTO ErrorLog (ErrorMessage) VALUES (CONCAT('Error updating salary for employee ID: ', emp\_id));

END;

IF (SELECT COUNT(\*) FROM Employees WHERE EmployeeID = emp\_id) = 0 THEN

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

ELSE

START TRANSACTION;

UPDATE Employees

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

WHERE EmployeeID = emp\_id;

COMMIT;

END IF;

END //

DELIMITER ;

**Call to the procedure**

CALL UpdateSalary(1, 10);

**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.

**Answer:**

DELIMITER //

CREATE PROCEDURE AddNewCustomer(

IN cust\_id INT,

IN name VARCHAR(100),

IN dob DATE,

IN balance DECIMAL(10, 2)

)

BEGIN

DECLARE EXIT HANDLER FOR SQLEXCEPTION

BEGIN

ROLLBACK;

INSERT INTO ErrorLog (ErrorMessage) VALUES (CONCAT('Error adding new customer with ID: ', cust\_id));

END;

START TRANSACTION;

IF (SELECT COUNT(\*) FROM Customers WHERE CustomerID = cust\_id) > 0 THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Customer ID already exists';

ELSE

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

VALUES (cust\_id, name, dob, balance, NOW());

COMMIT;

END IF;

END //

DELIMITER ;

**Call to the procedure**

CALL AddNewCustomer(3, 'Alice Johnson', '1982-04-23', 2000);