**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.

**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.

**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.

**Scenario-1:**

CREATE PROCEDURE SafeTransferFunds (

IN p\_from\_account\_id INT,

IN p\_to\_account\_id INT,

IN p\_amount DECIMAL(10,2)

)

BEGIN

DECLARE insufficient\_funds EXCEPTION FOR SQLSTATE '45000';

DECLARE CONTINUE HANDLER FOR insufficient\_funds

BEGIN

ROLLBACK;

INSERT INTO ErrorLog (error\_message, error\_time)

VALUES ('Insufficient funds for transfer', NOW());

END;

START TRANSACTION;

DECLARE v\_balance DECIMAL(10,2);

SELECT balance INTO v\_balance FROM Accounts WHERE account\_id = p\_from\_account\_id;

IF v\_balance < p\_amount THEN

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

ELSE

UPDATE Accounts SET balance = balance - p\_amount WHERE account\_id = p\_from\_account\_id;

UPDATE Accounts SET balance = balance + p\_amount WHERE account\_id = p\_to\_account\_id;

END IF;

COMMIT;

END //

**Scenario-2:**

DELIMITER //

CREATE PROCEDURE UpdateSalary (

IN p\_employee\_id INT,

IN p\_percentage DECIMAL(5,2)

)

BEGIN

DECLARE employee\_not\_found EXCEPTION FOR SQLSTATE '45000';

DECLARE CONTINUE HANDLER FOR employee\_not\_found

BEGIN

INSERT INTO ErrorLog (error\_message, error\_time)

VALUES (CONCAT('Employee ID ', p\_employee\_id, ' not found'), NOW());

END;

DECLARE v\_current\_salary DECIMAL(10,2);

SELECT salary INTO v\_current\_salary FROM Employees WHERE employee\_id = p\_employee\_id;

IF v\_current\_salary IS NULL THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Employee not found';

ELSE

UPDATE Employees

SET salary = salary + (salary \* p\_percentage / 100)

WHERE employee\_id = p\_employee\_id;

END IF;

END //

**Scenario-3:**

DELIMITER //

CREATE PROCEDURE AddNewCustomer (

IN p\_customer\_id INT,

IN p\_customer\_name VARCHAR(255),

IN p\_customer\_email VARCHAR(255)

)

BEGIN

DECLARE customer\_exists EXCEPTION FOR SQLSTATE '45000';

DECLARE CONTINUE HANDLER FOR customer\_exists

BEGIN

INSERT INTO ErrorLog (error\_message, error\_time)

VALUES (CONCAT('Customer ID ', p\_customer\_id, ' already exists'), NOW());

END;

DECLARE v\_existing\_customer\_id INT;

SELECT customer\_id INTO v\_existing\_customer\_id FROM Customers WHERE customer\_id = p\_customer\_id;

IF v\_existing\_customer\_id IS NOT NULL THEN

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

ELSE

INSERT INTO Customers (customer\_id, customer\_name, customer\_email)

VALUES (p\_customer\_id, p\_customer\_name, p\_customer\_email);

END IF;

END //

DELIMITER ;