**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.

-> **Solution:**

CREATE OR REPLACE PROCEDURE SafeTransferFunds (

p\_FromAccountID IN NUMBER,

p\_ToAccountID IN NUMBER,

p\_Amount IN NUMBER

)

IS

InsufficientFunds EXCEPTION;

PRAGMA EXCEPTION\_INIT(InsufficientFunds, -20001);

BEGIN

-- Check if the source account has sufficient funds

DECLARE

v\_FromAccountBalance NUMBER;

BEGIN

SELECT Balance INTO v\_FromAccountBalance

FROM Accounts

WHERE AccountID = p\_FromAccountID;

IF v\_FromAccountBalance < p\_Amount THEN

RAISE InsufficientFunds;

END IF;

END;

-- Perform the transfer

UPDATE Accounts

SET Balance = Balance - p\_Amount

WHERE AccountID = p\_FromAccountID;

UPDATE Accounts

SET Balance = Balance + p\_Amount

WHERE AccountID = p\_ToAccountID;

COMMIT;

EXCEPTION

WHEN InsufficientFunds THEN

ROLLBACK;

INSERT INTO ErrorLog (ErrorMessage, ErrorDate)

VALUES ('Insufficient funds for transfer from AccountID: ' || p\_FromAccountID, SYSDATE);

WHEN OTHERS THEN

ROLLBACK;

INSERT INTO ErrorLog (ErrorMessage, ErrorDate)

VALUES ('Error during transfer from AccountID: ' || p\_FromAccountID || ' to AccountID: ' || p\_ToAccountID || '. ' || SQLERRM, SYSDATE);

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.

-> **Solution:**

CREATE OR REPLACE PROCEDURE UpdateSalary (

p\_EmployeeID IN NUMBER,

p\_Percentage IN NUMBER

)

IS

EmployeeNotFound EXCEPTION;

PRAGMA EXCEPTION\_INIT(EmployeeNotFound, -20001);

BEGIN

-- Attempt to update the employee's salary

UPDATE Employees

SET Salary = Salary \* (1 + p\_Percentage / 100)

WHERE EmployeeID = p\_EmployeeID;

IF SQL%ROWCOUNT = 0 THEN

RAISE EmployeeNotFound;

END IF;

COMMIT;

EXCEPTION

WHEN EmployeeNotFound THEN

ROLLBACK;

INSERT INTO ErrorLog (ErrorMessage, ErrorDate)

VALUES ('Employee ID ' || p\_EmployeeID || ' not found.', SYSDATE);

WHEN OTHERS THEN

ROLLBACK;

INSERT INTO ErrorLog (ErrorMessage, ErrorDate)

VALUES ('Error updating salary for Employee ID ' || p\_EmployeeID || '. ' || SQLERRM, SYSDATE);

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.

-> **Solution:**

CREATE OR REPLACE PROCEDURE AddNewCustomer (

p\_CustomerID IN NUMBER,

p\_Name IN VARCHAR2,

p\_DOB IN DATE,

p\_Balance IN NUMBER

)

IS

CustomerAlreadyExists EXCEPTION;

PRAGMA EXCEPTION\_INIT(CustomerAlreadyExists, -20001);

BEGIN

-- Attempt to insert the new customer

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

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

COMMIT;

EXCEPTION

WHEN DUP\_VAL\_ON\_INDEX THEN

ROLLBACK;

RAISE CustomerAlreadyExists;

WHEN CustomerAlreadyExists THEN

INSERT INTO ErrorLog (ErrorMessage, ErrorDate)

VALUES ('Customer ID ' || p\_CustomerID || ' already exists.', SYSDATE);

WHEN OTHERS THEN

ROLLBACK;

INSERT INTO ErrorLog (ErrorMessage, ErrorDate)

VALUES ('Error adding customer ID ' || p\_CustomerID || '. ' || SQLERRM, SYSDATE);

END AddNewCustomer;

/