Error Logs Table:

CREATE TABLE ErrorLog (

ErrorID NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,

ErrorMessage VARCHAR2(4000),

ErrorDate DATE DEFAULT SYSDATE

);

Q1.

CREATE OR REPLACE PROCEDURE SafeTransferFunds(

p\_FromAccountID IN NUMBER,

p\_ToAccountID IN NUMBER,

p\_Amount IN NUMBER

)

IS

v\_FromBalance NUMBER;

v\_ToBalance NUMBER;

ex\_insufficient\_funds EXCEPTION;

ex\_invalid\_account EXCEPTION;

PRAGMA EXCEPTION\_INIT(ex\_insufficient\_funds, -20001);

PRAGMA EXCEPTION\_INIT(ex\_invalid\_account, -20002);

BEGIN

-- Get the balance of the from account

SELECT Balance INTO v\_FromBalance

FROM Accounts WHERE AccountID = p\_FromAccountID FOR UPDATE;

-- Get the balance of the to account

SELECT Balance INTO v\_ToBalance

FROM Accounts WHERE AccountID = p\_ToAccountID FOR UPDATE;

-- Check if sufficient funds are available

IF v\_FromBalance < p\_Amount THEN

RAISE ex\_insufficient\_funds;

END IF;

-- Deduct amount from the from account

UPDATE Accounts

SET Balance = Balance - p\_Amount, LastModified = SYSDATE

WHERE AccountID = p\_FromAccountID;

-- Add amount to the to account

UPDATE Accounts

SET Balance = Balance + p\_Amount, LastModified = SYSDATE

WHERE AccountID = p\_ToAccountID;

-- Insert transaction records

INSERT INTO Transactions (TransactionID, AccountID, TransactionDate, Amount, TransactionType)

VALUES (Transactions\_seq.NEXTVAL, p\_FromAccountID, SYSDATE, p\_Amount, 'TransOut');

INSERT INTO Transactions (TransactionID, AccountID, TransactionDate, Amount, TransactionType)

VALUES (Transactions\_seq.NEXTVAL, p\_ToAccountID, SYSDATE, p\_Amount, 'TransIn');

-- Commit the transaction

COMMIT;

EXCEPTION

WHEN ex\_insufficient\_funds THEN

ROLLBACK;

DBMS\_OUTPUT.PUT\_LINE('Error: Insufficient funds in the from account.');

WHEN NO\_DATA\_FOUND THEN

ROLLBACK;

DBMS\_OUTPUT.PUT\_LINE('Error: Invalid account ID.');

WHEN OTHERS THEN

ROLLBACK;

DBMS\_OUTPUT.PUT\_LINE('Error: ' || SQLERRM);

END SafeTransferFunds;

/

Q2.

CREATE OR REPLACE PROCEDURE UpdateSalary(

p\_EmployeeID IN NUMBER,

p\_Percentage IN NUMBER

)

IS

v\_Salary Employees.Salary%TYPE;

ex\_employee\_not\_found EXCEPTION;

PRAGMA EXCEPTION\_INIT(ex\_employee\_not\_found, -20001);

BEGIN

-- Select the current salary of the employee

SELECT Salary INTO v\_Salary

FROM Employees

WHERE EmployeeID = p\_EmployeeID

FOR UPDATE;

-- Update the salary by the given percentage

UPDATE Employees

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

WHERE EmployeeID = p\_EmployeeID;

-- Commit the transaction

COMMIT;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

ROLLBACK;

DBMS\_OUTPUT.PUT\_LINE('Error: Employee ID ' || p\_EmployeeID || ' not found.');

-- Optionally, you could log this error into an error log table

-- INSERT INTO ErrorLog (ErrorMessage, ErrorDate) VALUES ('Employee ID ' || p\_EmployeeID || ' not found.', SYSDATE);

WHEN OTHERS THEN

ROLLBACK;

DBMS\_OUTPUT.PUT\_LINE('Error: ' || SQLERRM);

END UpdateSalary;

/

Q3.

CREATE OR REPLACE PROCEDURE AddNewCustomer(

p\_CustomerID IN NUMBER,

p\_Name IN VARCHAR2,

p\_DOB IN DATE,

p\_Balance IN NUMBER

)

IS

ex\_customer\_exists EXCEPTION;

PRAGMA EXCEPTION\_INIT(ex\_customer\_exists, -1);

BEGIN

-- Try to insert the new customer

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

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

-- Commit the transaction

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Customer added successfully.');

EXCEPTION

WHEN DUP\_VAL\_ON\_INDEX THEN

ROLLBACK;

DBMS\_OUTPUT.PUT\_LINE('Error: Customer ID ' || p\_CustomerID || ' already exists.');

-- Log the error into the error log table

INSERT INTO ErrorLog (ErrorMessage)

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

END AddNewCustomer;

/