*Error Logs Table:*

CREATE TABLE ErrorLog (

ErrorID NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,

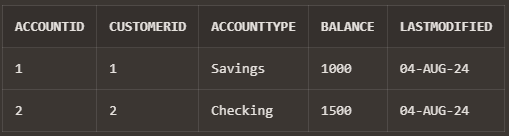
ErrorMessage VARCHAR2(4000),

ErrorDate DATE DEFAULT SYSDATE

);

Q1.

*Initial Accounts Table:*



*PL/SQL Block:*

CREATE OR REPLACE PROCEDURE SafeTransferFunds(

sourceAccountID IN NUMBER,

destinationAccountID IN NUMBER,

amount IN NUMBER

)

IS

-- Variables to hold account balances

sourceBalance NUMBER;

destinationBalance NUMBER;

-- Custom exception for insufficient funds

insufficient\_funds EXCEPTION;

BEGIN

-- Retrieve the source account balance

SELECT Balance INTO sourceBalance FROM Accounts WHERE AccountID = sourceAccountID FOR UPDATE;

-- Check if the source account has sufficient funds

IF sourceBalance < amount THEN

RAISE insufficient\_funds;

END IF;

-- Retrieve the destination account balance

SELECT Balance INTO destinationBalance FROM Accounts WHERE AccountID = destinationAccountID FOR UPDATE;

-- Deduct the amount from the source account

UPDATE Accounts

SET Balance = Balance - amount

WHERE AccountID = sourceAccountID;

-- Add the amount to the destination account

UPDATE Accounts

SET Balance = Balance + amount

WHERE AccountID = destinationAccountID;

-- Commit the transaction

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Transfer completed successfully.');

EXCEPTION

WHEN insufficient\_funds THEN

ROLLBACK;

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

WHEN NO\_DATA\_FOUND THEN

ROLLBACK;

DBMS\_OUTPUT.PUT\_LINE('Error: One of the accounts does not exist.');

WHEN OTHERS THEN

ROLLBACK;

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

END SafeTransferFunds;

/

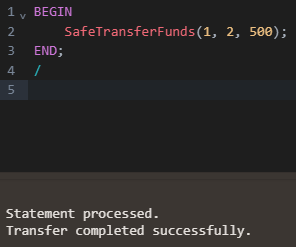
*Output:*

* Successful Transfer:

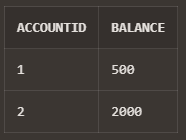
Initial Accounts Table:



Amount Transfer:

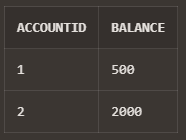


Output:

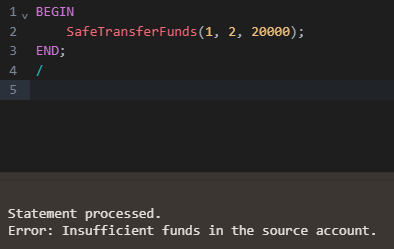


* Insufficient Funds Transfer:

Initial:

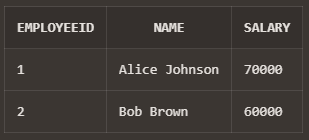


Transfer Fail:



Q2.

*Initial Employees Table:*



*PL/SQL Block:*

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;

/

*Output:*



Q3.

*Initial Customer Table:*



*PL/SQL Block:*

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;

/

*Output:*

