**Error Handling**

**Scenario 1: Safe Transfer Funds**

**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.

CREATE OR REPLACE PROCEDURE SafeTransferFunds(

p\_fromAccountID IN NUMBER,

p\_toAccountID IN NUMBER,

p\_amount IN NUMBER

) IS

v\_fromBalance Accounts.Balance%TYPE;

v\_toBalance Accounts.Balance%TYPE;

INSUFFICIENT\_FUNDS EXCEPTION;

BEGIN

SELECT Balance INTO v\_fromBalance FROM Accounts WHERE AccountID = p\_fromAccountID FOR UPDATE;

SELECT Balance INTO v\_toBalance FROM Accounts WHERE AccountID = p\_toAccountID FOR UPDATE;

IF v\_fromBalance < p\_amount THEN

RAISE INSUFFICIENT\_FUNDS;

END IF;

UPDATE Accounts SET Balance = Balance - p\_amount WHERE AccountID = p\_fromAccountID;

UPDATE Accounts SET Balance = Balance + p\_amount WHERE AccountID = p\_toAccountID;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Transfer successful from Account ' || p\_fromAccountID || ' to ' || p\_toAccountID);

EXCEPTION

WHEN INSUFFICIENT\_FUNDS THEN

ROLLBACK;

DBMS\_OUTPUT.PUT\_LINE('Error: Insufficient funds in account ' || p\_fromAccountID);

WHEN OTHERS THEN

ROLLBACK;

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

END;  
  
**Scenario 2: Update Employee Salary**

**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.

CREATE OR REPLACE PROCEDURE UpdateSalary(

p\_employeeID IN NUMBER,

p\_percentage IN NUMBER

) IS

v\_salary Employees.Salary%TYPE;

EMPLOYEE\_NOT\_FOUND EXCEPTION;

BEGIN

SELECT Salary INTO v\_salary FROM Employees WHERE EmployeeID = p\_employeeID FOR UPDATE;

IF SQL%NOTFOUND THEN

RAISE EMPLOYEE\_NOT\_FOUND;

END IF;

UPDATE Employees SET Salary = Salary \* (1 + p\_percentage / 100) WHERE EmployeeID = p\_employeeID;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Salary updated for Employee ID: ' || p\_employeeID);

EXCEPTION

WHEN EMPLOYEE\_NOT\_FOUND THEN

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

WHEN OTHERS THEN

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

END;  
  
**Scenario 3: Add New Customer with Integrity**

**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.  
  
CREATE OR REPLACE PROCEDURE AddNewCustomer(

p\_customerID IN NUMBER,

p\_name IN VARCHAR2,

p\_dob IN DATE,

p\_balance IN NUMBER

) IS

DUP\_VAL\_ON\_INDEX EXCEPTION;

BEGIN

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

VALUES (p\_customerID, p\_name, p\_dob, p\_balance, SYSDATE);

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

EXCEPTION

WHEN DUP\_VAL\_ON\_INDEX THEN

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

WHEN OTHERS THEN

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

END;