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.**

| CREATE OR REPLACE PROCEDURE SafeTransferFunds (  p\_from\_account IN NUMBER,  p\_to\_account IN NUMBER,  p\_amount IN NUMBER ) AS BEGIN  BEGIN  DECLARE  v\_balance NUMBER;  BEGIN  SELECT Balance INTO v\_balance  FROM Accounts  WHERE AccountID = p\_from\_account;    IF v\_balance < p\_amount THEN  RAISE\_APPLICATION\_ERROR(-20001, 'Insufficient funds in account ' || p\_from\_account);  END IF;  END;  UPDATE Accounts  SET Balance = Balance - p\_amount  WHERE AccountID = p\_from\_account;    UPDATE Accounts  SET Balance = Balance + p\_amount  WHERE AccountID = p\_to\_account;    COMMIT;    EXCEPTION  WHEN OTHERS THEN  ROLLBACK;  DBMS\_OUTPUT.PUT\_LINE('Error: ' || SQLERRM);  END; 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.**

| CREATE OR REPLACE PROCEDURE UpdateSalary (  p\_employee\_id IN NUMBER,  p\_percentage IN NUMBER ) AS BEGIN  BEGIN  UPDATE Employees  SET Salary = Salary \* (1 + p\_percentage / 100)  WHERE EmployeeID = p\_employee\_id;    IF SQL%ROWCOUNT = 0 THEN  RAISE\_APPLICATION\_ERROR(-20002, 'Employee ID ' || p\_employee\_id || ' does not exist');  END IF;    COMMIT;    EXCEPTION  WHEN OTHERS THEN  ROLLBACK;  DBMS\_OUTPUT.PUT\_LINE('Error: ' || SQLERRM);  END; 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.**

| CREATE OR REPLACE PROCEDURE AddNewCustomer (  p\_customer\_id IN NUMBER,  p\_name IN VARCHAR2,  p\_dob IN DATE,  p\_balance IN NUMBER ) AS BEGIN  BEGIN  INSERT INTO Customers (CustomerID, Name, DOB, Balance, LastModified)  VALUES (p\_customer\_id, p\_name, p\_dob, p\_balance, SYSDATE);    COMMIT;    EXCEPTION  WHEN DUP\_VAL\_ON\_INDEX THEN  DBMS\_OUTPUT.PUT\_LINE('Error: Customer ID ' || p\_customer\_id || ' already exists');  WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LINE('Error: ' || SQLERRM);  ROLLBACK;  END; END AddNewCustomer; |
| --- |