**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.

**Procedure:**SQL> CREATE OR REPLACE PROCEDURE SafeTransferFunds (

2 p\_FromAccountID IN NUMBER,

3 p\_ToAccountID IN NUMBER,

4 p\_Amount IN NUMBER

5 ) AS

6 v\_FromBalance NUMBER;

7 v\_ToBalance NUMBER;

8 BEGIN

9 -- Lock the accounts to prevent concurrency issues

10 SELECT Balance INTO v\_FromBalance FROM Accounts WHERE AccountID = p\_FromAccountID FOR UPDATE;

11 SELECT Balance INTO v\_ToBalance FROM Accounts WHERE AccountID = p\_ToAccountID FOR UPDATE;

12

13 -- Check if sufficient funds are available

14 IF v\_FromBalance < p\_Amount THEN

15 RAISE\_APPLICATION\_ERROR(-20001, 'Insufficient funds for transfer.');

16 END IF;

17

18 -- Perform the transfer

19 UPDATE Accounts SET Balance = Balance - p\_Amount WHERE AccountID = p\_FromAccountID;

20 UPDATE Accounts SET Balance = Balance + p\_Amount WHERE AccountID = p\_ToAccountID;

21

22 -- Commit the transaction

23 COMMIT;

24 EXCEPTION

25 WHEN OTHERS THEN

26 -- Rollback the transaction in case of error

27 ROLLBACK;

28 -- Log the error message

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

30 END SafeTransferFunds;

31 /

**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.

**Procedure:**SQL> CREATE OR REPLACE PROCEDURE UpdateSalary (

2 p\_EmployeeID IN NUMBER,

3 p\_Percentage IN NUMBER

4 ) AS

5 BEGIN

6 -- Update the salary

7 UPDATE Employees

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

9 WHERE EmployeeID = p\_EmployeeID;

10

11 -- Check if the update affected any rows

12 IF SQL%ROWCOUNT = 0 THEN

13 RAISE\_APPLICATION\_ERROR(-20002, 'Employee ID does not exist.');

14 END IF;

15

16 -- Commit the transaction

17 COMMIT;

18 EXCEPTION

19 WHEN OTHERS THEN

20 -- Rollback the transaction in case of error

21 ROLLBACK;

22 -- Log the error message

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

24 END UpdateSalary;

25 /

**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.

**Procedure:**

SQL> CREATE OR REPLACE PROCEDURE AddNewCustomer (

2 p\_CustomerID IN NUMBER,

3 p\_Name IN VARCHAR2,

4 p\_DOB IN DATE,

5 p\_Balance IN NUMBER

6 ) AS

7 v\_Count NUMBER;

8 BEGIN

9 -- Check if customer ID already exists

10 SELECT COUNT(\*)

11 INTO v\_Count

12 FROM Customers

13 WHERE CustomerID = p\_CustomerID;

14

15 IF v\_Count > 0 THEN

16 RAISE\_APPLICATION\_ERROR(-20003, 'Customer ID already exists.');

17 END IF;

18

19 -- Insert the new customer

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

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

22

23 -- Commit the transaction

24 COMMIT;

25 EXCEPTION

26 WHEN OTHERS THEN

27 -- Rollback the transaction in case of error

28 ROLLBACK;

29 -- Log the error message

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

31 END AddNewCustomer;

32 /