**Scenario 1: Handle Exceptions During Fund Transfers**

CREATE OR REPLACE PROCEDURE SafeTransferFunds (

p\_from\_account\_id IN NUMBER,

p\_to\_account\_id IN NUMBER,

p\_amount IN NUMBER

) AS

v\_from\_balance NUMBER;

v\_to\_balance NUMBER;

BEGIN

-- Check if both accounts exist and get current balances

SELECT balance INTO v\_from\_balance

FROM accounts

WHERE account\_id = p\_from\_account\_id;

SELECT balance INTO v\_to\_balance

FROM accounts

WHERE account\_id = p\_to\_account\_id;

-- Ensure sufficient funds

IF v\_from\_balance < p\_amount THEN

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

END IF;

-- Begin transaction

BEGIN

-- Transfer funds

UPDATE accounts

SET balance = balance - p\_amount

WHERE account\_id = p\_from\_account\_id;

UPDATE accounts

SET balance = balance + p\_amount

WHERE account\_id = p\_to\_account\_id;

-- Commit the transaction

COMMIT;

EXCEPTION

WHEN OTHERS THEN

-- Rollback transaction and log error

ROLLBACK;

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

RAISE;

END;

END SafeTransferFunds;

/

**Scenario 2: Manage Errors When Updating Employee Salaries**

CREATE OR REPLACE PROCEDURE UpdateSalary (

p\_employee\_id IN NUMBER,

p\_percentage IN NUMBER

) AS

v\_current\_salary NUMBER;

BEGIN

-- Get current salary

SELECT salary INTO v\_current\_salary

FROM employees

WHERE employee\_id = p\_employee\_id;

-- Update salary

UPDATE employees

SET salary = salary \* (1 + p\_percentage / 100)

WHERE employee\_id = p\_employee\_id;

-- Commit the transaction

COMMIT;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

-- Handle case where employee ID does not exist

DBMS\_OUTPUT.PUT\_LINE('Error: Employee ID ' || p\_employee\_id || ' does not exist.');

WHEN OTHERS THEN

-- Handle other exceptions

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

ROLLBACK;

END UpdateSalary;

/

**Scenario 3: Ensure Data Integrity When Adding a New Customer**

CREATE OR REPLACE PROCEDURE AddNewCustomer (

p\_customer\_id IN NUMBER,

p\_name IN VARCHAR2,

p\_balance IN NUMBER

) AS

BEGIN

-- Attempt to insert a new customer

BEGIN

INSERT INTO customers (customer\_id, name, balance)

VALUES (p\_customer\_id, p\_name, p\_balance);

-- Commit the transaction

COMMIT;

EXCEPTION

WHEN DUP\_VAL\_ON\_INDEX THEN

-- Handle case where customer ID already exists

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

WHEN OTHERS THEN

-- Handle other exceptions

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

ROLLBACK;

END;

END AddNewCustomer; /