**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.

**CODE:**

DECLARE

from\_acc NUMBER := 101;

to\_acc NUMBER := 102;

amount NUMBER := 200;

bal NUMBER;

BEGIN

SELECT balance INTO bal FROM accounts WHERE accountid = from\_acc;

IF bal < amount THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Not enough balance');

END IF;

UPDATE accounts SET balance = balance - amount WHERE accountid = from\_acc;

UPDATE accounts SET balance = balance + amount WHERE accountid = to\_acc;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Transfer successful.');

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

ROLLBACK;

DBMS\_OUTPUT.PUT\_LINE('Account not found.');

WHEN OTHERS THEN

ROLLBACK;

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

END;

/

**OUTPUT:**

****

**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.

**CODE:**

DECLARE

emp\_id NUMBER := 1;

percent NUMBER := 10;

current\_salary NUMBER;

BEGIN

SELECT salary INTO current\_salary FROM employees WHERE employee\_id = emp\_id;

UPDATE employees

SET salary = current\_salary + (current\_salary \* percent / 100)

WHERE employee\_id = emp\_id;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Salary updated successfully.');

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Employee not found.');

WHEN OTHERS THEN

ROLLBACK;

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

END;

/

**OUTPUT:**

****

**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.

**CODE:**

DECLARE

cust\_id NUMBER := 1;

cust\_name VARCHAR2(100) := 'Priya';

BEGIN

INSERT INTO customers(customer\_id, name)

VALUES(cust\_id, cust\_name);

COMMIT;

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

EXCEPTION

WHEN DUP\_VAL\_ON\_INDEX THEN

DBMS\_OUTPUT.PUT\_LINE('Customer ID already exists.');

WHEN OTHERS THEN

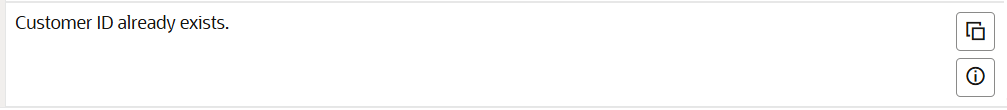
ROLLBACK;

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

END;

/

**OUTPUT:**

****