**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 Customers Table

-- Recreate table

CREATE TABLE accounts (

acc\_no NUMBER PRIMARY KEY,

holder\_name VARCHAR2(100),

balance NUMBER

);

-- Insert fresh data

INSERT INTO accounts VALUES (101, 'Koushika', 5000);

INSERT INTO accounts VALUES (102, 'Madhu', 3000);

COMMIT;

PL/SQL code;

CREATE OR REPLACE PROCEDURE AddNewCustomer(

id IN NUMBER,

cname IN VARCHAR2,

cage IN NUMBER,

cbalance IN NUMBER

) IS

BEGIN

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

VALUES (id, cname, cage, cbalance, 'FALSE');

COMMIT;

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

EXCEPTION

WHEN DUP\_VAL\_ON\_INDEX THEN

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

WHEN OTHERS THEN

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

END;

/

|  |
| --- |
|  |
|  |
| Table creation |
|  |
| PLSQL |
|  |

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.

CREATE TABLE employees (

emp\_id NUMBER PRIMARY KEY,

emp\_name VARCHAR2(100),

salary NUMBER

);  
INSERT INTO employees VALUES (201, 'Siva', 40000);

COMMIT;  
  
CREATE OR REPLACE PROCEDURE UpdateSalary(

emp\_id\_in IN NUMBER,

percent IN NUMBER

) IS

BEGIN

UPDATE employees

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

WHERE emp\_id = emp\_id\_in;

IF SQL%ROWCOUNT = 0 THEN

RAISE\_APPLICATION\_ERROR(-20003, 'Employee ID not found.');

END IF;

COMMIT;

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

EXCEPTION

WHEN OTHERS THEN

ROLLBACK;

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

END;

/  
  
EXEC UpdateSalary(201, 10);

|  |
| --- |
| TABLE CREATION |
|  |
| PLSQL |
|  |
|  |
|  |

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.

CREATE TABLE customers (

customer\_id NUMBER PRIMARY KEY,

name VARCHAR2(100),

age NUMBER,

balance NUMBER,

IsVIP VARCHAR2(5)

);  
  
CREATE OR REPLACE PROCEDURE AddNewCustomer(

id IN NUMBER,

cname IN VARCHAR2,

cage IN NUMBER,

cbalance IN NUMBER

) IS

BEGIN

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

VALUES (id, cname, cage, cbalance, 'FALSE');

COMMIT;

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

EXCEPTION

WHEN DUP\_VAL\_ON\_INDEX THEN

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

WHEN OTHERS THEN

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

END;

/  
  
EXEC AddNewCustomer(1, 'Ajay', 35, 7000);

|  |
| --- |
|  |
|  |
| TABLE CREATION |
|  |
| PLSQL |
|  |

OUTPUT