**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.

**QUERY:**

CREATE OR REPLACE PROCEDURE SafeTransferFunds (

sender\_account\_id IN NUMBER,

receiver\_account\_id IN NUMBER,

amount IN NUMBER

) AS

insufficient\_funds EXCEPTION;

PRAGMA EXCEPTION\_INIT(insufficient\_funds, -20000);

BEGIN

-- Check if the sender has sufficient balance

DECLARE

sender\_balance NUMBER;

BEGIN

SELECT balance INTO sender\_balance

FROM Accounts

WHERE account\_id = sender\_account\_id

FOR UPDATE;

IF sender\_balance < amount THEN

RAISE insufficient\_funds;

END IF;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

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

WHEN insufficient\_funds THEN

DBMS\_OUTPUT.PUT\_LINE('Insufficient funds in the sender account.');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('An error occurred while checking sender balance.');

END;

-- Transfer funds if sender has sufficient balance

IF sender\_balance >= amount THEN

UPDATE Accounts

SET balance = balance - amount

WHERE account\_id = sender\_account\_id;

UPDATE Accounts

SET balance = balance + amount

WHERE account\_id = receiver\_account\_id;

COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Funds transferred successfully.');

END IF;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

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

WHEN OTHERS THEN

ROLLBACK;

DBMS\_OUTPUT.PUT\_LINE('An error occurred during fund transfer. Transaction rolled back.');

END SafeTransferFunds;

**Explanation:**

We first check if the sender has sufficient balance, and if not, an insufficient\_funds exception is raised. If the sender has enough balance, the funds are transferred from the sender to the receiver account. Error handling is implemented to manage scenarios like account not found, insufficient funds, and any other unexpected errors. The transaction is committed if successful, and rolled back in case of errors to maintain data integrity.

**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.

**QUERY:**

CREATE OR REPLACE PROCEDURE UpdateSalary (

emp\_id IN NUMBER,

increase\_percentage IN NUMBER

) AS

employee\_not\_found EXCEPTION;

PRAGMA EXCEPTION\_INIT(employee\_not\_found, -20001);

BEGIN

-- Check if the employee exists

DECLARE

current\_salary NUMBER;

BEGIN

SELECT salary INTO current\_salary

FROM Employees

WHERE employee\_id = emp\_id;

IF current\_salary IS NULL THEN

RAISE employee\_not\_found;

END IF;

EXCEPTION

WHEN employee\_not\_found THEN

DBMS\_OUTPUT.PUT\_LINE('Employee with ID ' || emp\_id || ' not found.');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('An error occurred while checking employee details.');

END;

-- Update the employee's salary if the employee exists

IF current\_salary IS NOT NULL THEN

UPDATE Employees

SET salary = salary + (salary \* increase\_percentage / 100)

WHERE employee\_id = emp\_id;

COMMIT;

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

END IF;

EXCEPTION

WHEN OTHERS THEN

ROLLBACK;

DBMS\_OUTPUT.PUT\_LINE('An error occurred during salary update. Transaction rolled back.');

END UpdateSalary;

**Explanation:**

We first check if the employee exists based on the provided employee ID. If the employee is not found, we raise the employee\_not\_found exception. If the employee exists, we update the employee's salary by increasing it by the specified percentage. Error handling is implemented to manage scenarios like the employee not being found and any other unexpected errors. The transaction is committed if successful, and rolled back in case of errors to maintain data consistency.

**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.

**QUERY:**

CREATE OR REPLACE PROCEDURE AddNewCustomer (

customer\_id IN NUMBER,

customer\_name IN VARCHAR2,

customer\_email IN VARCHAR2

) AS

customer\_exists EXCEPTION;

PRAGMA EXCEPTION\_INIT(customer\_exists, -20001);

BEGIN

-- Check if a customer with the same ID already exists

DECLARE

existing\_customer\_id NUMBER;

BEGIN

SELECT customer\_id INTO existing\_customer\_id

FROM Customers

WHERE customer\_id = customer\_id;

IF existing\_customer\_id IS NOT NULL THEN

RAISE customer\_exists;

END IF;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

NULL; -- No action needed if no existing customer found

WHEN customer\_exists THEN

DBMS\_OUTPUT.PUT\_LINE('Customer with ID ' || customer\_id || ' already exists. Insertion prevented.');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('An error occurred while checking existing customers.');

END;

-- Insert the new customer if no existing customer with the same ID

IF existing\_customer\_id IS NULL THEN

INSERT INTO Customers (customer\_id, customer\_name, customer\_email)

VALUES (customer\_id, customer\_name, customer\_email);

COMMIT;

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

END IF;

EXCEPTION

WHEN OTHERS THEN

ROLLBACK;

DBMS\_OUTPUT.PUT\_LINE('An error occurred during customer insertion. Transaction rolled back.');

END AddNewCustomer;

**Explanation:**

We first check if a customer with the same ID already exists in the Customers table. If a customer is found, we raise the customer\_exists exception. If no existing customer is found, we proceed with inserting the new customer into the table. Error handling is implemented to manage scenarios like duplicate customer IDs and any other unexpected errors. The transaction is committed if the insertion is successful, and rolled back in case of errors to maintain data integrity.