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

Here, we'll write a stored procedure SafeTransferFunds that transfers funds between two accounts. If any error occurs (like insufficient funds), an error message will be logged, and the transaction will be rolled back.

CREATE OR REPLACE PROCEDURE SafeTransferFunds(

p\_from\_account\_id IN NUMBER,

p\_to\_account\_id IN NUMBER,

p\_amount IN NUMBER

) IS

insufficient\_funds EXCEPTION;

PRAGMA EXCEPTION\_INIT(insufficient\_funds, -20001);

BEGIN

-- Check if the source account has sufficient funds

DECLARE

v\_balance NUMBER;

BEGIN

SELECT balance INTO v\_balance FROM accounts WHERE account\_id = p\_from\_account\_id FOR UPDATE;

IF v\_balance < p\_amount THEN

RAISE insufficient\_funds;

END IF;

-- Perform the transfer

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;

EXCEPTION

WHEN insufficient\_funds THEN

ROLLBACK;

INSERT INTO error\_logs(message, log\_date)

VALUES ('Insufficient funds in account ' || p\_from\_account\_id, SYSDATE);

RAISE\_APPLICATION\_ERROR(-20001, 'Insufficient funds in the source account.');

WHEN OTHERS THEN

ROLLBACK;

INSERT INTO error\_logs(message, log\_date)

VALUES (SQLERRM, SYSDATE);

RAISE;

END;

END SafeTransferFunds;

/

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

This procedure, UpdateSalary, will increase the salary of an employee by a given percentage. If the employee ID does not exist, it will handle the exception and log an error message.

CREATE OR REPLACE PROCEDURE UpdateSalary(

p\_employee\_id IN NUMBER,

p\_percentage IN NUMBER

) IS

employee\_not\_found EXCEPTION;

PRAGMA EXCEPTION\_INIT(employee\_not\_found, -20002);

BEGIN

DECLARE

v\_salary NUMBER;

BEGIN

SELECT salary INTO v\_salary FROM employees WHERE employee\_id = p\_employee\_id FOR UPDATE;

-- Update the salary

UPDATE employees

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

WHERE employee\_id = p\_employee\_id;

COMMIT;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RAISE employee\_not\_found;

WHEN employee\_not\_found THEN

INSERT INTO error\_logs(message, log\_date)

VALUES ('Employee ID ' || p\_employee\_id || ' not found', SYSDATE);

RAISE\_APPLICATION\_ERROR(-20002, 'Employee ID not found.');

WHEN OTHERS THEN

INSERT INTO error\_logs(message, log\_date)

VALUES (SQLERRM, SYSDATE);

RAISE;

END;

END UpdateSalary;

/

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

This procedure, AddNewCustomer, will insert a new customer into the Customers table. If a customer with the same ID already exists, it will handle the exception by logging an error and preventing the insertion.

CREATE OR REPLACE PROCEDURE AddNewCustomer(

p\_customer\_id IN NUMBER,

p\_customer\_name IN VARCHAR2,

p\_customer\_email IN VARCHAR2

) IS

customer\_already\_exists EXCEPTION;

PRAGMA EXCEPTION\_INIT(customer\_already\_exists, -20003);

BEGIN

DECLARE

v\_dummy NUMBER;

BEGIN

SELECT 1 INTO v\_dummy FROM customers WHERE customer\_id = p\_customer\_id;

RAISE customer\_already\_exists;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

-- Insert new customer

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

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

COMMIT;

WHEN customer\_already\_exists THEN

INSERT INTO error\_logs(message, log\_date)

VALUES ('Customer ID ' || p\_customer\_id || ' already exists', SYSDATE);

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

WHEN OTHERS THEN

INSERT INTO error\_logs(message, log\_date)

VALUES (SQLERRM, SYSDATE);

RAISE;

END;

END AddNewCustomer;

/

**Summary**

* **Scenario 1**: SafeTransferFunds ensures that if there are insufficient funds, the transaction is rolled back, and an error message is logged.
* **Scenario 2**: UpdateSalary updates the employee's salary if the employee ID exists; otherwise, it logs an error.
* **Scenario 3**: AddNewCustomer adds a new customer if the ID does not already exist, ensuring data integrity by preventing duplicate IDs and logging errors.