**Exercise 5: Triggers**

**Scenario 1:** Automatically update the last modified date when a customer's record is updated.

* + **Question:** Write a trigger **UpdateCustomerLastModified** that updates the LastModified column of the Customers table to the current date whenever a customer's record is updated.

**CODE:**

CREATE OR REPLACE TRIGGER UpdateCustomerLastModified

BEFORE UPDATE ON CUSTOMERS

FOR EACH ROW

BEGIN

-- Update the LastModified column to the current date

:NEW.LastModified := SYSDATE;

END UpdateCustomerLastModified;

**Scenario 2:** Maintain an audit log for all transactions.

* + **Question:** Write a trigger **LogTransaction** that inserts a record into an AuditLog table whenever a transaction is inserted into the Transactions table.

**CODE:**

CREATE OR REPLACE TRIGGER LogTransaction

AFTER INSERT ON TRANSACTIONS

FOR EACH ROW

BEGIN

-- Insert a record into the AuditLog table

INSERT INTO AUDIT\_LOG (

LOG\_ID,

TRANSACTION\_ID,

LOG\_DATE,

ACTION,

DETAILS

) VALUES (

AUDIT\_LOG\_SEQ.NEXTVAL, -- Assuming there's a sequence for generating log IDs

:NEW.TRANSACTION\_ID,

SYSDATE,

'INSERT',

'Transaction of amount ' || :NEW.AMOUNT || ' for account ' || :NEW.ACCOUNT\_ID

);

END LogTransaction;

**Scenario 3:** Enforce business rules on deposits and withdrawals.

* + **Question:** Write a trigger **CheckTransactionRules** that ensures withdrawals do not exceed the balance and deposits are positive before inserting a record into the Transactions table.

**CODE:**

CREATE OR REPLACE TRIGGER CheckTransactionRules

BEFORE INSERT ON TRANSACTIONS

FOR EACH ROW

DECLARE

v\_balance ACCOUNTS.BALANCE%TYPE;

BEGIN

-- Fetch the current balance of the account

SELECT BALANCE INTO v\_balance

FROM ACCOUNTS

WHERE ACCOUNT\_ID = :NEW.ACCOUNT\_ID;

-- Check if the transaction is a withdrawal and ensure it does not exceed the balance

IF :NEW.AMOUNT < 0 THEN

IF v\_balance + :NEW.AMOUNT < 0 THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Insufficient balance for withdrawal in account ' || :NEW.ACCOUNT\_ID);

END IF;

END IF;

-- Check if the transaction is a deposit and ensure the amount is positive

IF :NEW.AMOUNT > 0 THEN

IF :NEW.AMOUNT <= 0 THEN

RAISE\_APPLICATION\_ERROR(-20002, 'Deposit amount must be positive.');

END IF;

END IF;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

-- Handle case where account does not exist

RAISE\_APPLICATION\_ERROR(-20003, 'Account ID ' || :NEW.ACCOUNT\_ID || ' does not exist.');

WHEN OTHERS THEN

-- Handle any unexpected errors

RAISE\_APPLICATION\_ERROR(-20004, 'Error checking transaction rules: ' || SQLERRM);

END CheckTransactionRules;