Scenario 1: Automatically Update the Last Modified Date

CREATE OR REPLACE TRIGGER UpdateCustomerLastModified

BEFORE UPDATE ON customers

FOR EACH ROW

BEGIN

-- Set the LastModified column to the current date

:NEW.LastModified := SYSDATE;

END UpdateCustomerLastModified;

/

Scenario 2: Maintain an Audit Log for All Transactions

CREATE OR REPLACE TRIGGER LogTransaction

AFTER INSERT ON transactions

FOR EACH ROW

BEGIN

-- Insert a record into the AuditLog table

INSERT INTO audit\_log (transaction\_id, transaction\_date, transaction\_amount, action)

VALUES (:NEW.transaction\_id, :NEW.transaction\_date, :NEW.transaction\_amount, 'INSERT');

END LogTransaction;

/

Scenario 3: Enforce Business Rules on Deposits and Withdrawals

CREATE OR REPLACE TRIGGER CheckTransactionRules

BEFORE INSERT ON transactions

FOR EACH ROW

DECLARE

v\_balance NUMBER;

BEGIN

-- Check if the transaction is a withdrawal

IF :NEW.transaction\_type = 'Withdrawal' THEN

-- Get the current balance of the account

SELECT balance INTO v\_balance

FROM accounts

WHERE account\_id = :NEW.account\_id;

-- Ensure that withdrawals do not exceed the balance

IF :NEW.transaction\_amount > v\_balance THEN

RAISE\_APPLICATION\_ERROR(-20002, 'Withdrawal amount exceeds the current balance.');

END IF;

ELSIF :NEW.transaction\_type = 'Deposit' THEN

-- Ensure that deposits are positive

IF :NEW.transaction\_amount <= 0 THEN

RAISE\_APPLICATION\_ERROR(-20003, 'Deposit amount must be positive.');

END IF;

END IF;

END CheckTransactionRules;

/