**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.

**QUERY:**

CREATE OR REPLACE TRIGGER UpdateCustomerLastModified

BEFORE UPDATE ON Customers

FOR EACH ROW

BEGIN

:NEW.LastModified := SYSDATE;

END;

**Explanation:**

BEFORE UPDATE ON Customers` specifies that the trigger should activate before an update operation on the Customers table. `FOR EACH ROW` indicates that the trigger should fire for each row that is updated. NEW.LastModified := SYSDATE; sets the LastModified column of the updated row to the current date and time using the SYSDATE function.

**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.

**QUERY:**

CREATE OR REPLACE TRIGGER LogTransaction

AFTER INSERT ON Transactions

FOR EACH ROW

BEGIN

INSERT INTO AuditLog (transaction\_id, transaction\_date, transaction\_type)

VALUES (:NEW.transaction\_id, SYSDATE, :NEW.transaction\_type);

END;

**Explanation:**

`AFTER INSERT ON Transactions` specifies that the trigger should activate after an insert operation on the Transactions table. `FOR EACH ROW` indicates that the trigger should fire for each row that is inserted. `INSERT INTO AuditLog` inserts a new record into the AuditLog table with the transaction\_id, transaction\_date (set to the current date and time using SYSDATE), and transaction\_type from the newly inserted row in the Transactions table.

**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.

**QUERY:**

CREATE OR REPLACE TRIGGER CheckTransactionRules

BEFORE INSERT ON Transactions

FOR EACH ROW

BEGIN

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

IF :NEW.amount > :NEW.balance THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Withdrawal amount exceeds available balance');

END IF;

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

IF :NEW.amount <= 0 THEN

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

END IF;

END IF;

END;

**Explanation:**

BEFORE INSERT ON Transactions` specifies that the trigger should activate before an insert operation on the Transactions table. `FOR EACH ROW` indicates that the trigger should fire for each row that is inserted. The trigger checks if the transaction type is a withdrawal or deposit and then enforces the corresponding business rule: For a withdrawal, it checks if the withdrawal amount exceeds the available balance. If it does, it raises an error. For a deposit, it checks if the deposit amount is positive. If it's not, it raises an error.