**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 trg\_update\_balance

AFTER INSERT ON Transactions

FOR EACH ROW

DECLARE

  v\_current\_balance NUMBER;

BEGIN

  SELECT Balance INTO v\_current\_balance

  FROM Accounts

  WHERE AccountID = :NEW.AccountID;

  IF UPPER(:NEW.TransactionType) = 'CREDIT' THEN

    v\_current\_balance := v\_current\_balance + :NEW.Amount;

  ELSIF UPPER(:NEW.TransactionType) = 'DEBIT' THEN

    v\_current\_balance := v\_current\_balance - :NEW.Amount;

  END IF;

  UPDATE Accounts

  SET Balance = v\_current\_balance,

      LastModified = SYSDATE

  WHERE AccountID = :NEW.AccountID;

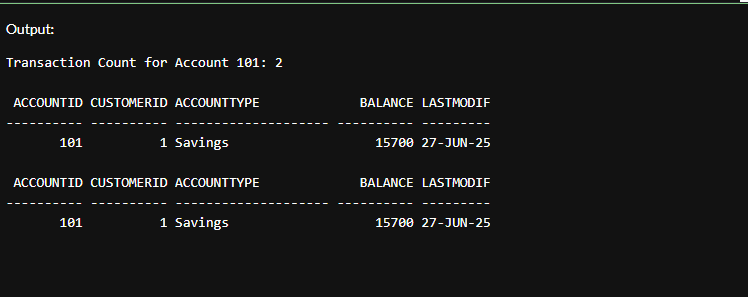
END;

/

INSERT INTO Transactions VALUES (10, 101, SYSDATE, 500, 'CREDIT');

INSERT INTO Transactions VALUES (11, 101, SYSDATE, 200, 'CREDIT');

SELECT \* FROM Accounts WHERE AccountID = 101;



**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 TABLE Customers\_Archive (

  CustomerID NUMBER,

  Name VARCHAR2(100),

  DOB DATE,

  Balance NUMBER,

  LastModified DATE,

  DeletedOn DATE

);

CREATE OR REPLACE TRIGGER trg\_archive\_customer

BEFORE DELETE ON Customers

FOR EACH ROW

BEGIN

  INSERT INTO Customers\_Archive (

    CustomerID, Name, DOB, Balance, LastModified, DeletedOn

  ) VALUES (

    :OLD.CustomerID, :OLD.Name, :OLD.DOB, :OLD.Balance, :OLD.LastModified, SYSDATE

  );

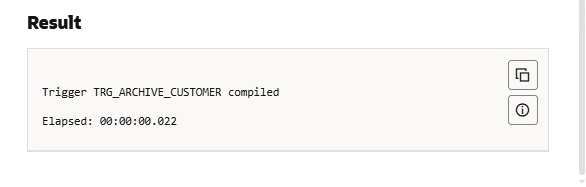
END;

/

DELETE FROM Customers WHERE CustomerID = 2;

COMMIT;

SELECT \* FROM Customers\_Archive;



**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:**

SHOW ERRORS TRIGGER trg\_prevent\_negative\_balance;

CREATE OR REPLACE TRIGGER trg\_prevent\_negative\_balance

BEFORE INSERT ON Transactions

FOR EACH ROW

DECLARE

  v\_balance NUMBER;

BEGIN

  IF UPPER(:NEW.TransactionType) = 'DEBIT' THEN

    SELECT Balance INTO v\_balance

    FROM Accounts

    WHERE AccountID = :NEW.AccountID;

    IF v\_balance < :NEW.Amount THEN

      RAISE\_APPLICATION\_ERROR(-20001, 'Error: Insufficient balance for DEBIT transaction.');

    END IF;

  END IF;

END;

/

SELECT table\_name FROM user\_tables WHERE table\_name = 'ACCOUNTS';

INSERT INTO Transactions VALUES (30, 101, SYSDATE, 500, 'DEBIT');

INSERT INTO Transactions VALUES (31, 101, SYSDATE, 999999, 'DEBIT');

**Output:**

