Database and

Management System

Lab

Lab Experiment – 16

Name: Shubhojit Mitra

Roll No: R2142230793

Sap_ID: 500120225

B.Tech CSE, SEM-III, B-1

Title: To understand the concepts of Trigger.

Objective: Students will be able to implement the concept of trigger.

1. Create a row level trigger for the customers table

```
CREATE OR REPLACE TRIGGER customer_audit_trigger

BEFORE INSERT OR UPDATE OR DELETE ON CUSTOMERS

FOR EACH ROW

BEGIN

-- For INSERTING THEN

DBMS_OUTPUT.PUT_LINE('INSERT Operation');

BEMS_OUTPUT.PUT_LINE('New Salary: ' || :NEW.SALARY);

-- FOR UPDATE operations

ELSIF UPDATING THEN

DBMS_OUTPUT.PUT_LINE('UPDATE Operation');

DBMS_OUTPUT.PUT_LINE('Old Salary: ' || :OLD.SALARY);

DBMS_OUTPUT.PUT_LINE('New Salary: ' || :NEW.SALARY);

DBMS_OUTPUT.PUT_LINE('Salary Difference: ' || (:NEW.SALARY - :OLD.SALARY));

-- FOR DELETE operations

ELSIF DELETING THEN

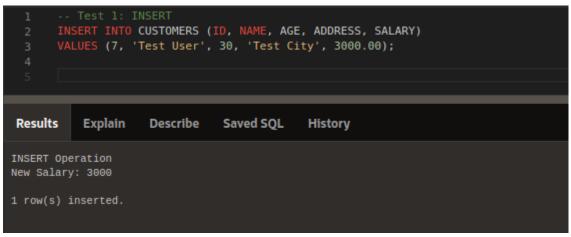
DBMS_OUTPUT.PUT_LINE('DELETE Operation');

DBMS_OUTPUT.PUT_LINE('Deleted Salary: ' || :OLD.SALARY);

END IF;

END;

/ Selection of the customer and the custome
```



```
-- Test 2: UPDATE
        UPDATE CUSTOMERS
        SET SALARY = 2500.00
        WHERE ID = 1;
Results
           Explain
                     Describe
                                Saved SQL
                                             History
UPDATE Operation
Old Salary: 2000
New Salary: 2500
Salary Difference: 500
1 row(s) updated.
        -- Test 3: DELETE
        DELETE FROM CUSTOMERS
        WHERE ID = 7;
Results
           Explain
                    Describe
                                Saved SQL
                                             History
DELETE Operation
Deleted Salary: 3000
1 row(s) deleted.
```

Database and

Management System

Lab

Lab Experiment – 17

Name: Shubhojit Mitra Roll No: R2142230793 Sap_ID: 500120225

B.Tech CSE, SEM-III, B-1

Title: To understand the concepts of Trigger.

Objective: Students will be able to implement the concept of trigger.

```
CREATE OR REPLACE TRIGGER SALARY_VIOLATION
BEFORE INSERT OR UPDATE OF SALARY, SUPERSSN ON EMPLOYEE

FOR EACH ROW

DECLARE

v_super_salary NUMBER;
v_dept_mgr_ssn VARCHAR2(9);

BEGIN

-- Only proceed with checks if supervisor exists

IF :NEW.SUPERSSN IS NOT NULL THEN

-- Get supervisor's salary

SELECT SALARY INTO v_super_salary

FROM EMPLOYEE

WHERE SSN = :NEW.SUPERSSN;

-- Check if employee's salary is greater than supervisor's salary

IF :NEW.SALARY > v_super_salary THEN

RAISE_APPLICATION_ERROR(-20001,

Employee salary ('|| v_super_salary ||') cannot be greater than supervisor salary ('|| v_super_salary ||')');

END IF;

END IF;
```

```
-- Get department manager's SSN for the employee's department

IF :NEW.DNO IS NOT NULL THEN

SELECT MGRSSN INTO v_dept_mgr_ssn
FROM DEPARTMENT

WHERE DNUMBER = :NEW.DNO;

-- If the employee is the department manager, salary must be at least 50000

IF :NEW.SSN = v_dept_mgr_ssn AND :NEW.SALARY < 50000 THEN

RAISE_APPLICATION_ERROR(-20002,

| 'Department manager salary must be at least 50000');

END IF;

-- Validate basic salary rules

IF :NEW.SALARY < 20000 THEN

RAISE_APPLICATION_ERROR(-20003,

| 'Salary cannot be less than 20000');

END IF;
```

```
-- Validate supervisor changes

IF UPDATING('SUPERSSN') AND :NEW.DNO IS NOT NULL THEN

-- Prevent department managers from having supervisors

IF :NEW.SSN = v_dept_mgr_ssn_AND :NEW.SUPERSSN IS NOT NULL THEN

RAISE_APPLICATION_ERROR(-20004,

| Department managers cannot have supervisors');

END IF;

END IF;

END IF;

END IF;

SO /
```

