USMAN INSTITUTE OF TECHNOLOGY

Department of Computer Science

CS311 Introduction to Database Systems

Lab#11

Objective:

- DATABASE TRIGGERS (I)

Name of Student: Muhammad Waleed

Roll No: 20B-115-SE Sec. B

Date of Experiment:

..

Marks Obtained/Remarks: ______________________________

Signature: ______________________________

What is a trigger?

A *Trigger* is a PL/SQL block that executes implicitly whenever a particular event takes place. A trigger can be either a database trigger or an application trigger.

Database triggers execute implicitly when an INSERT, UPDATE, or DELETE statement is issued against the associated table, no matter which user is connected or which application is used.

Application triggers execute implicitly whenever a particular event occurs within an application. An example of an application that uses triggers extensively is one developed with Developer/2000 Form Builder.

Note: Database triggers can be defined only on tables, not on views. However, if a DML operation is issued against a view, triggers on the base table(s) of a view are fired.

Guidelines for designing triggers

- Only use database triggers for centralized, global operations that should be fired for the triggering statement, regardless of which user or application issues the statement.
- Do not define triggers to implement integrity rules that can be done by using declarative constraints.
- The excessive use of triggers can result in complex interdependencies, which may be difficult to maintain in large applications. Only use triggers when necessary, and beware of recursive and cascading effects.

Database Trigger Types

The trigger type determines the number of times the trigger action is to be executed: once for every row affected by the triggering statement (such as a multiple row UPDATE), or once for the triggering statement no matter how many rows it affects.

Statement Trigger

A statement trigger is fired once on behalf of the triggering event, even if no rows are affected at all. Statement triggers are useful if the trigger action does not depend on data of rows that are affected or data provided by the triggering event itself. For example, a trigger that performs a complex security check on the current user.

Row Trigger

A Row trigger fires each time the table is affected by the triggering event. If the triggering event affects no row(s), a row trigger is not executed at all.
Row triggers are useful if the trigger action depends on data of rows that are affected or data provided by the triggering event itself.

Creating Statement Triggers

Syntax for creating Statement Triggers

CREATE [OR REPLACE] TRIGGER *trigger_name*
Timing event1 [OR *event2* OR *event3*] ON *table_name*
PL/SQL block;

Trigger Components

Before coding the trigger block, decide on the components of the trigger:- Trigger timing:

BEFORE or AFTER
Triggering event: INSERT or UPDATE or DELETE
Table Name: ON table
Trigger Type: Row or Statement

```
Trigger body:  DECLARE
               BEGIN
               END;
```

Trigger Timing

Indicates the time when the trigger fires in relation to the triggering event: BEFORE or AFTER.
<u>BEFORE Triggers</u>
This type of trigger is frequently used in the following situations:

- When the trigger action should determine whether that triggering statement should be allowed to complete. This allows to eliminate unnecessary processing of the triggering statement and its eventual rollback in cases where an exception is raised in the triggering action.
- To derive column values before completing a triggering INSERT or UPDATE statement.

<u>AFTER Triggers</u>
This type of trigger is frequently used in the following situations:

- When the triggering statement is to be completed before executing the triggering action.
- If a BEFORE trigger is already present, and an after trigger can perform different actions on the same triggering statement.

Triggering Event

The triggering event or statement can be an INSERT, UPDATE, or DELETE statement on a table.

- When the triggering event is an UPDATE, we can include a column list to identify which column(s) must be changed to fire the trigger. We cannot specify a column list for an INSERT or for a DELETE statement, as they always affect entire rows.
- The triggering event can contain multiple DML statements. In this way, we can differentiate what code to execute depending on the statement that caused the triggers to fire.

Trigger Body
The trigger action defines what needs to be done when the triggering event is issued. It can contain SQL and PL/SQL statements, define PL/SQL constructs such as variables, cursors, exceptions and so on. Additionally row triggers have access to the old and new column values of the row being processed by the trigger, using correlation names. The trigger body is defined with an anonymous PL/SQL block.

```
[DECLARE]
BEGIN
[EXCEPTION]
END;
```

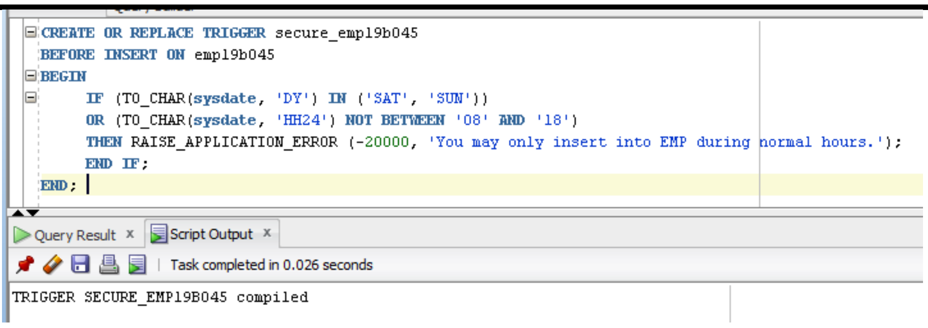
Before Statement Trigger:
We can create a *BEFORE statement* trigger in order to prevent the triggering operation from succeeding if a certain condition is violated.

For example, create a trigger to restrict inserts into the EMP table to certain business hours on Monday through Friday. If a user attempted to insert a row into the EMP table on Saturday, for example, the user will see the message, the trigger will fail, and the triggering statement will be rolled back.

RAISE_APPLICATION_ERROR is a server-side built-in procedure that prints a message to the user and causes the PL/SQL block to fail. When a database trigger fails, the triggering statement is automatically rolled back by the Oracle Server.

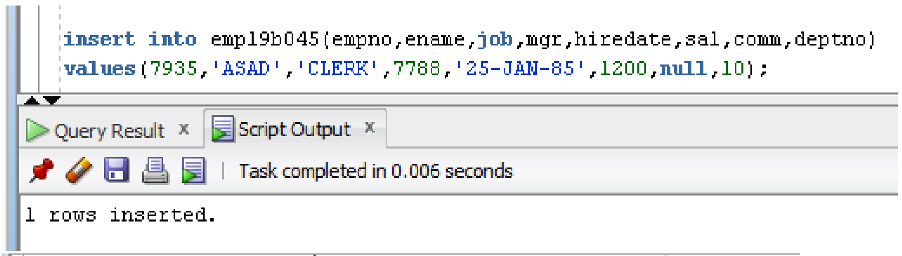
Creating Before Insert Trigger:

```
CREATE OR REPLACE TRIGGER secure_emp
BEFORE INSERT ON emp
BEGIN
    IF (TO_CHAR(sysdate, 'DY') IN ('SAT', 'SUN'))
    OR (TO_CHAR(sysdate, 'HH24') NOT BETWEEN '08' AND '18')      THEN
RAISE_APPLICATION_ERROR (-20000, 'You may only insert into EMP during
normal hours.');
    END IF;
END;
```



```
CREATE OR REPLACE TRIGGER secure_emp19b045
BEFORE INSERT ON emp19b045
BEGIN
    IF (TO_CHAR(sysdate, 'DY') IN ('SAT', 'SUN'))
    OR (TO_CHAR(sysdate, 'HH24') NOT BETWEEN '08' AND '18')
    THEN RAISE_APPLICATION_ERROR (-20000, 'You may only insert into EMP during normal hours.');
    END IF;
END;
```

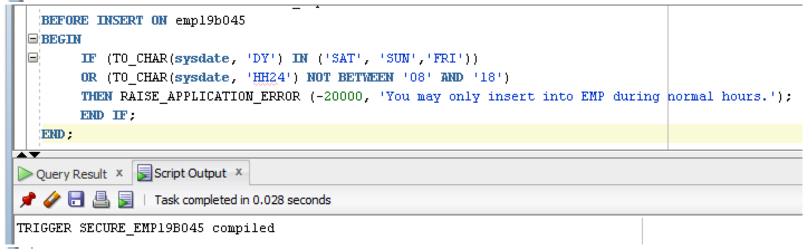
```
TRIGGER SECURE_EMP19B045 compiled
```



```
insert into empl9b045(empno,ename,job,mgr,hiredate,sal,comm,deptno)
values(7935,'ASAD','CLERK',7788,'25-JAN-85',1200,null,10);
```

Task completed in 0.006 seconds

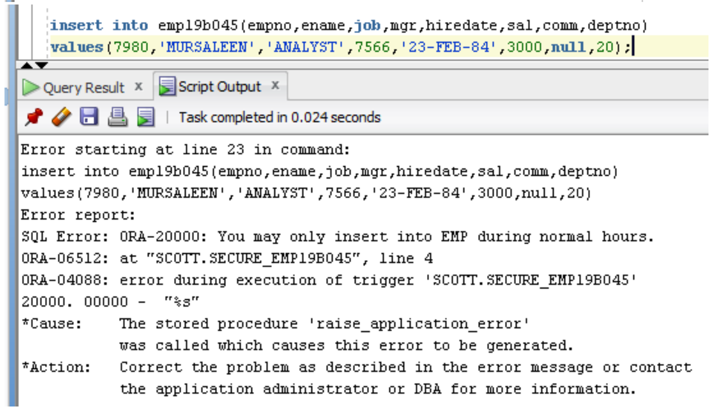
```
1 rows inserted.
```



```
BEFORE INSERT ON empl9b045
BEGIN
    IF (TO_CHAR(sysdate, 'DY') IN ('SAT', 'SUN','FRI'))
    OR (TO_CHAR(sysdate, 'HH24') NOT BETWEEN '08' AND '18')
    THEN RAISE_APPLICATION_ERROR (-20000, 'You may only insert into EMP during normal hours.');
    END IF;
END;
```

Task completed in 0.028 seconds

```
TRIGGER SECURE_EMP19B045 compiled
```



```
insert into empl9b045(empno,ename,job,mgr,hiredate,sal,comm,deptno)
values(7980,'MURSALEEN','ANALYST',7566,'23-FEB-84',3000,null,20);
```

Task completed in 0.024 seconds

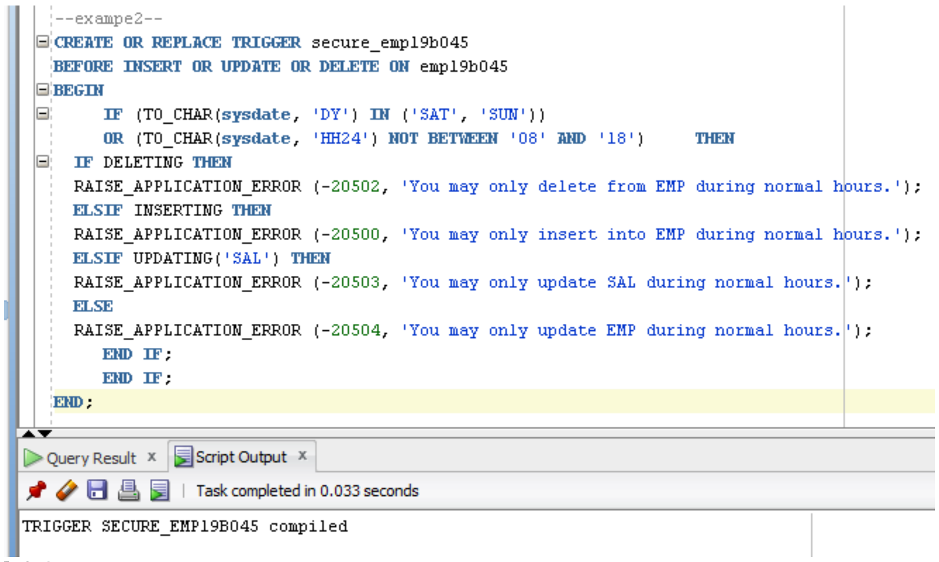
```
Error starting at line 23 in command:
insert into empl9b045(empno,ename,job,mgr,hiredate,sal,comm,deptno)
values(7980,'MURSALEEN','ANALYST',7566,'23-FEB-84',3000,null,20)
Error report:
SQL Error: ORA-20000: You may only insert into EMP during normal hours.
ORA-06512: at "SCOTT.SECURE_EMP19B045", line 4
ORA-04088: error during execution of trigger 'SCOTT.SECURE_EMP19B045'
20000. 00000 -  "%s"
*Cause:    The stored procedure 'raise_application_error'
           was called which causes this error to be generated.
*Action:   Correct the problem as described in the error message or contact
           the application administrator or DBA for more information.
```

Using Conditional Predicates

We can combine several triggering events into one by taking advantage of the special conditional predicates INSERTING, UPDATING, and DELETING within the trigger body. For example, create one trigger to restrict all data manipulation events on the EMP table to certain business hours, Monday through Friday. Also use BEFORE statement triggers to initialize global variables or flags, and to validate complex business rules.

```
CREATE OR REPLACE TRIGGER secure_emp
BEFORE INSERT OR UPDATE OR DELETE ON emp
BEGIN
    IF (TO_CHAR(sysdate, 'DY') IN ('SAT', 'SUN'))
    OR (TO_CHAR(sysdate, 'HH24') NOT BETWEEN '08' AND '18')     THEN
         IF DELETING THEN
          RAISE_APPLICATION_ERROR (-20502, 'You may only delete from EMP      during
normal hours.');
         ELSIF INSERTING THEN
          RAISE_APPLICATION_ERROR (-20500, 'You may only insert into EMP      during
normal hours.');
         ELSIF UPDATING('SAL') THEN
          RAISE_APPLICATION_ERROR (-20503, 'You may only update SAL      during normal
hours.');
         ELSE
```

```
          RAISE_APPLICATION_ERROR (-20504, 'You may only update EMP      during
normal hours.');
    END IF;
    END IF;
END;
```

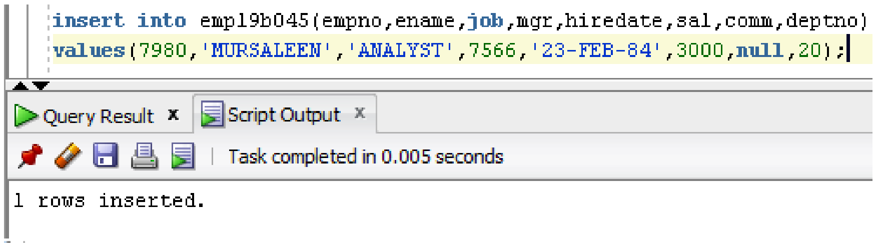


```
--exampe2--
CREATE OR REPLACE TRIGGER secure_emp19b045
BEFORE INSERT OR UPDATE OR DELETE ON emp19b045
BEGIN
      IF (TO_CHAR(sysdate, 'DY') IN ('SAT', 'SUN'))
      OR (TO_CHAR(sysdate, 'HH24') NOT BETWEEN '08' AND '18')      THEN
  IF DELETING THEN
  RAISE_APPLICATION_ERROR (-20502, 'You may only delete from EMP during normal hours.');
  ELSIF INSERTING THEN
  RAISE_APPLICATION_ERROR (-20500, 'You may only insert into EMP during normal hours.');
  ELSIF UPDATING('SAL') THEN
  RAISE_APPLICATION_ERROR (-20503, 'You may only update SAL during normal hours.');
  ELSE
  RAISE_APPLICATION_ERROR (-20504, 'You may only update EMP during normal hours.');
      END IF;
      END IF;
END;
```

Query Result × Script Output ×

Task completed in 0.033 seconds

```
TRIGGER SECURE_EMP19B045 compiled
```

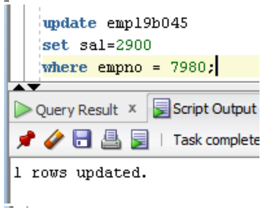


```
insert into emp19b045(empno,ename,job,mgr,hiredate,sal,comm,deptno)
values(7980,'MURSALEEN','ANALYST',7566,'23-FEB-84',3000,null,20);
```

Query Result × Script Output ×

Task completed in 0.005 seconds

```
1 rows inserted.
```

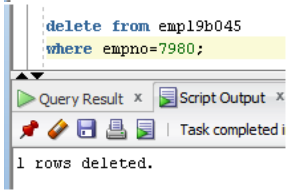


```
update emp19b045
set sal=2900
where empno = 7980;
```

Query Result × Script Output

Task complete

```
1 rows updated.
```



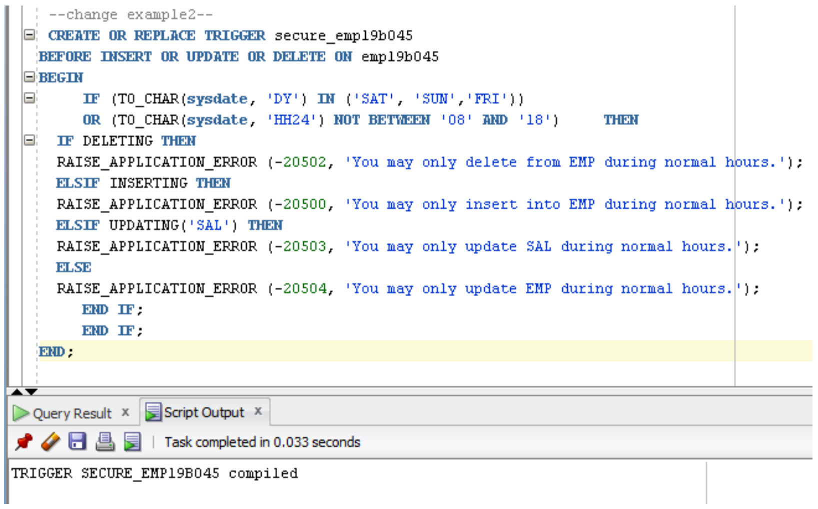
```
delete from emp19b045
where empno=7980;
```

Query Result × Script Output ×

Task completed i

```
1 rows deleted.
```

CHANGE

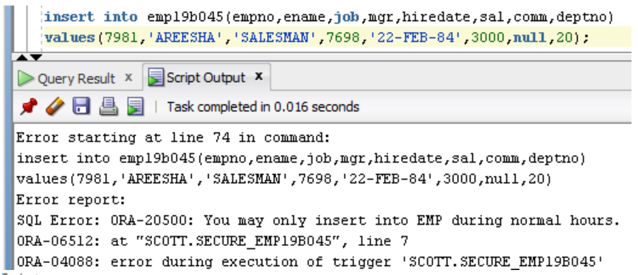


```
--change example2--
CREATE OR REPLACE TRIGGER secure_emp19b045
BEFORE INSERT OR UPDATE OR DELETE ON emp19b045
BEGIN
      IF (TO_CHAR(sysdate, 'DY') IN ('SAT', 'SUN','FRI'))
      OR (TO_CHAR(sysdate, 'HH24') NOT BETWEEN '08' AND '18')      THEN
  IF DELETING THEN
  RAISE_APPLICATION_ERROR (-20502, 'You may only delete from EMP during normal hours.');
  ELSIF INSERTING THEN
  RAISE_APPLICATION_ERROR (-20500, 'You may only insert into EMP during normal hours.');
  ELSIF UPDATING('SAL') THEN
  RAISE_APPLICATION_ERROR (-20503, 'You may only update SAL during normal hours.');
  ELSE
  RAISE_APPLICATION_ERROR (-20504, 'You may only update EMP during normal hours.');
      END IF;
      END IF;
END;
```

Query Result × Script Output ×

Task completed in 0.033 seconds

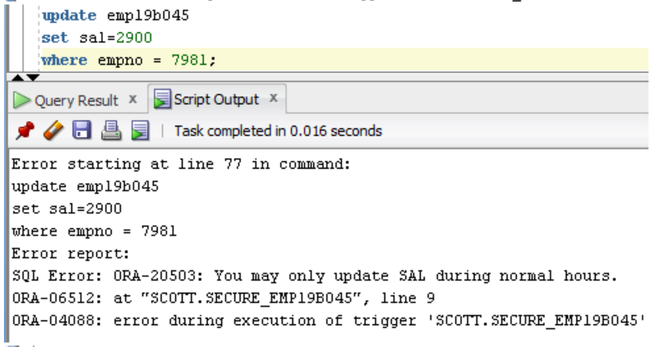
```
TRIGGER SECURE_EMP19B045 compiled
```



```
insert into emp19b045(empno,ename,job,mgr,hiredate,sal,comm,deptno)
values(7981,'AREESHA','SALESMAN',7698,'22-FEB-84',3000,null,20);
```

Query Result × Script Output × | Task completed in 0.016 seconds

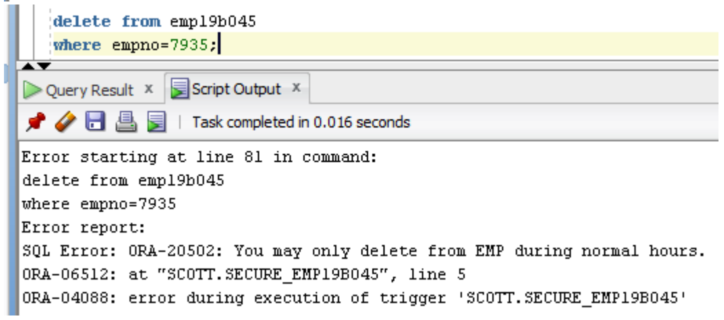
```
Error starting at line 74 in command:
insert into emp19b045(empno,ename,job,mgr,hiredate,sal,comm,deptno)
values(7981,'AREESHA','SALESMAN',7698,'22-FEB-84',3000,null,20)
Error report:
SQL Error: ORA-20500: You may only insert into EMP during normal hours.
ORA-06512: at "SCOTT.SECURE_EMP19B045", line 7
ORA-04088: error during execution of trigger 'SCOTT.SECURE_EMP19B045'
```



```
update emp19b045
set sal=2900
where empno = 7981;
```

Query Result × Script Output × | Task completed in 0.016 seconds

```
Error starting at line 77 in command:
update emp19b045
set sal=2900
where empno = 7981
Error report:
SQL Error: ORA-20503: You may only update SAL during normal hours.
ORA-06512: at "SCOTT.SECURE_EMP19B045", line 9
ORA-04088: error during execution of trigger 'SCOTT.SECURE_EMP19B045'
```



```
delete from emp19b045
where empno=7935;
```

Query Result × Script Output × | Task completed in 0.016 seconds

```
Error starting at line 81 in command:
delete from emp19b045
where empno=7935
Error report:
SQL Error: ORA-20502: You may only delete from EMP during normal hours.
ORA-06512: at "SCOTT.SECURE_EMP19B045", line 5
ORA-04088: error during execution of trigger 'SCOTT.SECURE_EMP19B045'
```

After Statement Trigger:

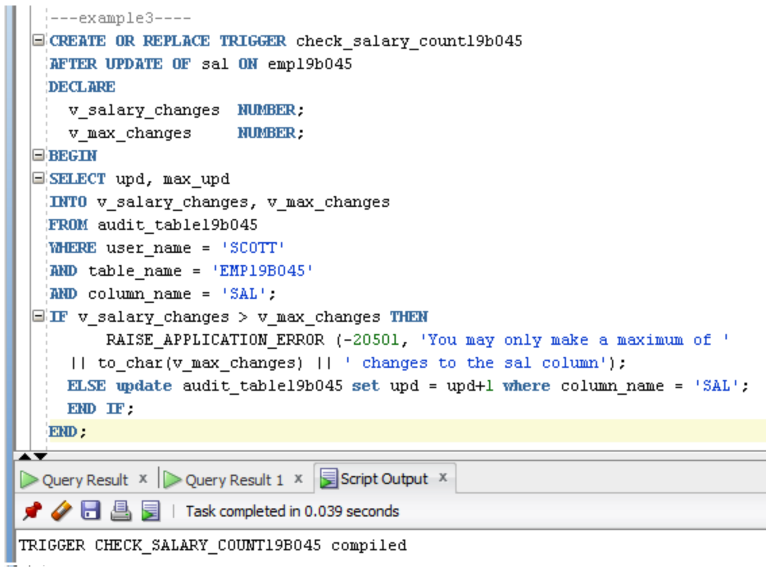
We can create an *AFTER Statement* trigger in order to audit the triggering operation or perform a calculation after an operation has completed.
Suppose we have a user defined audit table that lists users and counts their data manipulation operations. After any user has updated the SAL column in the EMP table, use the audit table to ensure that the number of salary changes does not exceed the maximum permitted for that user.

User Audit Table

USER_ NAME	TABLE_NAME	COLUMN_NAME	INS	UPD	DEL	MAX_INS	MAX_UPD	MAX_DEL
SCOTT	EMP		1	1	1	5	5	5
SCOTT	EMP	SAL		1			5	
SCOTT	EMP		0	0	0	5	0	0

After Update Trigger:

```
CREATE OR REPLACE TRIGGER check_salary_count
AFTER UPDATE OF sal ON emp
DECLARE
  v_salary_changes  NUMBER;
  v_max_changes     NUMBER;
BEGIN
SELECT upd, max_upd
INTO v_salary_changes, v_max_changes
FROM audit_table
WHERE user_name = user
AND tablename = 'EMP'
AND column_name = 'SAL';
IF v_salary_changes > v_max_changes THEN
        RAISE_APPLICATION_ERROR (-20501, 'You may only make a maximum of '
         || to_char(v_max_changes) || ' changes to the sal column'); END IF; END;
```

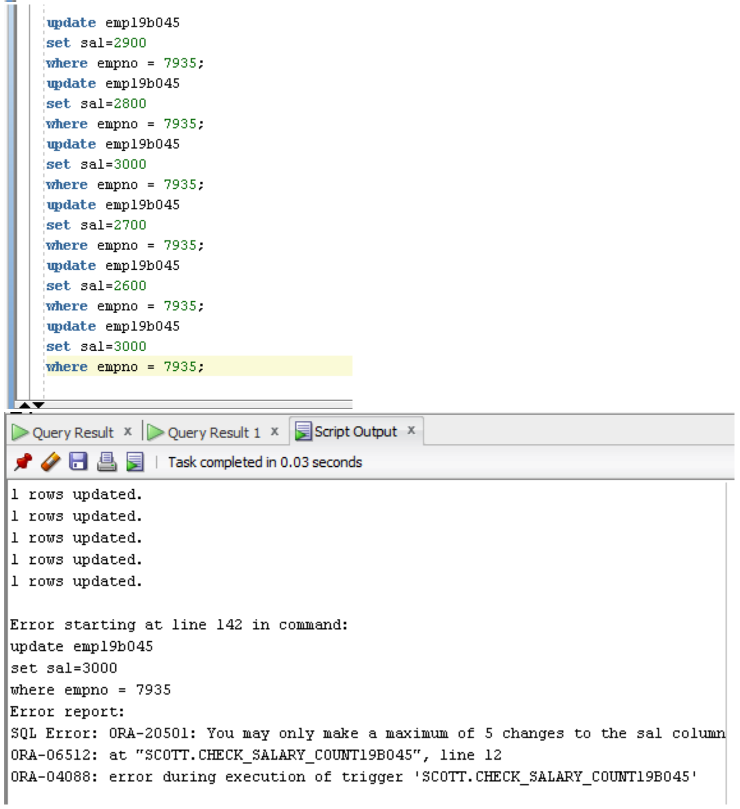


```
---example3----
CREATE OR REPLACE TRIGGER check_salary_count19b045
AFTER UPDATE OF sal ON emp19b045
DECLARE
  v_salary_changes  NUMBER;
  v_max_changes     NUMBER;
BEGIN
SELECT upd, max_upd
INTO v_salary_changes, v_max_changes
FROM audit_table19b045
WHERE user_name = 'SCOTT'
AND table_name = 'EMP19B045'
AND column_name = 'SAL';
IF v_salary_changes > v_max_changes THEN
      RAISE_APPLICATION_ERROR (-20501, 'You may only make a maximum of '
  || to_char(v_max_changes) || ' changes to the sal column');
  ELSE update audit_table19b045 set upd = upd+1 where column_name = 'SAL';
  END IF;
END;
```

Query Result × Query Result 1 × Script Output ×

Task completed in 0.039 seconds

```
TRIGGER CHECK_SALARY_COUNT19B045 compiled
```



```
update emp19b045
set sal=2900
where empno = 7935;
update emp19b045
set sal=2800
where empno = 7935;
update emp19b045
set sal=3000
where empno = 7935;
update emp19b045
set sal=2700
where empno = 7935;
update emp19b045
set sal=2600
where empno = 7935;
update emp19b045
set sal=3000
where empno = 7935;
```

Query Result × Query Result 1 × Script Output ×

Task completed in 0.03 seconds

```
1 rows updated.
1 rows updated.
1 rows updated.
1 rows updated.
1 rows updated.

Error starting at line 142 in command:
update emp19b045
set sal=3000
where empno = 7935
Error report:
SQL Error: ORA-20501: You may only make a maximum of 5 changes to the sal column
ORA-06512: at "SCOTT.CHECK_SALARY_COUNT19B045", line 12
ORA-04088: error during execution of trigger 'SCOTT.CHECK_SALARY_COUNT19B045'
```

After Delete Trigger: Example

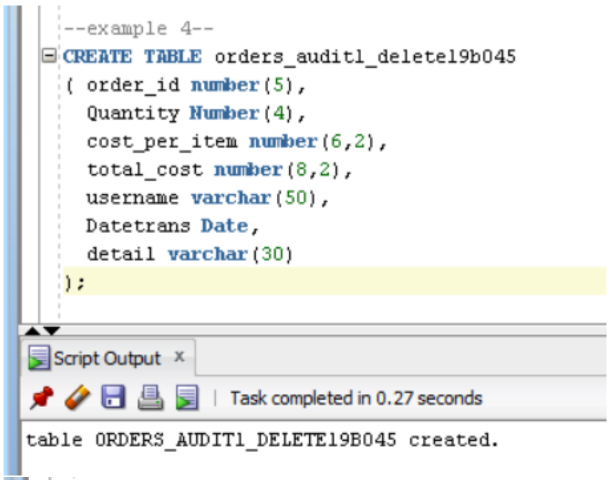
Creating an Audit Table:

CREATE TABLE orders_audit1_delete
(order_id number(5),
  quantity number(4),
  cost_per_item number(6,2),
  total_cost number(8,2),

```
  username varchar(50),
  datetrans date,
  detail varchar(30)
);
```

<u>Creating After Delete Trigger:</u>

```
CREATE OR REPLACE TRIGGER orders_after_delete1
AFTER delete
   ON orders
   FOR EACH ROW

DECLARE
   v_username varchar2(10);
details varchar(30);

BEGIN
details:='record has been deleted'|| TO_CHAR(:old.order_id);

   -- Find username of person performing the INSERT into the table
   SELECT user INTO v_username
   FROM dual;
   -- Insert record into audit table
   INSERT INTO orders_audit1_delete
   ( order_id,
     quantity,
     cost_per_item,
     total_cost,
     username,datetrans,detail )
   VALUES
   ( :old.order_id,
     :old.quantity,
     :old.cost_per_item,
     :old.total_cost,
     v_username,sysdate,details );
END;
```

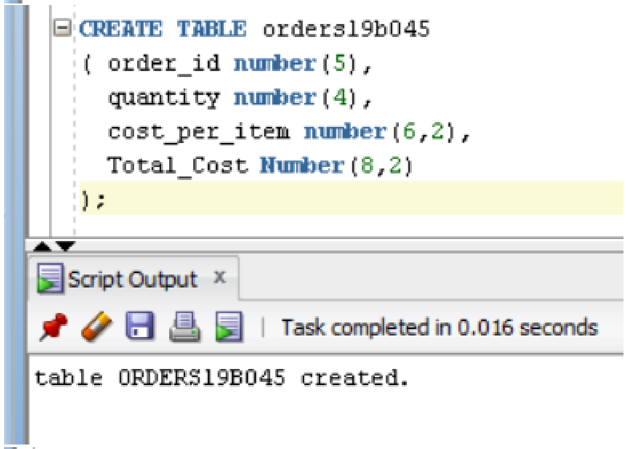


```
--example 4--
CREATE TABLE orders_audit1_delete19b045
( order_id number(5),
  Quantity Number(4),
  cost_per_item number(6,2),
  total_cost number(8,2),
  username varchar(50),
  Datetrans Date,
  detail varchar(30)
);
```

Script Output

Task completed in 0.27 seconds

```
table ORDERS_AUDIT1_DELETE19B045 created.
```

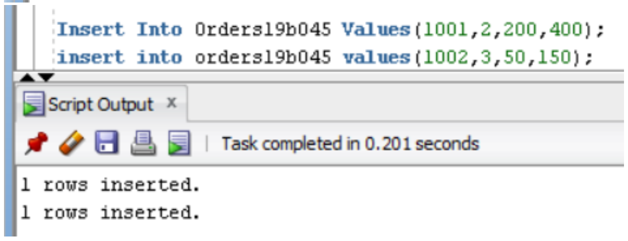


```
CREATE TABLE orders19b045
( order_id number(5),
  quantity number(4),
  cost_per_item number(6,2),
  Total_Cost Number(8,2)
);
```

Script Output

Task completed in 0.016 seconds

```
table ORDERS19B045 created.
```

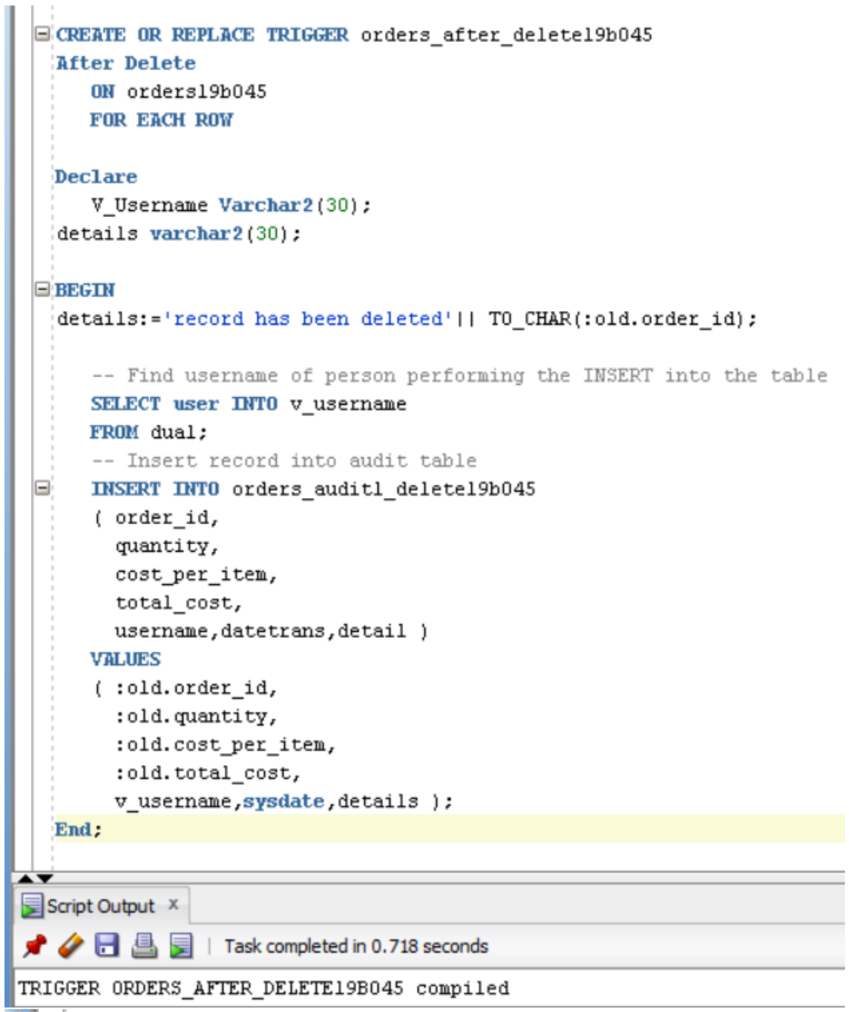


```
Insert Into Orders19b045 Values(1001,2,200,400);
insert into orders19b045 values(1002,3,50,150);
```

Script Output

Task completed in 0.201 seconds

```
1 rows inserted.
1 rows inserted.
```

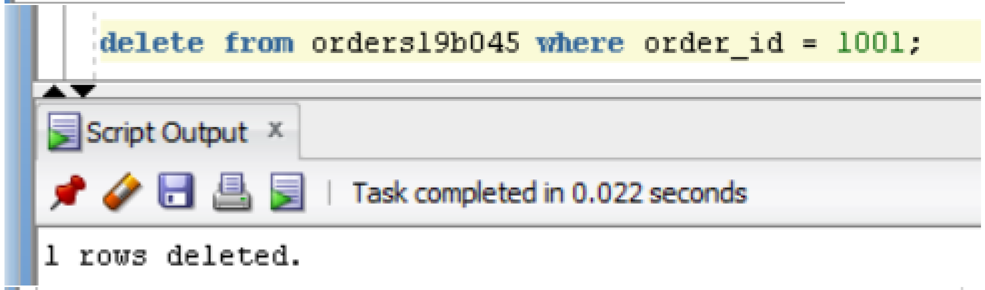


```
CREATE OR REPLACE TRIGGER orders_after_delete19b045
After Delete
   ON orders19b045
   FOR EACH ROW

Declare
   V_Username Varchar2(30);
details varchar2(30);

BEGIN
details:='record has been deleted'|| TO_CHAR(:old.order_id);

   -- Find username of person performing the INSERT into the table
   SELECT user INTO v_username
   FROM dual;
   -- Insert record into audit table
   INSERT INTO orders_auditl_delete19b045
   ( order_id,
     quantity,
     cost_per_item,
     total_cost,
     username,datetrans,detail )
   VALUES
   ( :old.order_id,
     :old.quantity,
     :old.cost_per_item,
     :old.total_cost,
     v_username,sysdate,details );
End;
```

Script Output

Task completed in 0.718 seconds

```
TRIGGER ORDERS_AFTER_DELETE19B045 compiled
```

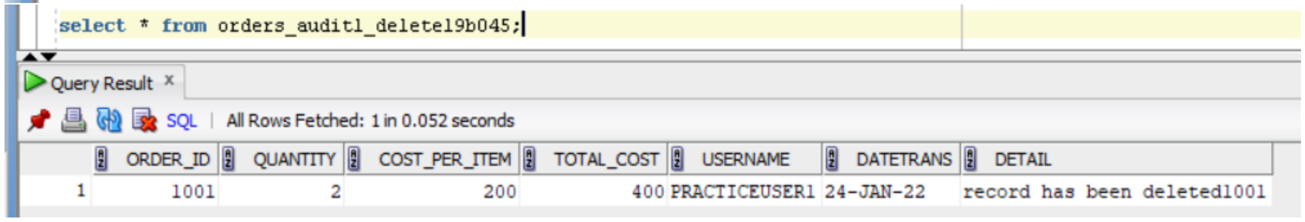


```
delete from orders19b045 where order_id = 1001;
```

Script Output

Task completed in 0.022 seconds

```
1 rows deleted.
```



```
select * from orders_auditl_delete19b045;
```

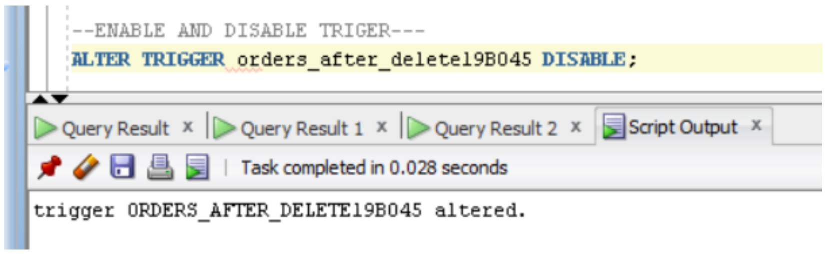
Query Result

All Rows Fetched: 1 in 0.052 seconds

	ORDER_ID	QUANTITY	COST_PER_ITEM	TOTAL_COST	USERNAME	DATETRANS	DETAIL
1	1001	2	200	400	PRACTICEUSER1	24-JAN-22	record has been deleted1001

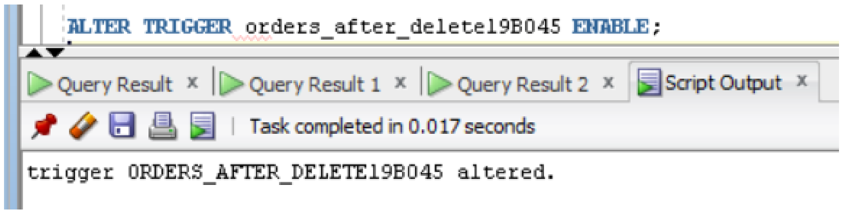
Disable/Enable a database trigger

ALTER TRIGGER *trigger_name DISABLE* | *ENABLE*;



```
--ENABLE AND DISABLE TRIGER---
ALTER TRIGGER orders_after_delete19B045 DISABLE;
```

```
Task completed in 0.028 seconds
trigger ORDERS_AFTER_DELETE19B045 altered.
```

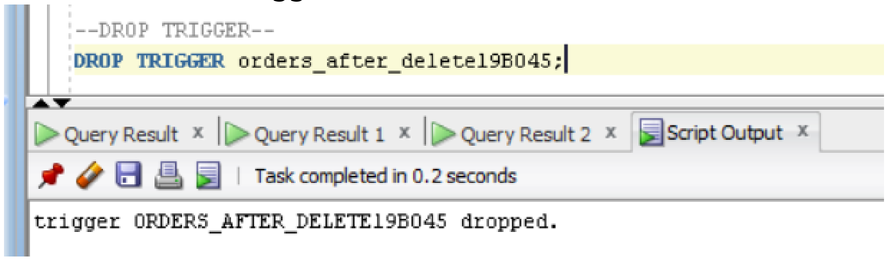


```
ALTER TRIGGER orders_after_delete19B045 ENABLE;
```

```
Task completed in 0.017 seconds
trigger ORDERS_AFTER_DELETE19B045 altered.
```

Removing a Trigger

DROP TIGGER *trigger_name*;



```
--DROP TRIGGER--
DROP TRIGGER orders_after_delete19B045;
```

```
Task completed in 0.2 seconds
trigger ORDERS_AFTER_DELETE19B045 dropped.
```

EXERCISE

1. What are triggers? Differentiate between database triggers and row triggers.

A **trigger** is a PL/SQL block that executes implicitly whenever a particular event takes place. A trigger can be either a database trigger or an application trigger.

Database triggers execute implicitly when an INSERT, UPDATE, or DELETE statement is issued against the associated table, no matter which user is connected, or which application is used. Triggers can be broadly classified into Row Level and Statement Level triggers.
A **Row trigger** fires each time the table is affected by the triggering event. If the triggering event affects no row(s), a row trigger is not executed at all.
Row triggers are useful if the trigger action depends on data of rows that are affected, or data provided by the triggering event itself.

2. .Differentiate between Statement and Row triggers?

Row Triggers:

Row level triggers executes once for each and every row in the transaction. It is used for data auditing purpose.

Statement Trigger:

Statement level triggers executes only once for each single transaction. It is used for enforcing all additional security on the transactions performed on the table.

3. What is meant by triggering event? Give examples.

A triggering event or statement is the SQL statement, database event, or user event that causes a trigger to fire. A triggering event can be one or more of the following:

- An INSERT, UPDATE, or DELETE statement on a specific table.
- A CREATE, ALTER, or DROP statement on any schema object
- A database startup or instance shutdown
- A specific error message or any error message
- A user logon or logoff

---

4. Give three examples of a situation when a BEFORE statement trigger is needed?

- create a trigger to restrict on EMPLOYEES to insert data in certain business hours on Monday through Friday
- We will create a BEFORE INSERT trigger to maintain a summary table from another table.
- create a trigger to restrict on Staff to update data in certain business hours.

---

5. Give three examples of a situation when a AFTER statement trigger is needed?

- Suppose we have a user defined audit table that lists users and counts their data manipulation operations. After any user has updated the SAL column in the EMP table, use the audit table to ensure that the number of salary changes does not exceed the maximum permitted for that user.
- Create after delete trigger when user delete record from table it will save in another table called deleted record. It will help to rollback your data
- Create after update trigger to ensure that the data is successfully updated in specific table or not.