|  |  |
| --- | --- |
| **Ex. No: 07** | **PL/SQL – TRIGGER OPERATIONS** |
| Date | 12-03-2024 |

**Objective:**

To execute the given queries using Trigger Operations.

**Description**

**TRIGGER:**

Trigger operations are actions or events that initiate a specific process or set of actions

within a system or program. These triggers can be based on various conditions such as time,

user input, data changes, or external events. When a trigger is activated, it prompts the system

to perform predefined operations or tasks automatically. This automation helps streamline

workflows, improve efficiency, and ensure timely responses to critical events. Common

examples of trigger operations include sending automated emails based on user actions,

updating database records when certain conditions are met, or executing specific tasks in

response to external API calls.

A trigger in SQL is a concise piece of code that runs automatically when certain events occur

on a table. These events can be INSERT, UPDATE, or DELETE operations.

Triggers act as guardians, ensuring data integrity by responding to events and enforcing rules.

SYNTAX:

CREATE TRIGGER Trigger\_Name

[BEFORE | AFTER] [INSERT | UPDATE | DELETE] ON Table\_Name

[FOR EACH ROW | FOR EACH COLUMN]

[trigger\_body];

**TYPES OF TRIGGERS:**

**BEFORE Triggers:**

These triggers fire before the specified event (e.g., INSERT, UPDATE, or DELETE)

takes place. Useful for validation or modification before data changes.

**AFTER Triggers:**

These triggers fire after the event completes successfully.Often used for logging, auditing, or

additional actions.

**Row-Level Triggers (FOR EACH ROW):**

These triggers operate on each affected row individually.Commonly used for enforcing

referential integrity or maintaining history.

**Statement-Level Triggers (FOR EACH STATEMENT):**

These triggers operate on the entire set of affected rows.Useful for bulk operations or

aggregations.

**Nested Triggers:**

Some databases allow triggers to invoke other triggers.Use with caution to avoid infinite

loops.

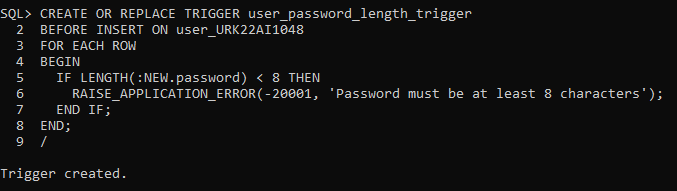
**Instead Of Triggers:**

These triggers replace the default action for a specific event.Often used with views or complex data modifications.

**Questions**

**1. Create a BEFORE INSERT Trigger for the “User” Table that ensures that the**

**passwords are at least 8 Characters.**

****

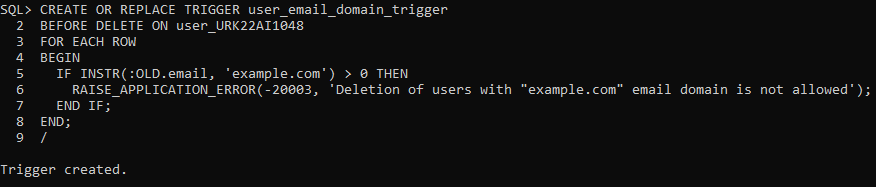
**2. Create a BEFORE UPDATE Trigger for the “User” Table that does not allow email**

**addresses to be null.**

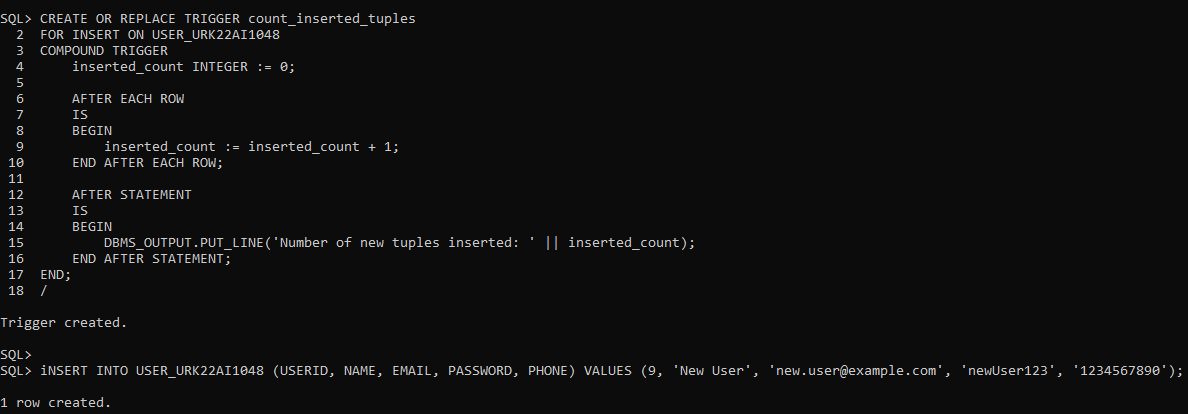
**A computer screen with white text

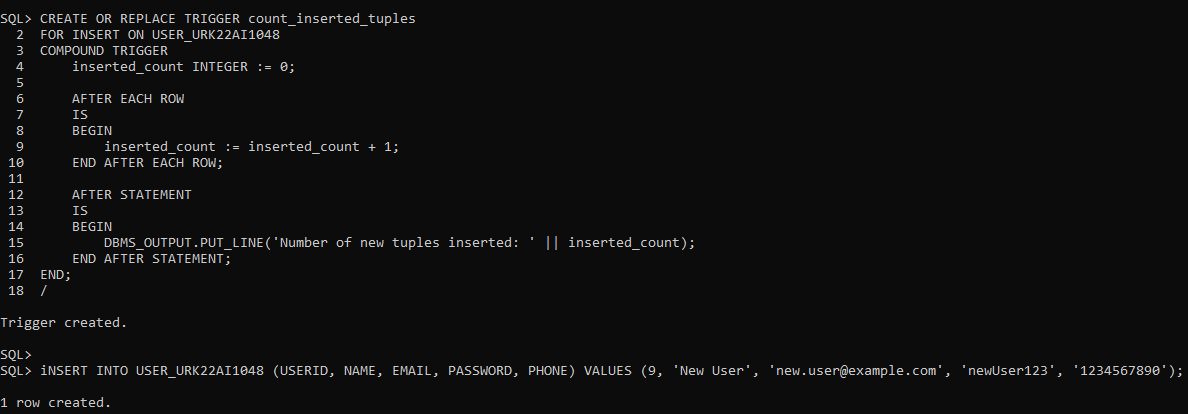
Description automatically generated**

**3. Create a BEFORE DELETE Trigger for the “User” Table that prevents the deletion of users with specific email domains (like &quot;example.com&quot;).**

****

**4. Write an AFTER INSERT trigger to count number of new tuples inserted using each**

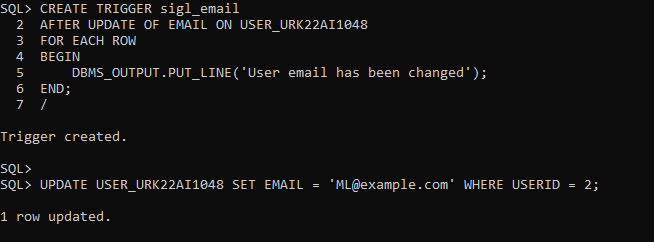
****

****

**A black screen with white text

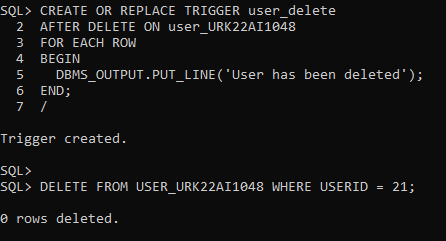
Description automatically generated**

**5. Create an AFTER UPDATE Trigger for the “User” Table that signals when a user&#39;s email is changed.**

****

**6. Create an AFTER DELETE Trigger for the “User” Table that signals when a user is**

**deleted.**

****

**7. Create a BEFORE INSERT Trigger for the “Event” Table that ensures the event&#39;s date is in the future.**

**A screen shot of a computer error

Description automatically generated**

**8. Create a BEFORE UPDATE Trigger for the “Event” Table that Ensures that the**

**event&#39;s time is not set to before 7:00 AM (assuming you use 24-hour format for your Time column).**

**A screen shot of a computer error

Description automatically generated**

**9. Create an AFTER DELETE Trigger for the “Event” Table that signals when an event is deleted.**

**A screen shot of a computer

Description automatically generated**

**10. Create an AFTER UPDATE Trigger for the “Event” Table that signals when an**

**event&#39;s time is changed.**

**A screen shot of a computer

Description automatically generated**

**11. Create a BEFORE INSERT Trigger for the “Venue” Table that ensures the name of the venue is not empty.**

**A computer screen with white text

Description automatically generated**

**12. Create a BEFORE DELETE Trigger for the “Venue” Table that Prevents deletion if the Venueid is less than 105.**

**A screen shot of a computer

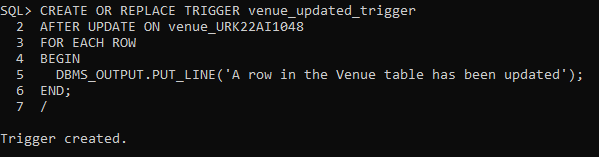
Description automatically generated**

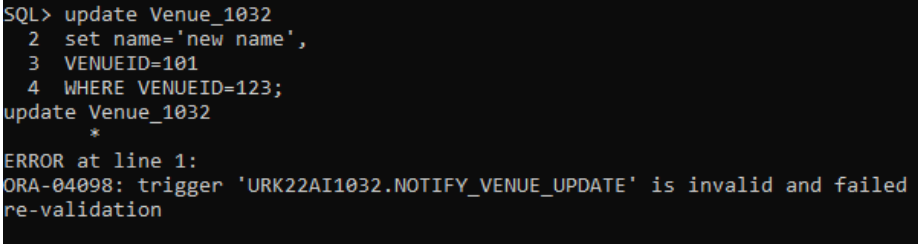
**13. Create an AFTER INSERT Trigger for the “Venue” Table that signals when a new row is added to it.**

**A computer screen with white text

Description automatically generated**

**14. Create an AFTER UPDATE**

****

****

**Result:**

The given queries executed by the set operations and joins successfully.