Assignment - 07

Title: Create Triggers using PL/SQL.

AIM: Write PL/SQL Triggers for the creations of insert trigger, delete trigger and update trigger on the given problem statement.

Objective: To study and use Triggers using MySQL PL/SQL block.

Theory:

- Triggers in PL/SQL
  ↳ Triggers in PL/SQL (adapted for MySQL) are database objects that automatically execute in response to DML events (INSERT, UPDATE, DELETE) on a table. They allow embedding procedural logic directly in the database to enforce rules, audit changes or synchronize data. Unlike procedures / functions, triggers are event-driven and cannot be called directly.

- Advantage of Triggers
  - Data Integrity Enforcement: Automatically validate and correct data during operations.
  - Automation and Consistency: Centralize business logic, ensuring rules apply uniformly accross all application.
  - Auditing and logging: Track changes without modifying application code.
  - Security Enhancements: Restrict unauthorized modifications transparently.

- Usages of Triggers.
  ↳ Its commonly used for:
  - Input validation (eg: checking field lengthss or calculated values like age).
  - Maintaing audit trails (e.g. logging deletions)
  - Updating related tables
  - Enforcing complex constraints not possible with simple CHECKs.
  - Cascading Operations.

- Type of Triggers in PL/SQL
  - Trigger level
    - > Row-Level Triggers: Fire once per affected row (FOR EACH ROW). Access NEW/OLD row data; ideal for per-row validations.
    - > Statement-Level Triggers: Fire once per SQL statement. No row-level access; used for aggregate actions.

  - Trigger Timings
    - > BEFORE Triggers: Run before the event; useful for modifying data (eg. setting defaults) or validating/aborting (via SIGNAL).
    - > AFTER Triggers: Run after the event; suitable for logging or post-validation actions.

  - Trigger Events:
    - > INSERT: On new row addition.
    - > UPDATE: On row modifications
    - > DELETE: On row removal.

- NEW & OLD Clause / Trigger variables.
  - NEW: Pseudorow for new values (INSERT: entire new row; UPDATE: update columns).
  - OLD: Pseudorow for prior values (UPDATE: old row; DELETE: row being deleted)

- Dropping Triggers
  ↳ Use DROP TRIGGER [IF EXISTS] trigger_name; to remove the trigger. This halts its execution but preserves table data.

Syntax:

```
DELIMITER //
CREATE TRIGGER trigger_name
    {BEFORE | AFTER} {INSERT | UPDATE | DELETE}
    ON table_name
    FOR EACH ROW
    [[FOLLOWS | PRECEDES] other_trigger]
BEGIN
    -- Rocedural blocks
    -- Access colcums
    --  Errors & logs handling
END //
DELIMITER ;
```

Input: Trigger for the given problem statement

Output: Data as per request.

Conclusion: Thus, we have learned creating and using triggers in SQL.

FAQs:

1. Enlist Advantages of Triggers?
→ i) Automatic enforcement of rules without app changes.
ii) Built-in auditing for compliance.
iii) Ensures referential and business integrity.
iv) Reduces code duplication.
v) Transparent to users.

2. Enlist Disadvantages of Triggers?
→ i) Debugging challenges (logic hidden in DB)
ii) Potential performance impact on large transactions.
iii) Risk of unexpected & cascading effects.
iv) Difficult to disable temporarily.
v) Increase DB complexity.

3. What are the Applications of Triggers?
→ i) Validation (e.g. age/PRN checks).
ii) Change logging (eg: delete audits)
iii) Data synchronization (e.g: updating views)
iv) Security (e.g: access controls)
v) Workflow automation (e.g: notifications).