Subject: Database Management System & SQL Laboratory

EXPERIMENT NO.: 03

TITLE: Implement Triggers for the given DB.

LEARNING OBJECTIVES:

1. To **Understand the Concept of SQL Triggers**:.
2. To learn Learn Trigger Syntax.

THEORY:

1. A **trigger** is a special type of stored procedure that automatically executes or "fires" in response to certain events on a specified table or view in a database.

2 . Types of Triggers:

- **BEFORE Trigger**: Executes before the event operation (e.g., before an INSERT, UPDATE, or DELETE occurs).
- **AFTER Trigger**: Executes after the event operation has occurred.
- **INSTEAD OF Trigger**: Used on views to substitute the trigger action in place of the triggering operation.

3. Row-Level vs Statement-Level Triggers:

- **Row-level Trigger**: Fires once for each row affected by the triggering event.
- **Statement-level Trigger**: Fires once for the entire operation, regardless of how many rows are affected.

4. Trigger Components:

- **Trigger Event**: The event that causes the trigger to activate (e.g., INSERT, UPDATE, DELETE).
- **Trigger Timing**: Defines when the trigger fires, either BEFORE or AFTER the event.
- **Trigger Action**: The SQL statements that are executed when the trigger fires.

5. NEW and OLD Variables:

- In INSERT triggers, the NEW keyword refers to the new row being inserted.

- In UPDATE triggers, OLD refers to the previous values, and NEW refers to the updated values.
- In DELETE triggers, OLD refers to the row values before deletion.

6. Purpose of Triggers:

- **Data Integrity**: Automatically enforce business rules and data validation.
- **Audit Trails**: Track changes made to a table by recording old and new values.
- **Cascading Changes**: Automatically update or modify other related tables in response to changes.

7.Advantages of Triggers:

- **Automation**: Triggers reduce manual interventions and can automate routine tasks.
- **Consistency**: Triggers help ensure consistency across the database by enforcing rules automatically.
- **Audit Capabilities**: They can log detailed information about data changes without requiring explicit logging code.

8. Disadvantages of Triggers:

- **Performance Overhead**: Excessive or complex triggers can lead to performance degradation as they introduce additional processing for every affected row.
- **Hidden Logic**: Triggers can obscure the logic of an application, making it harder to track where certain changes occur.
- **Recursive Triggers**: Triggers can cause recursion (trigger firing itself repeatedly) if not designed carefully, leading to unintended consequences.

9. Recursive and Nested Triggers:

- Some database systems support **recursive triggers**, where one trigger causes another to fire, and **nested triggers**, where a trigger calls another trigger.

10. Trigger Restrictions:

- Certain operations might not be allowed within triggers (e.g., creating or dropping tables).
- A trigger cannot directly modify the table on which it is defined (in some database systems) to avoid recursive firing.

11. Trigger Syntax:

```
 CREATE TRIGGER trigger_name
{BEFORE | AFTER } {INSERT | UPDATE | DELETE}
ON table_name
FOR EACH ROW
BEGIN
  -- SQL statements (trigger action)
END;
```

12. Trigger Use Cases:

- **Audit Logging**: Automatically record changes to important tables.
- **Enforcing Business Rules**: E.g., preventing negative values in a column.
- **Maintaining Derived Data**: Automatically update aggregated data (e.g., totals or averages) when underlying data changes.
- **Cascading Operations**: Automatically update related tables (e.g., cascading deletes).

NOTE : Please ensure that you also add the Industrial Problem (2) in your submission/document along with the existing content.

References for Theory:

- Silberschatz A., Korth H., Sudarshan S., "Database System Concepts", MGH
- Connally T, Begg C., "Database Systems", Pearson Education
- Raghurama Krishan, "Database Management Systems", McGrawHill
- S.K.Singh, "Database Systems : Concepts, Design and Application", Pearson

CONCLUSION: