DBMS: LAB 6
TRIGGERS, FUNCTIONS AND PROCEDURE Commands
University Fest Management System

OBJECTIVE: To learn and understand Triggers, Procedures, and Functions while executing queries in MySQL.

TRIGGERS

- Automatically execute SQL statements in response to INSERT, UPDATE, or DELETE events on a table.

PROCEDURES

- A set of SQL statements grouped together, executed using CALL, to perform tasks or operations.

FUNCTIONS

- Similar to procedures but must return a single value and can be used directly inside SQL queries

---

INSTRUCTIONS:

As a part of LAB 6, there are 3 tasks to be completed with respect to the case study shared earlier with the ER diagram and Relational Schema of University Fest Management:

- TASK 1: Create and implement Triggers to automatically perform actions on table events (INSERT, UPDATE, DELETE).
- TASK 2: Create and implement Functions to perform calculations or return values that can be used in SQL queries.
- TASK 3: Create and implement Procedures to group SQL statements and execute them using CALL for performing operations or tasks.

● As a part of the submission process, the following are to be submitted:

- A PDF document, containing all the Screenshots for all three tasks as suggested
  ▪ Name of the file: <your SRN>_University_Fest_DB_Lab6.pdf
- The ".sql" file for the same, shall contain all the commands that have been executed in the lab
  ▪ Name of the file: <your SRN>_University_Fest_DB_Lab6.sql

Example:
Refer to the sample submissions given below. This will give you an idea about the details that must be included in your submissions.

NOTE: Screenshots can be taken either from **MySQL Workbench** or **Command Line**.

For every task (Trigger, Function, Procedure), **3 screenshots are required**:

1. **Definition** of the Trigger/Function/Procedure (CREATE statement).
2. **Execution** of the Trigger/Function/Procedure (INSERT/UPDATE/CALL/SELECT as applicable).
3. **Result/Output** after execution (showing the effect or returned value).

Sample submission:

BEFORE:

COMMAND:

AFTER:

TASK 1 [TRIGGERS]:

1. Create a trigger that automatically decreases the total_quantity of an item in the stall_items table whenever a new row is inserted into the purchased table. (hint: Use AFTER INSERT on purchased, and update the corresponding record in stall_items.)

Before: --describe the table/s
Command – sql command
After: --describe the table/s

2. Create a trigger that prevents a participant from purchasing more than 5 units of any single item in a single transaction (i.e., quantity > 5) in the purchased table.
(hint: Use BEFORE INSERT and raise an error if NEW.quantity > 5.)

Before: --describe the table/s
Command – sql command
After: --describe the table/s

TASK 2 [PROCEDURES]:

1. Write a **stored procedure** GetStallSales that takes a stall_id as input and prints the **total revenue** generated from that stall based on the purchased table and item prices from stall_items.
   (hint: Join purchased with stall_items and calculate SUM(price_per_unit * quantity))

Before: --describe the table/s
Command – sql command
After: --describe the table/s

2. Create a procedure RegisterParticipant that registers a participant (SRN) for an event (event_id) by inserting into the registration table. The procedure should take the event_id, SRN, and registration_id as parameters.
   (hint: Use input parameters and basic INSERT.)

Before: --describe the table/s
Command – sql command
After: --describe the table/s

TASK 3 [FUNCTIONS]:

1. Create a **function** GetEventCost that accepts an event_id and returns the **total price** a participant would pay to register for that event (i.e., return the event's price from the event table).
   (hint: Simple SELECT with RETURN.)

Before: --describe the table/s
Command – sql command
After: --describe the table/s

2. Create a **function** `GetParticipantPurchaseTotal(SRN)` that returns the **total amount** a participant has spent across all stalls.
   (hint: Aggregate using `SUM(price_per_unit * quantity)` by joining `purchased` and `stall_items`.)

Before: --describe the table/s
Command – sql command
After: --describe the table/s