- TRIGGERS:

INTRODUCTION:

A trigger in SQL is a special type of stored procedure that is automatically executed in response to a specific event or action in a database. A trigger can be defined to execute either before or after the event that triggers it.

Triggers can be used for a variety of purposes in SQL, including:

- Enforcing business rules and data integrity: Triggers can be used to ensure that data in a database is consistent and follows certain rules. For example, a trigger could be used to enforce a business rule that requires a certain field to be filled out before a new record can be inserted into a table.
- Auditing and logging: Triggers can be used to track changes to data in a database. For example, a trigger could be used to record the user ID and timestamp of any changes made to a specific table.
- Replicating data: Triggers can be used to replicate data between databases. For example, a trigger could be used to automatically update a backup database whenever changes are made to a primary database.
- Cascading updates and deletes: Triggers can be used to automatically update or delete related records in other tables when a record in a primary table is updated or deleted.
- Generating complex calculations or reports: Triggers can be used to perform complex calculations or generate reports based on data in a database. For example, a trigger could be used to generate a report of the top-selling products each month.

Triggers are defined using SQL syntax and are associated with a specific table or view. The syntax for defining a trigger includes the event that triggers the trigger, the table or view that the trigger is associated with, and the code that is executed when the trigger is triggered.

Triggers can be a powerful tool in SQL, but they should be used with caution as they can have a significant impact on performance and can be difficult to debug if not implemented correctly.

Data Description:

Table 1: Attributes	Table 2: Attributes
Student id	Student id
Student name	student name
Mail id	mail id
Mobile no.	mobile no

Problem:

Let's say you are studying SQL for two weeks. In your institute, there is an employee who has been maintaining the student's details and Student Details Backup tables. He / She is deleting the records from the student details after the students completed the course and keeping the backup in the student details backup table by inserting the records every time. You are noticing this daily and now you want to help him/her by not inserting the records for backup purpose when he/she delete the records. Write a

trigger that should be capable enough to insert the student details in the backup table whenever the employee deletes records from the student details table.

Step1: Creating the tables:

```
create table student_details
(student_id int(3) not null,
student_name varchar(30) not null,
mail_id varchar(35) not null,
mobile_number int(12) not null,primary key(student_id)
);
create table student_details_backup
(student_id int(3) not null,
student_name varchar(30) not null,
mail_id varchar(35) not null,
mobile_number int(12), primary key(student_id),foreign key (student_id) references student_details(student_id));
```

Step 2: Inserting the values:

```
insert into student_details values(1,"jenny","anjenny@gmail.com",945215635);
insert into student_details values(2,"tessa","attessa@gmail.com",956248521);
insert into student_details values(3,"terence","teterence@gmail.com",958452156);
insert into student_details values(4,"jasmine","jasmine@gmail.com",985214578);
```

Step 3: Creating the Trigger:

```
delimiter //
create trigger stud_det_backup
before delete
on student_details
for each row
begin
insert into student_details_backup(student_id,student_name,mail_id,mobile_number)
values (old.student_id,old.student_name,old.mail_id,old.mobile_number);
end;
//
delimiter ;
```

EXPLANATION:

This trigger is named "stud_det_backup" that is associated with the "student_details" table. The trigger is set to execute before a row is deleted from the "student_details" table.

When the trigger is triggered, it executes the code inside the "begin" and "end" block. In this case, the trigger inserts a row into the "student_details_backup" table, which is assumed to exist and have the

same structure as the "student_details" table. The values for the inserted row are taken from the row being deleted, which is referred to using the "old" keyword.

The "delimiter" statements are used to change the delimiter used in the SQL code from the default ";" to "//". This is necessary because the trigger definition includes a semicolon, and changing the delimiter prevents the semicolon from ending the entire SQL command prematurely.

Overall, this trigger is used to create a backup of a row in the "student_details" table before it is deleted. This can be useful for auditing and data recovery purposes.

OUTPUT:

Student_detail table Initially:

```
select * from student_details;
```

student_id	student_name	mail_id	mobile_number
1	jenny	anjenny@gmail.com	945215635
2	tessa	attessa@gmail.com	956248521
3	terence	teterence@gmail.com	958452156
4	jasmine	jasmine@gmail.com	985214578

Student_details_backup table:

```
select * from student_details_backup;
```

student_id	student_name	mail_id	mobile_number
NULL	NULL	NULL	NULL

Deleting a student's detail from the student_detail table:

```
delete from student_details where student_id=4;
```

Student_detail table after creating the trigger and deleting a row:

student_id	student_name	mail_id	mobile_number
1	jenny	anjenny@gmail.com	945215635
2	tessa	attessa@gmail.com	956248521
3	terence	teterence@gmail.com	958452156

Student_details_backup table after creating the trigger and deleting a row:

student_id	student_name	mail_id	mobile_number
4	jasmine	jasmine@gmail.com	985214578

CONCLUSION:

The trigger was created successfully. The row with the student id 4 was deleted in the table containing the student detail. Upon deleting the row, the information directly got inserted in the student_details_backup table.