1. Explain the concept of constraints and differentiate between triggers with an real time PL/SQL statement.

ANSWER 1: Constraints are used to enforce rules on the data in a table. They are used to limit the type of data that can go into a table and ensure the accuracy and consistency of the data. There are several types of constraints in SQL, including:

* NOT NULL: Ensures that a column cannot have a NULL value.
* UNIQUE: Ensures that all values in a column are unique.
* PRIMARY KEY: A combination of NOT NULL and UNIQUE. It identifies each row in a table uniquely.
* FOREIGN KEY: Ensures referential integrity for a record in another table.
* CHECK: Ensures that all values in a column satisfy certain conditions.
* DEFAULT: Sets a default value for a column when no value is specified.

Triggers are special types of stored procedures that are automatically executed in response to certain events or actions that occur within a database. They are used to enforce business rules or to perform complex calculations that are not possible with simple SQL statements.

Here’s an example of how to create a trigger:

CREATE TRIGGER trigger\_name

AFTER INSERT

ON table\_name

FOR EACH ROW

BEGIN

-- trigger code goes here

END;

In this example, the trigger is called “trigger\_name” and it is executed after an insert operation on the “table\_name” table. The trigger code goes inside the BEGIN and END statements.

1. What are other uses of triggers?

ANSWER 2: Triggers are used to enforce business rules or to perform complex calculations that are not possible with simple SQL statements. Here are some other uses of triggers:

* Auditing: Triggers can be used to track changes to data in a table. For example, you can create a trigger that logs all changes made to a particular column in a table.
* Data validation: Triggers can be used to validate data before it is inserted into a table. For example, you can create a trigger that checks the format of an email address before it is inserted into a table.
* Data transformation: Triggers can be used to transform data before it is inserted into a table. For example, you can create a trigger that converts all text to uppercase before it is inserted into a table.
* Cascading updates and deletes: Triggers can be used to automatically update or delete related records when a record in another table is updated or deleted.

1. Write the syntax for creating a trigger?

ANSWER 3:

Here’s an example of how to create a trigger:

CREATE TRIGGER trigger\_name

AFTER INSERT

ON table\_name

FOR EACH ROW

BEGIN

-- trigger code goes here

END;

In this example, the trigger is called “trigger\_name” and it is executed after an insert operation on the “table\_name” table. The trigger code goes inside the BEGIN and END statements.

You can replace “AFTER INSERT” with other events such as “BEFORE UPDATE”, “AFTER DELETE”, etc. depending on when you want the trigger to be executed.

1. Create student, department t and college table and execute the below.
   1. Create trigger for auto CGPA
   2. Stop rows inserting marks more than 100 marks
   3. How many triggers are possible for a table?
   4. Update percentage after each semester, using a update trigger

Note: Generate 20 matching records

ANSWER 4: ere’s an example of how to create a trigger for auto CGPA:

CREATE TRIGGER auto\_cgpa

AFTER INSERT ON student

FOR EACH ROW

BEGIN

UPDATE student SET cgpa = (SELECT AVG(marks) FROM marks WHERE student\_id = NEW.id);

END;

This trigger is called “auto\_cgpa” and it is executed after an insert operation on the “student” table. The trigger code updates the “cgpa” column in the “student” table with the average marks of the student.

Here’s an example of how to create a trigger to stop rows inserting marks more than 100 marks:

CREATE TRIGGER check\_marks

BEFORE INSERT ON marks

FOR EACH ROW

BEGIN

IF NEW.marks > 100 THEN

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'Marks cannot be more than 100';

END IF;

END;

This trigger is called “check\_marks” and it is executed before an insert operation on the “marks” table. The trigger code checks if the marks being inserted are more than 100 and raises an error if they are.

The number of triggers that can be created for a table is not limited.

Here’s an example of how to create a trigger to update percentage after each semester:

CREATE TRIGGER update\_percentage

AFTER INSERT ON marks

FOR EACH ROW

BEGIN

UPDATE student SET percentage = (SELECT AVG(marks) FROM marks WHERE student\_id = NEW.student\_id) WHERE id = NEW.student\_id;

END;

This trigger is called “update\_percentage” and it is executed after an insert operation on the “marks” table. The trigger code updates the “percentage” column in the “student” table with the average marks of the student.