Question 1 : Explain the fundamental differences between DDL, DML, and DQL commands in SQL. Provide one example for each type of command.

Answer :

1. DDL (Data Definition Language)

- **Purpose: Defines and modifies the structure of the database objects such as tables, schemas, and indexes.**

- **Effect: Changes made by DDL commands are permanent once executed (auto-committed).**

- **Common Commands: `CREATE`, `ALTER`, `DROP`, `TRUNCATE`**

2. DML (Data Manipulation Language)

- **Purpose: Used to manipulate data stored within database tables (insert, update, delete).**

- **Effect: Changes are not permanent until explicitly committed using `COMMIT`.**

- **Common Commands: `INSERT`, `UPDATE`, `DELETE`**

3. DQL (Data Query Language)

Purpose: Retrieves data from database tables based on certain conditions.

Effect: Only reads data; does not modify it.

Common Command: `SELECT`

Question 2:

What is the purpose of SQL constraints? Name and describe three common types of constraints, providing a simple scenario where each would be useful.

Answer:
SQL constraints are rules applied to table columns that ensure the accuracy, consistency, and integrity of data in a database. They prevent invalid or inconsistent data from being entered and help maintain reliable relationships between tables.

1. Primary Key Constraint:
This ensures that each record in a table is unique and not null. It uniquely identifies every row in a table.
Example: **In a *Student* table, each student has a unique Student ID that cannot be duplicated or left blank.**

2. Foreign Key Constraint:
This establishes a relationship between two tables and ensures that the value in one table corresponds to a valid value in another.
Example: **In an *Orders* table, the Customer ID must exist in the *Customers* table to ensure every order is linked to a valid customer.**

3. Unique Constraint:
This ensures that all values in a particular column are distinct.
Example: **In a *Users* table, each user must have a unique email address so that no two users can register with the same email.**

Question 3:

Explain the difference between LIMIT and OFFSET clauses in SQL. How would you use them together to retrieve the third page of results, assuming each page has 10 records?

Answer:

The LIMIT and OFFSET clauses in SQL are used to control the number of rows returned in a query, especially when dealing with large datasets or implementing pagination.

- **LIMIT: Specifies the maximum number of records to return.**

- **OFFSET: Specifies the number of records to skip before starting to return results.**

These two clauses are often used together to divide results into pages.

Example scenario:
If each page contains 10 records, then:

- **Page 1 → starts at record 0 (OFFSET 0)**

- **Page 2 → starts at record 10 (OFFSET 10)**

- **Page 3 → starts at record 20 (OFFSET 20)**

The LIMIT and OFFSET clauses in SQL are used to control the number of rows returned in a query, especially when dealing with large datasets or implementing pagination.

- **LIMIT: Specifies the maximum number of records to return.**

- **OFFSET: Specifies the number of records to skip before starting to return results.**

These two clauses are often used together to divide results into pages.

Example scenario:
If each page contains 10 records, then:

- **Page 1 → starts at record 0 (OFFSET 0)**

- **Page 2 → starts at record 10 (OFFSET 10)**

- **Page 3 → starts at record 20 (OFFSET 20)**

The LIMIT and OFFSET clauses in SQL are used to control the number of rows returned in a query, especially when dealing with large datasets or implementing pagination.

- **LIMIT: Specifies the maximum number of records to return.**

- **OFFSET: Specifies the number of records to skip before starting to return results.**

These two clauses are often used together to divide results into pages.

Example scenario:
If each page contains 10 records, then:

- **Page 1 → starts at record 0 (OFFSET 0)**

- **Page 2 → starts at record 10 (OFFSET 10)**

- **Page 3 → starts at record 20 (OFFSET 20)**

So, to retrieve the third page of results, we would skip the first 20 records and display the next 10 records.

In simple terms, OFFSET decides where to start, and LIMIT decides how many results to display

Question 4 : What is a Common Table Expression (CTE) in SQL, and what are its main benefits? Provide a simple SQL example demonstrating its usage.

Answer :

A Common Table Expression (CTE) is a temporary, named result set in SQL that exists only during the execution of a single query. It is defined using the WITH keyword and is often used to make complex queries easier to read, maintain, and organize.

Main Benefits of a CTE:

1. **Improves Readability: Breaks down large and complex SQL queries into smaller, understandable parts.**

2. **Reusability: The same result set can be referenced multiple times within a single query.**

3. **Recursive Queries: CTEs can call themselves, which is useful for working with hierarchical data (like employee-manager relationships).**

4. **Simplifies Debugging: Easier to test or modify individual parts of a query.**

Example Scenario (Explanation Only):

Suppose you want to display all employees earning more than the average salary. A CTE can first calculate the average, then use that value to filter employees above it — this keeps the query clean and organized.

Question 5 : Describe the concept of SQL Normalization and its primary goals. Briefly explain the first three normal forms (1NF, 2NF, 3NF).

Answer :

Normalization is a process in SQL used to organize data in a database efficiently by reducing data redundancy and improving data integrity. It involves dividing large tables into smaller, related tables and defining relationships between them using keys.

Primary Goals of Normalization:

1. **To eliminate duplicate data (data redundancy).**

2. **To ensure data consistency and accuracy.**

3. **To make database maintenance easier.**

4. **To improve query performance and storage efficiency.**

The First Three Normal Forms (1NF, 2NF, 3NF):

- **1NF (First Normal Form):**

1NF (First Normal Form):
A table is in 1NF if it contains only atomic (indivisible) values — no repeating groups or arrays.
Example: **Each column should hold a single value like one phone number per row, not multiple numbers in the same cell.**

2NF (Second Normal Form):

A table is in 2NF if it is already in 1NF and all non-key attributes depend entirely on the primary key, not on just part of it.

Example: **If a table uses a composite key (like `StudentID + CourseID`), no column should depend on only `StudentID` or only `CourseID`.**

3NF (Third Normal Form):

A table is in 3NF if it is in 2NF and no non-key attribute depends on another non-key attribute (i.e., no transitive dependency).

Example: **If Student → Department and Department → `HOD`, then `HOD` should not be stored in the student table.**