Introduction to SQL and Advanced Function | Assignment

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

Answer:- The fundamental differences between DDL, DML, and DQL commands in SQL lie in their purposes and operations:

- **DDL (Data Definition Language)** commands define and manage the structure of the database itself, such as creating or modifying tables and schemas.
  Example: `CREATE TABLE Students (ID INT, Name VARCHAR(50));`

- **DML (Data Manipulation Language)** commands manipulate the data stored within the database tables by adding, updating, or deleting records.
  Example: `INSERT INTO Students (ID, Name) VALUES (1, 'Alice');`

- **DQL (Data Query Language)** commands are primarily used to query and retrieve data from the database without modifying it.
  Example: `SELECT * FROM Students;`

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 columns or tables in a database to enforce data accuracy, integrity, and reliability by limiting the type of data that can be entered or changed. They ensure that the data adheres to specified rules, preventing invalid or inconsistent information from being stored.

Three common types of SQL constraints and their use cases are:

- **NOT NULL:** Ensures that a column cannot have null (empty) values. Useful in scenarios like a "User" table where the "email" field must always have a value to contact the user.
- **PRIMARY KEY:** Uniquely identifies each row in a table. It combines NOT NULL and UNIQUE constraints. For example, in an "Orders" table, the "order_id" column would be a primary key to uniquely identify each order record.

- **FOREIGN KEY:** Ensures referential integrity by linking a column in one table to a primary key or unique key in another table. For example, a "Product_ID" column in an "Orders" table references an "ID" column in a "Products" table, ensuring that an order can only contain products that exist.

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 difference between LIMIT and OFFSET clauses in SQL is:

- **LIMIT** specifies the maximum number of rows to return in the result set.
- **OFFSET** specifies the number of rows to skip before starting to return rows.

They are generally used together for pagination, where OFFSET skips the rows of previous pages and LIMIT restricts the number of rows to the page size.

To retrieve the third page of results with 10 records per page, you would skip the first 20 rows (2 pages × 10 records) using OFFSET, then limit the result to 10 rows with LIMIT. The SQL query looks like this:

```
SELECT * FROM table_name ORDER BY some_column LIMIT 10 OFFSET 20;
```

This returns records 21 to 30, representing the third page of results, assuming a page size of 10.

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)** in SQL is a temporary named result set that you can reference within a SELECT, INSERT, UPDATE, or DELETE statement. It is defined using the `WITH` keyword and helps simplify complex queries by breaking them into smaller, more readable parts.

Main Benefits of CTEs

- Improved Readability: CTEs make complex queries easier to read and maintain by separating logic into named blocks.
- Reusability: You can reference the same CTE multiple times within a single query, avoiding the need to repeat subqueries.

- Recursive Queries: CTEs support recursion, which is useful for hierarchical or tree-like data structures.
- Better Organization: CTEs help organize SQL statements, making them easier to debug and understand.

imple SQL Example

Here's a basic example of a CTE that calculates the average salary by department:

```
 WITH AvgSalaryByDept AS (
    SELECT Department, AVG(Salary) AS AvgSalary
    FROM Employees
    GROUP BY Department
)
SELECT Department, AvgSalary
FROM AvgSalaryByDept
WHERE AvgSalary > 50000;
```

Question 5: Describe the concept of SQL Normalization and its primary goals. Briefly explain the first three normal forms (1NF, 2NF, 3NF).
Answer: SQL Normalization is the process of organizing data in a database to reduce redundancy and improve data integrity. Its primary goals are to eliminate duplicate data, minimize data anomalies (such as update, insert, and delete anomalies), simplify database maintenance, and enhance query performance by structuring data efficiently.

The first three normal forms (1NF, 2NF, 3NF) are key stages in this process:

- **First Normal Form (1NF)**: Ensures each column contains atomic (indivisible) values, and each record is unique with no repeating groups or arrays in rows.

- **Second Normal Form (2NF)**: Achieved when the table is in 1NF and all non-key attributes are fully functionally dependent on the entire primary key, which eliminates partial dependency on part of a composite key.

- **Third Normal Form (3NF):** Achieved when the table is in 2NF and all non-key attributes are not only dependent on the primary key but are also independent of each other, removing transitive dependencies.