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

Ans– In SQL, commands are grouped based on **what kind of operation they perform on a database**. The three common categories are **DDL, DML, and DQL**.

1. DDL (Data Definition Language)

DDL commands define, modify, or remove the **structure of database objects** such as tables, schemas, or indexes.

Key characteristics:

- Work on database structure, not the actual data
- Changes are usually **auto-committed** (cannot be rolled back in most DBMS

Example:

```
CREATE TABLE Students (
   StudentID INT,
   Name VARCHAR(50),
   Age INT
);
```

2. DML (Data Manipulation Language)

DML commands are used to **insert, update, delete, or modify data** stored in database tables.

Key characteristics:

- Work on the data inside tables
- Changes can usually be **rolled back** (transaction-controlled)

Example

```
INSERT INTO Students (StudentID, Name, Age)
```

VALUES (1, 'Suraj', 22);

3. DQL (Data Query Language)

Purpose:
DQL commands are used to **retrieve data** from one or more tables.

Key characteristics:

- Read-only operations
- Do not modify database structure or data

Example:

SELECT Name, Age

FROM Students

WHERE Age > 20;

Q-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.

Ans– SQL constraints are rules applied to table columns to **control the type, accuracy, and integrity of data** stored in a database. Their main purpose is to ensure **data consistency, correctness, and reliability** by preventing invalid data from being inserted or updated.

1. PRIMARY KEY Constraint

Description:
Ensures that each row in a table is **uniquely identifiable**. A primary key cannot contain **NULL values** and must be **unique**.

Useful scenario:
In a **Students** table, each student must have a unique student ID.

2. FOREIGN KEY Constraint

Description:
Ensures **referential integrity** by linking a column in one table to the primary key of another table. It prevents invalid references.

Useful scenario:
In an **Orders** table, every order must belong to an existing customer.

3. NOT NULL Constraint

Description:
Prevents a column from storing **NULL values**, ensuring that a value must always be provided.

Useful scenario:
A **User Registration** system where a username is mandatory.

Q-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?

ANS– Both `LIMIT` and `OFFSET` are used to control how many rows are returned and from where the result set starts, which is especially useful for pagination.

1. `LIMIT`

Purpose:
Specifies the **maximum number of rows** to return in the result set.

Example:

```
SELECT * FROM Employees

LIMIT 10;
```

Returns only the first 10 rows.

2. `OFFSET`

Purpose:
Specifies **how many rows to skip** before starting to return rows.

Example:

```
SELECT * FROM Employees

OFFSET 20;
```

Skips the first 20 rows and returns the remaining rows.

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

ANS– A **Common Table Expression (CTE)** is a **temporary, named result set** defined within the execution scope of a single SQL statement. It is created using the `WITH` keyword and can be referenced like a table in the main query.

Main Benefits of Using a CTE

1. **Improved Readability and Clarity**
   Breaks complex queries into logical, easy-to-understand parts.

2. **Reusability Within a Query**
   The same CTE can be referenced multiple times in the main query.

3. **Simplifies Complex Joins and Subqueries**
   Replaces deeply nested subqueries with cleaner, more maintainable SQL.

4. **Supports Recursive Queries**
   CTEs can be recursive, making them ideal for hierarchical data (e.g., organizational charts).

EXAMPLE:

```
WITH AvgSalary AS (
    SELECT AVG(Salary) AS avg_salary
    FROM Employees
)
SELECT Name, Salary
FROM Employees
WHERE Salary > (SELECT avg_salary FROM AvgSalary);
```

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

ANS– SQL normalization is the process of organizing data in a relational database to reduce redundancy and improve data integrity. It involves dividing large tables into smaller, well-structured tables and defining relationships between them using keys.

Primary Goals of Normalization

1. **Eliminate data redundancy**
   Avoid storing the same data in multiple places.

2. **Ensure data integrity and consistency**
   Prevent update, insert, and delete anomalies.

3. **Improve database maintainability**
   Make the structure easier to understand and modify.

4. **Efficient data storage**
   Reduce unnecessary data duplication.

1. First Normal Form (1NF)

Rule:

- Each column must contain **atomic (indivisible) values**
- No repeating groups or multi-valued attributes
- Each record must be uniquely identifiable

Second Normal Form (2NF)

Rule:

- Must be in **1NF**
- All **non-key attributes** must be **fully dependent** on the **entire primary key**
- No **partial dependency** (applies to composite keys)

Third Normal Form (3NF)

Rule:

- Must be in **2NF**
- No **transitive dependency**
  (Non-key attributes should not depend on other non-key attributes)