# Assignment Code: DA-AG-014 Introduction to SQL and Advanced Functions | Assignment

**Yogita Hiwale**

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

**Answer:**
SQL commands are divided into categories based on their purpose:

**DDL (Data Definition Language)**

Purpose: Database schema create/modify/delete karna

Example:

CREATE TABLE Students (ID INT, Name VARCHAR(50));

**DML (Data Manipulation Language)**

Table data insert/update/delete

Example:

INSERT INTO Students VALUES (1, 'Rahul');

**DQL (Data Query Language)**

Data retrieve

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 ensure correct, valid, and reliable data inside tables.

**PRIMARY KEY**

Unique + Not Null

Scenario: Every customer must have a unique CustomerID.

**FOREIGN KEY**

Links two tables

Scenario: Orders table uses CustomerID (FK) to match Customers table.

**UNIQUE**

Value must not repeat

Scenario: Email address must be unique for every user.

**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:**

**LIMIT:**

To records fetch

**OFFSET:**

To skip specific number of  records

3rd page, each page = 10 records

3rd page ka start = (3 − 1) × 10 = 20

```
SELECT *

FROM table_name

LIMIT 10 OFFSET 20;
```

**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.**

A CTE (Common Table Expression) is a temporary result set in SQL that is defined using the WITH clause.
It exists only for the duration of the query and can be referenced just like a temporary table within the main query.
CTEs make complex queries easier to write, read, and maintain.

**Benefits of a CTE:**

1. **Improves readability**
   CTEs break down complex queries into smaller, more understandable parts.
2. **Avoids repeating subqueries**
   The same logic doesn't need to be written multiple times in the query.
3. **Helps write recursive queries**
   Useful for hierarchical data like employee–manager structures or category trees.
4. **Organizes SQL code better**
   Makes long queries clean, structured, and more manageable.

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

SQL Normalization is a database design technique used to organize data in a structured way.
Its main purpose is to reduce data redundancy (duplicate data) and improve data integrity by dividing large tables into smaller, related tables.

**Primary Goals of Normalization:**

1. Eliminate redundant (duplicate) data
   Prevents storing the same information in multiple places.

2. Ensure data integrity
   Helps maintain accuracy and consistency of data.
3. Organize data efficiently
   Structures the database so that updates, insertions, and deletions happen smoothly.
4. Reduce anomalies
   Avoids update, insert, and delete anomalies.

**First Three Normal Forms (1NF, 2NF, 3NF)**

**First Normal Form (1NF)**

A table is in 1NF if:

Each column contains atomic (indivisible) values.

There are no repeating groups or arrays.

Each record is unique.

Example:
A customer's phone numbers must be stored in separate rows, not in one column as: "9876543210, 9123456789"

**Second Normal Form (2NF)**

A table is in 2NF if:

It is already in 1NF

There is no partial dependency, meaning:

Non-key columns must depend on the entire primary key, not part of it.

Applies mainly to tables with composite primary keys.

**Third Normal Form (3NF)**

**A table is in 3NF if:**

It is already in 2NF

There are no transitive dependencies, meaning:

Non-key columns should not depend on another non-key column.

Example:
If CustomerID → City and City → Pincode, then Pincode depends indirectly on CustomerID, which violates 3NF.

**/*Question 6 : Create a database named ECommerceDB and perform the following tasks:*/**

```
CREATE DATABASE ECommerceDB;

USE ECommerceDB;
```

**/* 1. Create the following tables with appropriate data types and constraints:**

● Categories

○ CategoryID (INT, PRIMARY KEY)

○ CategoryName (VARCHAR(50), NOT NULL, UNIQUE) */

```
CREATE TABLE Categories (
    CategoryID INT PRIMARY KEY,
    CategoryName VARCHAR(50) NOT NULL UNIQUE
);
```

/*● Products

○ ProductID (INT, PRIMARY KEY)

○ ProductName (VARCHAR(100), NOT NULL, UNIQUE)

○ CategoryID (INT, FOREIGN KEY → Categories)

○ Price (DECIMAL(10,2), NOT NULL)

```
○ StockQuantity (INT) */

CREATE TABLE Products (
    ProductID INT PRIMARY KEY,
    ProductName VARCHAR(100) NOT NULL UNIQUE,
    CategoryID INT,
    Price DECIMAL(10,2) NOT NULL,
    StockQuantity INT,
    FOREIGN KEY (CategoryID) REFERENCES Categories(CategoryID)
);

/* ● Customers
○ CustomerID (INT, PRIMARY KEY)
○ CustomerName (VARCHAR(100), NOT NULL)
○ Email (VARCHAR(100), UNIQUE)
○ JoinDate (DATE) */

CREATE TABLE Customers (
    CustomerID INT PRIMARY KEY,
    CustomerName VARCHAR(100) NOT NULL,
    Email VARCHAR(100) UNIQUE,
    JoinDate DATE
);

/* ● Orders
○ OrderID (INT, PRIMARY KEY)
```

○ CustomerID (INT, FOREIGN KEY → Customers)

○ OrderDate (DATE, NOT NULL)

○ TotalAmount (DECIMAL(10,2)) */

```
CREATE TABLE Orders (
    OrderID INT PRIMARY KEY,
    CustomerID INT,
    OrderDate DATE NOT NULL,
    TotalAmount DECIMAL(10,2),
    FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)
);
```

**/* 2. Insert the following records into each table**

**● Categories */**

```
INSERT INTO Categories VALUES
(1, 'Electronics'),
(2, 'Books'),
(3, 'Home Goods'),
(4, 'Apparel');

-- Products

INSERT INTO Products VALUES
(101, 'Laptop Pro', 1, 1200.00, 50),
(102, 'SQL Handbook', 2, 45.50, 200),
```

(103, 'Smart Speaker', 1, 99.99, 150),

(104, 'Coffee Maker', 3, 75.00, 80),

(105, 'Novel : The Great SQL', 2, 25.00, 120),

(106, 'Wireless Earbuds', 1, 150.00, 100),

(107, 'Blender X', 3, 120.00, 60),

(108, 'T-Shirt Casual', 4, 20.00, 300);

-- Customers

INSERT INTO Customers VALUES

(1, 'Alice Wonderland', 'alice@example.com', '2023-01-10'),

(2, 'Bob the Builder', 'bob@example.com', '2022-11-25'),

(3, 'Charlie Chaplin', 'charlie@example.com', '2023-03-01'),

(4, 'Diana Prince', 'diana@example.com', '2021-04-26');

-- Orders

INSERT INTO Orders VALUES

(1001, 1, '2023-04-26', 1245.50),

(1002, 2, '2023-10-12', 99.99),

(1003, 1, '2023-07-01', 145.00),

(1004, 3, '2023-01-14', 150.00),

(1005, 2, '2023-09-24', 120.00),

(1006, 1, '2023-06-19', 20.00);

**/* Question 7 : Generate a report showing CustomerName, Email, and the TotalNumberofOrders for each customer. Include customers who have not placed any orders, in which case their TotalNumberofOrders should be 0. Order the results by CustomerName.**

Answer : */

```
SELECT
   c.CustomerName,
   c.Email,
   COUNT(o.OrderID) AS TotalNumberOfOrders
FROM Customers c
LEFT JOIN Orders o
   ON c.CustomerID = o.CustomerID
GROUP BY c.CustomerID, c.CustomerName, c.Email
ORDER BY c.CustomerName;
```

**/* Question 8 : Retrieve Product Information with Category: Write a SQL query to display the ProductName, Price, StockQuantity, and CategoryName for all products. Order the results by CategoryName and then ProductName alphabetically.**

**Answer : */**

```
SELECT
   p.ProductName,
```

```
    p.Price,
    p.StockQuantity,
    c.CategoryName
FROM Products p
JOIN Categories c
    ON p.CategoryID = c.CategoryID
ORDER BY c.CategoryName, p.ProductName;
```

**/* Question 9 : Write a SQL query that uses a Common Table Expression (CTE) and a Window Function (specifically ROW_NUMBER() or RANK()) to display the CategoryName, ProductName, and Price for the top 2 most expensive products in each CategoryName. */**

```
WITH RankedProducts AS (
    SELECT
        c.CategoryName,
        p.ProductName,
        p.Price,
        ROW_NUMBER() OVER (
            PARTITION BY c.CategoryName
            ORDER BY p.Price DESC
        ) AS rn
    FROM Products p
    JOIN Categories c ON p.CategoryID = c.CategoryID
)
SELECT CategoryName, ProductName, Price
```

```
FROM RankedProducts
WHERE rn <= 2;
```

**/* Question 10 : You are hired as a data analyst by Sakila Video Rentals, a global movie rental company. The management team is looking to improve decision-making by analyzing existing customer, rental, and inventory data.**

**Using the Sakila database, answer the following business questions to support key strategic initiatives.**

**Tasks & Questions:**

**1. Identify the top 5 customers based on the total amount they've spent. Include customer name, email, and total amount spent.**

**2. Which 3 movie categories have the highest rental counts? Display the category name and number of times movies from that category were rented.**

**3. Calculate how many films are available at each store and how many of those have never been rented.**

**4. Show the total revenue per month for the year 2023 to analyze business seasonality.**

**5. Identify customers who have rented more than 10 times in the last 6 months. */**

```
USE sakila;

/* 1. Identify the top 5 customers based on the total amount they've spent. Include customer
name, email, and total amount spent.  */

SELECT
   c.first_name,
   c.last_name,
```

```
    c.email,
    SUM(p.amount) AS total_spent
FROM payment p
JOIN customer c ON p.customer_id = c.customer_id
GROUP BY c.customer_id
ORDER BY total_spent DESC
LIMIT 5;

/* 2. Which 3 movie categories have the highest rental counts? Display the category name
and number of times movies from that category were rented.  */

SELECT
    cat.name AS CategoryName,
    COUNT(*) AS RentalCount
FROM rental r
JOIN inventory i ON r.inventory_id = i.inventory_id
JOIN film_category fc ON i.film_id = fc.film_id
JOIN category cat ON fc.category_id = cat.category_id
GROUP BY cat.category_id
ORDER BY RentalCount DESC
LIMIT 3;


/* 3. Calculate how many films are available at each store and how many of those have
never been rented.  */
```

```
SELECT
   s.store_id,
   COUNT(i.inventory_id) AS total_films,
   COUNT(i.inventory_id)
   - COUNT(r.rental_id) AS never_rented
FROM store s
JOIN inventory i ON s.store_id = i.store_id
LEFT JOIN rental r ON i.inventory_id = r.inventory_id
GROUP BY s.store_id;


/* 4. Show the total revenue per month for the year 2023 to analyze business seasonality.   */
SELECT
   MONTH(payment_date) AS month,
   SUM(amount) AS total_revenue
FROM payment
WHERE YEAR(payment_date) = 2023
GROUP BY MONTH(payment_date)
ORDER BY month;


/* 5. Identify customers who have rented more than 10 times in the last 6 months. */

SELECT
   c.first_name,
   c.last_name,
```

```
    c.email,
    COUNT(r.rental_id) AS total_rentals
FROM rental r
JOIN customer c ON r.customer_id = c.customer_id
WHERE r.rental_date >= DATE_SUB(CURDATE(), INTERVAL 6 MONTH)
GROUP BY c.customer_id
HAVING total_rentals > 10;
```