

# Introduction to SQL Queries and Practical Examples

## 1. Introduction to SQL

- **SQL (Structured Query Language)** is a standard language for interacting with relational databases.
  - **Key operations** in SQL are summarized by **CRUD**:
    - **C** – Create → INSERT
    - **R** – Read → SELECT
    - **U** – Update → UPDATE
    - **D** – Delete → DELETE
  - SQL is used in databases like **MySQL, Oracle, SQL Server, PostgreSQL**, etc.
- 

## 2. SQL Query Types

### 2.1 Data Definition Language (DDL)

Used to define database structure:

- **CREATE** → Create tables or databases
- **ALTER** → Modify table structure
- **DROP** → Delete tables or databases
- **TRUNCATE** → Remove all data from a table

#### Example:

-- Create a table

```
CREATE TABLE Students (  
    StudentID INT PRIMARY KEY,  
    Name VARCHAR(50),  
    Age INT,  
    Grade VARCHAR(5)  
);
```

-- Alter table to add a new column

```
ALTER TABLE Students ADD COLUMN Email VARCHAR(50);
```

-- Drop table

```
DROP TABLE Students;
```

### 2.2 Data Manipulation Language (DML)

Used to manipulate data in tables:

- **INSERT** → Add new records
- **UPDATE** → Modify existing records
- **DELETE** → Remove records

#### Example:

-- Insert data

```
INSERT INTO Students (StudentID, Name, Age, Grade)  
VALUES (1, 'Ali', 12, 'A');
```

```
-- Update data
UPDATE Students
SET Age = 13
WHERE StudentID = 1;
```

```
-- Delete data
DELETE FROM Students
WHERE StudentID = 1;
```

## 2.3 Data Query Language (DQL)

- **SELECT** → Read/retrieve data
- You can use **WHERE, ORDER BY, GROUP BY, HAVING** clauses.

### Example:

```
-- Select all data
SELECT * FROM Students;
```

```
-- Select specific columns
SELECT Name, Grade FROM Students;
```

```
-- Filtering with WHERE
SELECT * FROM Students
WHERE Age > 12;
```

```
-- Ordering
SELECT * FROM Students
ORDER BY Name ASC;
```

```
-- Aggregate functions
SELECT Grade, COUNT(*) AS TotalStudents
FROM Students
GROUP BY Grade
HAVING COUNT(*) > 1;
```

## 2.4 Data Control Language (DCL)

- **GRANT** → Give permissions
- **REVOKE** → Remove permissions

### Example:

```
GRANT SELECT, INSERT ON Students TO 'user1';
REVOKE INSERT ON Students FROM 'user1';
```

---

# 3. SQL Query Concepts

## 3.1 Filtering Data

- Use **WHERE** with operators: =, <>, >, <, >=, <=
- Use **AND, OR, NOT** for multiple conditions

```
SELECT * FROM Students
WHERE Age >= 12 AND Grade = 'A';
```

### 3.2 Pattern Matching

- Use **LIKE** with % and \_

-- Names starting with 'A'

```
SELECT * FROM Students  
WHERE Name LIKE 'A%';
```

-- Names with second letter 'l'

```
SELECT * FROM Students  
WHERE Name LIKE '_l%';
```

### 3.3 Null Handling

- IS NULL and IS NOT NULL

```
SELECT * FROM Students  
WHERE Email IS NULL;
```

### 3.4 Joins

- Combine multiple tables
  - **INNER JOIN** → Only matching rows
  - **LEFT JOIN** → All rows from left table
  - **RIGHT JOIN** → All rows from right table
  - **FULL OUTER JOIN** → All rows from both tables

#### Example:

```
CREATE TABLE Classes (  
    ClassID INT PRIMARY KEY,  
    ClassName VARCHAR(50)  
);
```

-- Inner Join example

```
SELECT Students.Name, Classes.ClassName  
FROM Students  
INNER JOIN Classes ON Students.StudentID = Classes.ClassID;
```

### 3.5 Subqueries

- Nested query inside another query

```
SELECT Name FROM Students  
WHERE Age > (SELECT AVG(Age) FROM Students);
```

### 3.6 SQL Functions

- **Aggregate:** SUM, COUNT, AVG, MIN, MAX
- **String:** CONCAT, LENGTH, UPPER, LOWER
- **Date:** NOW(), CURDATE(), DATE\_ADD()

#### Example:

```
SELECT COUNT(*) AS TotalStudents FROM Students;  
SELECT CONCAT(Name, ' (', Grade, ')') AS Info FROM Students;  
SELECT NOW() AS CurrentDateTime;
```

## 4. Practical Exercises

Here are exercises to practice SQL:

**Exercise 1:** Create a table Employees with columns: EmpID, Name, Salary, Department.

**Exercise 2:** Insert 5 employee records.

**Exercise 3:** Retrieve all employees with salary > 50000.

**Exercise 4:** Update salary of employee EmpID=3 to 60000.

**Exercise 5:** Delete an employee record with EmpID=5.

**Exercise 6:** Retrieve employees ordered by Salary DESC.

**Exercise 7:** Find the total salary of each department.

**Exercise 8:** Create a table Departments and join with Employees to list employee names with department names.

---

## 5. Tips for Writing Efficient Queries

1. Always use **WHERE** to filter data.
2. Avoid **SELECT \*** in production queries.
3. Index columns used frequently in joins or where conditions.
4. Use **EXPLAIN** to analyze query execution.
5. Test queries with sample data first.