

# Introduction to SQL

## What is SQL?

SQL stands for **Structured Query Language**. It is a standard programming language used to create, manage, and manipulate databases. SQL is mainly used to communicate with relational database systems such as MySQL, Oracle Database, Microsoft SQL Server, and PostgreSQL.

SQL allows users to store data, retrieve data, update existing data, delete unwanted data, and control access to the data stored in a database. It is one of the most important languages used in software development, data analysis, and database administration.

## What is a Database?

A database is an organized collection of data that is stored electronically. It allows information to be stored in a structured format so that it can be easily accessed, managed, and updated.

For example, a college database may store student information such as student ID, name, age, course, and marks. A bank database may store customer details, account numbers, and transaction records.

Databases help in managing large amounts of data efficiently and securely.

## What is RDBMS?

RDBMS stands for Relational Database Management System. It is a type of database management system that stores data in the form of tables. Each table consists of rows and columns.

In a table:

- Each row represents a record.
- Each column represents a field or attribute.

The term “relational” means that tables can be related to each other using keys such as primary key and foreign key.

## Structure of a Table

A table in SQL contains rows and columns.

For example, a Student table may look like this:

ID | Name | Age

1 | Rahul | 20

2 | Anita | 21

Here:

- ID, Name, and Age are columns.
- Each row represents one student's record.

Tables are the basic structure where data is stored in SQL.

## Types of SQL Commands

SQL commands are divided into five major categories based on their purpose.

These categories are:

1. DDL (Data Definition Language)
2. DML (Data Manipulation Language)
3. DQL (Data Query Language)
4. DCL (Data Control Language)
5. TCL (Transaction Control Language)

### 1.Data Definition Language (DDL)

DDL commands are used to define and modify the structure of the database. These commands deal with tables and database schema.

Common DDL commands include CREATE, ALTER, DROP, and TRUNCATE.

CREATE is used to create a new table or database.

ALTER is used to modify an existing table structure.

DROP is used to permanently delete a table.

TRUNCATE removes all records from a table but keeps the table structure.

DDL commands change the structure of the database.

#### CREATE

Used to create a database or table.

### ◆ Create Database

```
CREATE DATABASE college;
```

### ◆ Use Database

```
USE college;
```

### ◆ Create Table

```
CREATE TABLE students (  
    id INT PRIMARY KEY,  
    name VARCHAR(50),  
    age INT,  
    department VARCHAR(50)  
);
```

## 2. DROP

Used to delete database or table permanently.

### ◆ Drop Table

```
DROP TABLE students;
```

### ◆ Drop Database

```
DROP DATABASE college;
```

## 3. ALTER

Used to modify table structure.

### ◆ Add Column

```
ALTER TABLE students  
ADD email VARCHAR(100);
```

### ◆ Modify Column

```
ALTER TABLE students
```

MODIFY age INT;

### ◆ Drop Column

ALTER TABLE students

DROP email;

## 4. TRUNCATE

Deletes all records but keeps table structure.

TRUNCATE TABLE students;

## 2.Data Manipulation Language (DML)

DML commands are used to manage the data inside the tables.

Common DML commands include INSERT, UPDATE, and DELETE.

INSERT is used to add new records into a table.

UPDATE is used to modify existing records.

DELETE is used to remove records from a table.

DML commands affect the actual data stored in the database.

### 1. INSERT

Used to add data.

INSERT INTO students (id, name, age, department)

VALUES (1, 'Anita', 20, 'CSE');

Insert multiple rows:

INSERT INTO students VALUES

(2, 'Rahul', 21, 'ECE'),

(3, 'Sneha', 22, 'IT');

### 2. SELECT

Used to retrieve data.

#### ◆ **Select all**

```
SELECT * FROM students;
```

#### ◆ **Select specific columns**

```
SELECT name, age FROM students;
```

#### ◆ **With condition**

```
SELECT * FROM students
```

```
WHERE age > 20;
```

### **3. UPDATE**

Used to modify data.

```
UPDATE students
```

```
SET age = 23
```

```
WHERE id = 3;
```

### **4. DELETE**

Used to delete specific records.

```
DELETE FROM students
```

```
WHERE id = 2;
```

### **3.Data Query Language (DQL)**

DQL is mainly used to retrieve data from the database.

The most important DQL command is SELECT.

The SELECT statement is used to fetch data from one or more tables. It can retrieve all records or specific records based on conditions.

For example, we can select all students or select only students whose age is greater than 18.

DQL is one of the most commonly used parts of SQL.

### **WHERE with Operators**

**SELECT \* FROM students**

**WHERE department = 'CSE';**

**SELECT \* FROM students**

**WHERE age BETWEEN 20 AND 22;**

**SELECT \* FROM students**

**WHERE name LIKE 'A%';**

### **ORDER BY**

**Sort results.**

**SELECT \* FROM students**

**ORDER BY age DESC;**

### **DISTINCT**

**Remove duplicates.**

**SELECT DISTINCT department FROM students;**

## **4.Data Control Language (DCL)**

DCL commands are used to control access to the database.

The main DCL commands are GRANT and REVOKE.

GRANT is used to give permission to a user.

REVOKE is used to remove permission from a user.

DCL ensures database security by controlling who can access or modify the data.

## **5. Transaction Control Language (TCL)**

TCL commands are used to manage transactions in a database.

A transaction is a sequence of operations performed as a single unit.

Common TCL commands are COMMIT, ROLLBACK, and SAVEPOINT.

COMMIT saves changes permanently.

ROLLBACK undoes changes if an error occurs.

SAVEPOINT sets a point in a transaction to which we can later return.

TCL helps maintain data integrity.

### **Transactions (TCL)**

#### **COMMIT**

COMMIT;

#### **ROLLBACK**

ROLLBACK;

#### **SAVEPOINT**

SAVEPOINT sp1;

## **SQL Constraints**

Constraints are rules applied to table columns to maintain data accuracy and integrity.

Some common constraints are:

NOT NULL ensures that a column cannot have empty values.

UNIQUE ensures that all values in a column are different.

PRIMARY KEY uniquely identifies each record.

FOREIGN KEY links one table with another table.

CHECK ensures that values meet a specific condition.

DEFAULT assigns a default value if no value is provided.

Constraints help in maintaining correct and valid data.

## **Keys in SQL**

Keys are used to uniquely identify records and establish relationships between tables.

A Primary Key uniquely identifies each record in a table. It cannot contain NULL values and must be unique.

A Foreign Key creates a relationship between two tables. It refers to the primary key of another table.

Keys help maintain relational structure in databases.

## **SQL Operators**

SQL operators are used to perform operations in queries.

Arithmetic operators perform mathematical operations like addition and subtraction.

Comparison operators compare values such as equal to, greater than, or less than.

Logical operators like AND, OR, and NOT combine multiple conditions.

Operators are mainly used in WHERE clauses to filter data.

## **Aggregate Functions**

Aggregate functions perform calculations on multiple rows and return a single result.

Common aggregate functions include:

COUNT() – Counts the number of records

SUM() – Calculates total value

AVG() – Calculates average

MAX() – Finds maximum value

MIN() – Finds minimum value

Aggregate functions are often used with GROUP BY.

### **COUNT**

```
SELECT COUNT(*) FROM students;
```

### **SUM**

```
SELECT SUM(age) FROM students;
```

### **AVG**



```
SELECT AVG(age) FROM students;
```

## **MAX**

```
SELECT MAX(age) FROM students;
```

## **MIN**

```
SELECT MIN(age) FROM students;
```

## **GROUP BY Clause**

The GROUP BY clause groups rows that have the same values in specified columns.

It is mainly used with aggregate functions to summarize data.

For example, we can group students by age and count how many students belong to each age group.

### **GROUP BY**

Used with aggregate functions.

```
SELECT department, COUNT(*)
```

```
FROM students
```

```
GROUP BY department;
```

## **HAVING Clause**

The HAVING clause is used to filter grouped results.

It works with GROUP BY and aggregate functions.

While WHERE filters rows before grouping, HAVING filters groups after grouping.

### **HAVING**

Used to filter grouped data.

```
SELECT department, COUNT(*)
```

```
FROM students  
GROUP BY department  
HAVING COUNT(*) > 1;
```

## **ORDER BY Clause**

The ORDER BY clause is used to sort results in ascending or descending order.

It helps in arranging output in a meaningful way.

### **Constraints**

Used to restrict data.

#### **NOT NULL**

```
name VARCHAR(50) NOT NULL
```

#### **UNIQUE**

```
email VARCHAR(100) UNIQUE
```

#### **PRIMARY KEY**

```
id INT PRIMARY KEY
```

#### **FOREIGN KEY**

```
CREATE TABLE marks (  
    mark_id INT PRIMARY KEY,  
    student_id INT,  
    FOREIGN KEY (student_id) REFERENCES students(id)  
);
```

## **Joins in SQL**

Joins are used to combine data from two or more tables.

Different types of joins include:

INNER JOIN returns matching records from both tables.

LEFT JOIN returns all records from the left table and matching records from the right table.

RIGHT JOIN returns all records from the right table and matching records from the left table.

FULL JOIN returns all records from both tables.

Joins are very important for relational databases.

## **Joins**

Used to combine data from multiple tables.

### **INNER JOIN**

```
SELECT students.name, marks.mark_id
```

```
FROM students
```

```
INNER JOIN marks
```

```
ON students.id = marks.student_id;
```

### **LEFT JOIN**

```
SELECT students.name, marks.mark_id
```

```
FROM students
```

```
LEFT JOIN marks
```

```
ON students.id = marks.student_id;
```

### **RIGHT JOIN**

```
SELECT students.name, marks.mark_id
```

```
FROM students
```

```
RIGHT JOIN marks
```

```
ON students.id = marks.student_id;
```

## **Views in SQL**

A view is a virtual table based on a SELECT query.

It does not store data itself but displays data from one or more tables.

Views improve security and simplify complex queries.

## **Views**

A **View** is a virtual table.

#### **Create View**

```
CREATE VIEW cse_students AS  
SELECT * FROM students  
WHERE department = 'CSE';
```

#### **Use View**

```
SELECT * FROM cse_students;
```

#### **Drop View**

```
DROP VIEW cse_students;
```

### **Normalization**

Normalization is the process of organizing data to reduce redundancy.

It divides large tables into smaller tables and establishes relationships.

The main normal forms are 1NF, 2NF, and 3NF.

Normalization improves database efficiency and consistency.