Name: Sonal Bhujbal

Database Management System

SQL Basic

SQL is designed for beginners and professionals.

SQL (Structured Query Language) is used to perform operations on the records stored in the database, such as updating records, inserting records, deleting records, creating and modifying database tables, views, etc.

SQL is not a database system, but it is a query language.

## **SQL Commands**

The most common SQL commands which are highly used are mentioned below:

1. CREATE command
2. UPDATE command
3. DELETE command
4. SELECT command
5. DROP command
6. INSERT command

## **What is Database**

## Database Management System

Database management The database is a collection of inter-related data which is used to retrieve, insert and delete the data efficiently. It is also used to organize the data in the form of a table, schema, views, and reports, etc.

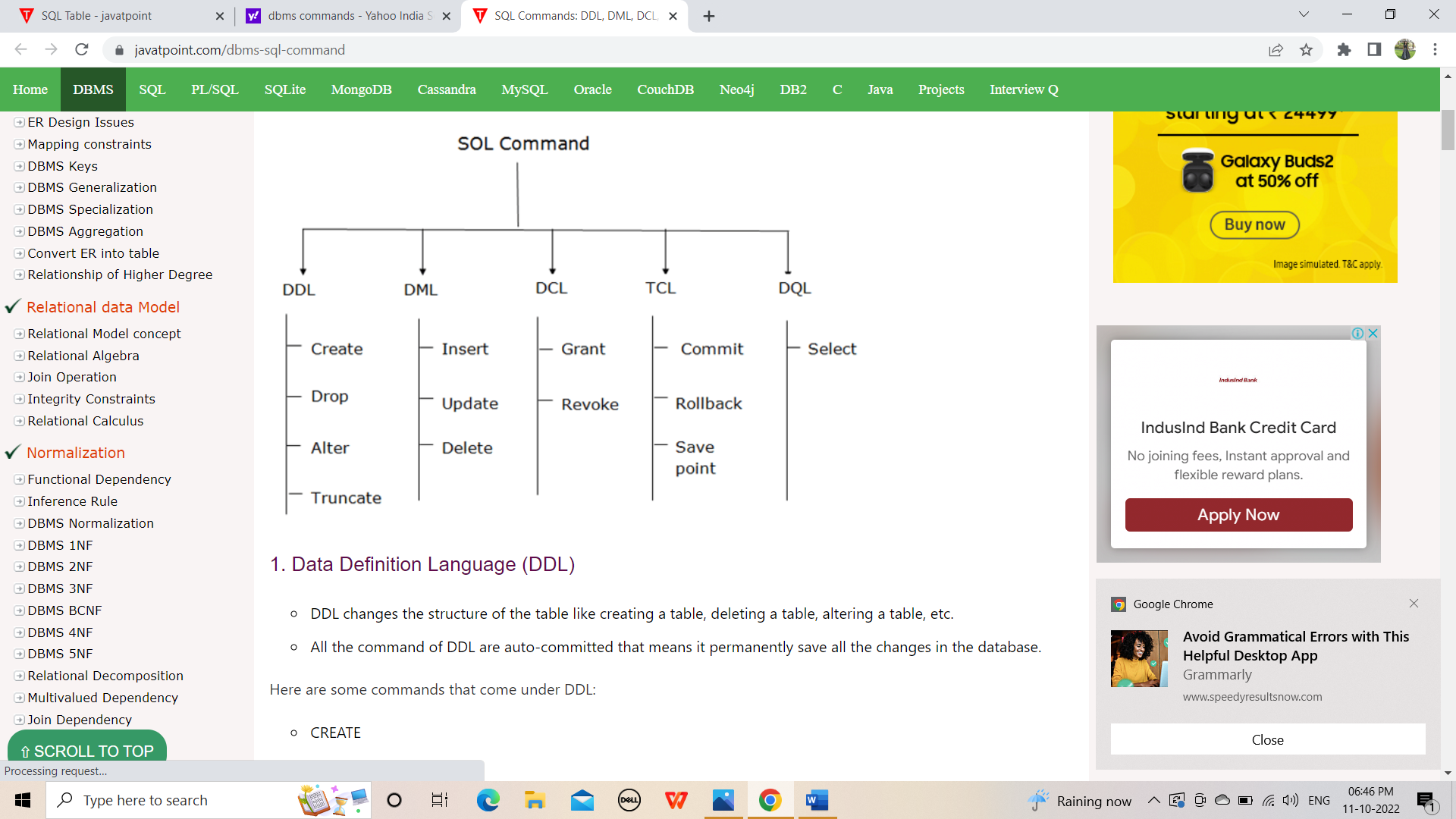
system is a software which is used to manage the database. For example: [MySQL](https://www.javatpoint.com/mysql-tutorial), [Oracle](https://www.javatpoint.com/oracle-tutorial), etc are a very popular commercial database which is used in different applications.

# SQL Commands

* SQL commands are instructions. It is used to communicate with the database. It is also used to perform specific tasks, functions, and queries of data.
* SQL can perform various tasks like create a table, add data to tables, drop the table, modify the table, set permission for users.

## **Types of SQL Commands**

There are five types of SQL commands: DDL, DML, DCL, TCL, and DQL.



### 1. Data Definition Language (DDL)

* DDL changes the structure of the table like creating a table, deleting a table, altering a table, etc.
* All the command of DDL are auto-committed that means it permanently save all the changes in the database.

Here are some commands that come under DDL:

* CREATE
* ALTER
* DROP
* TRUNCATE

1. **CREATE** It is used to create a new table in the database.

**Syntax:**

CREATE TABLE TABLE\_NAME (COLUMN\_NAME DATATYPES [....]);

**Example:**

CREATE TABLE EMPLOYEE (Name VARCHAR2(20), Email VARCHAR2(100), DOB DATE);

**b. DROP:** It is used to delete both the structure and record stored in the table.

**Syntax**

DROP TABLE table\_name;

**Example**

DROP TABLE EMPLOYEE;

**c. ALTER:**  It is used to alter the structure of the database. This change could be either to modify the characteristics of an existing attribute or probably to add a new attribute.

**Syntax:**

To add a new column in the table

ALTER TABLE table\_name ADD column\_name COLUMN-definition;

To modify existing column in the table:

ALTER TABLE table\_name MODIFY (column\_definitions....);

**EXAMPLE**

ALTER TABLE STU\_DETAILS ADD (ADDRESS VARCHAR2(20));

ALTER TABLE STU\_DETAILS MODIFY (NAME VARCHAR2(20));

TRUNCATE: It is used to delete all the rows from the table and free the space containing the table.

**Syntax:**

TRUNCATE TABLE table\_name;

**Example:**

TRUNCATE TABLE EMPLOYEE;

### 2. Data Manipulation Language

* DML commands are used to modify the database. It is responsible for all form of changes in the database.
* The command of DML is not auto-committed that means it can't permanently save all the changes in the database. They can be rollback.

Here are some commands that come under DML:

INSERT

UPDATE

DELETE

**a. INSERT:** The INSERT statement is a SQL query. It is used to insert data into the row of a table.

**Syntax:**

INSERT INTO TABLE\_NAME

(col1, col2, col3, .... col N)

VALUES (value1, value2, value3, .... valueN);

Or

INSERT INTO TABLE\_NAME

VALUES (value1, value2, value3, .... valueN);

**For example:**

INSERT INTO javatpoint (Author, Subject) VALUES ("Sonal", "DBMS");

**b. UPDATE:** This command is used to update or modify the value of a column in the table.

**Syntax:**

UPDATE table\_name SET [column\_name1= value1, ...column\_nameN = valueN] [WHERE CONDITION]

**For example:**

UPDATE students

SET User\_Name = 'Sonal'

WHERE Student\_Id = '3'

**c. DELETE:** It is used to remove one or more row from a table.

**Syntax:**

DELETE FROM table\_name [WHERE condition];

### 3. Data Control Language

DCL commands are used to grant and take back authority from any database user.

Here are some commands that come under DCL:

* Grant
* Revoke

**a. Grant:** It is used to give user access privileges to a database.

**Example**

GRANT SELECT, UPDATE ON MY\_TABLE TO SOME\_USER, ANOTHER\_USER;

**b. Revoke:** It is used to take back permissions from the user.

**Example**

REVOKE SELECT, UPDATE ON MY\_TABLE FROM USER1, USER2;

**4.Transaction Control Language**

TCL commands can only use with DML commands like INSERT, DELETE and UPDATE only.

These operations are automatically committed in the database that's why they cannot be used while creating tables or dropping them.

Here are some commands that come under TCL:

* COMMIT
* ROLLBACK
* SAVEPOINT

**a. Commit:** Commit command is used to save all the transactions to the database.

**Syntax:**

COMMIT;

**Example:**

DELETE FROM CUSTOMERS

WHERE AGE = 25;

COMMIT;

**b. Rollback:** Rollback command is used to undo transactions that have not already been saved to the database.

**Syntax:**

ROLLBACK;

**Example:**

DELETE FROM CUSTOMERS

WHERE AGE = 25;

ROLLBACK;

**c. SAVEPOINT:** It is used to roll the transaction back to a certain point without rolling back the entire transaction.

**Syntax:**

SAVEPOINT SAVEPOINT\_NAME;

### **5. Data Query Language**

DQL is used to fetch the data from the database.

It uses only one command:

* SELECT
  1. **SELECT:** This is the same as the projection operation of relational algebra. It is used to select the attribute based on the condition described by WHERE clause.

**What is a Primary Key**

A Primary Key is the minimal set of attributes of a table that has the task to uniquely identify the rows, or we can say the tuples of the given particular table.

A primary key of a relation is one of the possible candidate keys which the database designer thinks it's primary. It may be selected for convenience, performance and many other reasons.

### **Syntax for creating primary key constraint:**

**The primary key constraint can be defined at the column level or table level.**

**At column level**

<column\_name><datatype> Primary key;

**At table level**

Primary key(<column\_name1>[column\_name>] ....);

**Example**

CREATE TABLE STUDENT\_DETAIL (Roll\_no int NOT NULL PRIMARY KEY, Name varchar (20), Marks int);

## **The SQL MIN() and MAX() Functions:**

The MIN() function returns the smallest value of the selected column.

The MAX() function returns the largest value of the selected column.

SELECT MIN(*column\_name*)  
FROM *table\_name*  
WHERE *condition*;

SELECT MAX(*column\_name*)  
FROM *table\_name*  
WHERE *condition*;

## The SQL COUNT(), AVG() and SUM() Functions:

### 

### **COUNT() Syntax:**

### SELECT COUNT (*column\_name*) FROM *table\_name* WHERE *condition*;

### **AVG () Syntax:**

### SELECT AVG (*column\_name*) FROM *table\_name* WHERE *condition*;

### **SUM() Syntax:**

### SELECT SUM (*column\_name*) FROM *table\_name* WHERE *condition*;

### **SQL JOIN**

A JOIN clause is used to combine rows from two or more tables, based on a related column between them.

Let's look at a selection from the "Orders" table:

## Different Types of SQL JOINs

Here are the different types of the JOINs in SQL:

* (INNER) JOIN: Returns records that have matching values in both tables
* LEFT (OUTER) JOIN: Returns all records from the left table, and the matched records from the right table
* RIGHT (OUTER) JOIN: Returns all records from the right table, and the matched records from the left table
* FULL (OUTER) JOIN: Returns all records when there is a match in either left or right table

## SQL INNER JOIN Keyword

The INNER JOIN keyword selects records that have matching values in both tables.

SELECT *column\_name(s)*  
FROM *table1*  
INNER JOIN *table2*ON *table1.column\_name*=*table2.column\_name*;

## SQL LEFT JOIN Keyword

The LEFT JOIN keyword returns all records from the left table (table1), and the matching records from the right table (table2). The result is 0 records from the right side, if there is no match.

### 

### **LEFT JOIN Syntax**

SELECT *column\_name(s)*  
FROM *table1*  
LEFT JOIN *table2*ON *table1.column\_name*=*table2.column\_name*;

## SQL RIGHT JOIN Keyword

The RIGHT JOIN keyword returns all records from the right table (table2), and the matching records from the left table (table1). The result is 0 records from the left side, if there is no match.

### **RIGHT JOIN Syntax**

SELECT *column\_name(s)*  
FROM *table1*  
RIGHT JOIN *table2*ON *table1.column\_name*=*table2.column\_name*

## SQL FULL OUTER JOIN Keyword

The FULL OUTER JOIN keyword returns all records when there is a match in left (table1) or right (table2) table records.

**Tip:** FULL OUTER JOIN and FULL JOIN are the same.

### **FULL OUTER JOIN Syntax**

SELECT *column\_name(s)*  
FROM *table1*  
FULL OUTER JOIN *table2*ON *table1.column\_name*=*table2.column\_name*WHERE *condition*;

## **The SQL BETWEEN Operator**

The BETWEEN operator selects values within a given range. The values can be numbers, text, or dates.

The BETWEEN operator is inclusive: begin and end values are included.

### **BETWEEN Syntax**

SELECT *column\_name(s)*  
FROM *table\_name*  
WHERE *column\_name*BETWEEN *value1* AND *value2;*

select\*from employee;

+--------+----------+------------+----------+

| emp\_id | emp\_name | emp\_salary | emp\_city |

+--------+----------+------------+----------+

|      1 | Sonal   |       50000 | Pune |

|      2 | Sayali |       30000 | Mumbai     |

|      3 | Ratika   |       20000 | Delhi  |

+--------+----------+------------+----------+

3 rows in set (0.00 sec)

mysql> select sum(emp\_salary),emp\_city from employee group by emp\_city;

+-----------------+----------+

| sum(emp\_salary) | emp\_city |

+-----------------+----------+

|            50000 | Pune  |

|            30000 | Mumbai     |

|            20000 | Delhi  |

+-----------------+----------+

3 rows in set (0.07 sec)

mysql> insert into employee values( 4,'Kajal',35000,'Nashik'),(5,'Neha',32000,'Latur'),(6,'Tanuja',4000,'Nagpur');

Query OK, 3 rows affected (0.10 sec)

Records: 3  Duplicates: 0  Warnings: 0

mysql> select\*from employee;

+--------+----------+------------+----------+

| emp\_id | emp\_name | emp\_salary | emp\_city |

+--------+----------+------------+----------+

|      1 | Sonal  |       50000 | Pune |

|      2 | Sayali   |       30000 | Mumbai     |

|      3 | Ratika   |       20000 | Delhi   |

|      4 | Kajal    |       35000 | Nashik  |

|      5 | Neha  |       32000 |Latur    |

|      6 | Tanuja   |       40000 | Nagpur  |

+--------+----------+------------+----------+

6 rows in set (0.00 sec)

mysql> select sum(emp\_salary),emp\_city from employee group by emp\_city;

|       50000 | Pune |

|       30000 | Mumbai     |

|       20000 | Delhi   |

|       35000 | Nashik  |

|       32000 |Latur    |

|       40000 | Nagpur  |

6 rows in set (0.00 sec)

mysql> select sum(emp\_salary),emp\_city from employee group by emp\_city having sum(emp\_salary)>40000;

+-----------------+----------+

| sum(emp\_salary) | emp\_city |

+-----------------+----------+

|            50000 | pune     |

+-----------------+----------+

1 row in set (0.17 sec)

mysql> insert into employee values( 7,'Roshni',20000,'Pune'),(8,'Priya',25000,'delhi'),(9,'Pratiksha',30500,'Kolhapur');

Query OK, 2 rows affected (0.06 sec)

Records: 2  Duplicates: 0  Warnings: 0

mysql> select\*from employee;

+--------+----------+------------+----------+

| emp\_id | emp\_name | emp\_salary | emp\_city |

+--------+----------+------------+----------+

|      1 | Sonal  |       50000 | Pune |

|      2 | Sayali   |       30000 | Mumbai     |

|      3 | Ratika   |       20000 | Delhi   |

|      4 | Kajal    |       35000 | Nashik  |

|      5 | Neha  |       32000 |Latur    |

|      6 | Tanuja   |       40000 | Nagpur  |

|      7 | Roshni    |       20000 | Pune  |

|      8 | Priya |       25000 |  Delhi |

|      9 | Pratiksha   |       30500 | Kolhapur  |

+--------+----------+------------+----------+

9 rows in set (0.00 sec)

mysql> select sum(emp\_salary),emp\_city from employee1 group by emp\_city having sum(emp\_salary)>30000;

+-----------------+----------+

| sum(emp\_salary) | emp\_city |

+-----------------+----------+

|       50000 | Pune |

|       35000 | Nashik  |

|       32000 |Latur    |

|       40000 | Nagpur  |

|       30500 | Kolhapur  |

+-----------------+----------+

5 rows in set (0.00 sec)

mysql> use Sonal;

Database changed

mysql> create table stud(name varchar(20),id int,age int,address varchar(30));

Query OK, 0 rows affected (0.44 sec)

mysql> insert into stud values('Sonal',1, 11,'Pune'),('Sayali',2,12,'Mumbai');

Query OK, 2 rows affected (0.11 sec)

Records: 2  Duplicates: 0  Warnings: 0

mysql> insert into stud values('Ratika',3,13,'Delhi'),('Kajal',4,14,'Nashik');

Query OK, 2 rows affected (0.05 sec)

Records: 2  Duplicates: 0  Warnings: 0

mysql> select\*from stud;

+--------+------+------+---------+

| name   | id   | age  | address |

+--------+------+------+---------+

| Sonal |    1 |   11| Pune|

| Sayali  |    2 |   12 | Mumbai  |

| Ratika|    3 |   13 | Delhi    |

| Kajal |    4 |   14| Nashik  |

+--------+------+------+---------+

4 rows in set (0.00 sec)

mysql> select count(age)as countage from stud;

+----------+

| countage |

+----------+

|        4 |

+----------+

1 row in set (0.08 sec)

mysql> select\*from stud;

+--------+------+------+---------+

| name   | id   | age  | address |

| Sonal |    1 |   11| Pune|

| Sayali  |    2 |   12 | Mumbai  |

| Ratika|    3 |   13 | Delhi    |

| Kajal |    4 |   14| Nashik  |

+--------+------+------+---------+

4 rows in set (0.00 sec)

mysql> select \*from stud limit 2;

+--------+------+------+---------+

| name   | id   | age  | address |

+--------+------+------+---------+

| Sonal |    1 |   11 | Pune |

| Sayali  |    2 |   12 | Mumbai    |

+--------+------+------+---------+

2 rows in set (0.00 sec)

mysql> select \*from stud limit 3;

+--------+------+------+---------+

| name   | id   | age  | address |

+--------+------+------+---------+

| Sonal |    1 |   11| Pune|

| Sayali  |    2 |   12 | Mumbai  |

| Ratika|    3 |   13 | Delhi    |

+--------+------+------+---------+

3 rows in set (0.00 sec)

mysql> select \*from stud;

+--------+------+------+---------+

| name   | id   | age  | address |

+--------+------+------+---------+

| Sonal |    1 |   11| Pune|

| Sayali  |    2 |   12 | Mumbai  |

| Ratika|    3 |   13 | Delhi    |

| Kajal |    4 |   14| Nashik  |

+--------+------+------+---------+

4 rows in set (0.00 sec)

mysql> select distinct age from stud;

+------+

| age  |

+------+

|   11|

|   12|

|   13 |

+------+

3 rows in set (0.11 sec)

mysql> select distinct id from stud;

+------+

| id   |

+------+

|    1 |

|    2 |

|    3 |

|    4 |

+------+

4 rows in set (0.00 sec)

mysql> select count(age) as totalage from stud;

+----------+

| totalage |

+----------+

|        4 |

+----------+

1 row in set (0.00 sec)

mysql> select count(\*)from stud;

+----------+

| count(\*) |

+----------+

|        4 |

+----------+

1 row in set (0.29 sec)

mysql> select count(distinct age)as totalage from stud;

+----------+

| totalage |

+----------+

|        3 |

+----------+

1 row in set (0.02 sec)

mysql> select count( age)as totalage from stud where name='Sonal';

+----------+

| totalage |

+----------+

|        1 |

+----------+

1 row in set (0.09 sec)

mysql>

mysql> select\*from stud;

+--------+------+------+---------+

| name   | id   | age  | address |

+--------+------+------+---------+

| Sonal |    1 |   11| Pune|

| Sayali  |    2 |   12 | Mumbai  |

| Ratika|    3 |   13 | Delhi    |

| Kajal |    4 |   14| Nashik  |

+--------+------+------+---------+

4 rows in set (0.00 sec)

mysql> select age from stud order by age desc limit 1;

+------+

| age  |

+------+

|   14 |

+------+

1 row in set (0.09 sec)

mysql> select name from stud order by name desc limit 1;

+--------+

| name   |

+--------+

| Kajal|

+--------+

1 row in set (0.02 sec)

mysql> select name from stud order by name asc limit 1;

+--------+

| name   |

+--------+

| Sonal |

+--------+

1 row in set (0.00 sec)

mysql> select age from stud order by age asc limit 1;

+------+

| age  |

+------+

|   11 |

+------+

1 row in set (0.00 sec)

mysql> select sum(age) as totalage from stud where age>20;

+----------+

| totalage |

+----------+

|       50 |

+----------+

1 row in set (0.03 sec)

mysql> select sum(age) as totalage from stud where age>11;

+----------+

| totalage |

+----------+

|       39 |

+----------+

1 row in set (0.00 sec)

mysql> select\* from stud where age=11 and name='Sonal';

+--------+------+------+---------+

| name   | id   | age  | address |

+--------+------+------+---------+

| Sonal |    1 |   11 | Pune |

+--------+------+------+---------+

1 row in set (0.04 sec)

mysql> select\* from stud where age=11 or name='Sonal';

+--------+------+------+---------+

| name   | id   | age  | address |

+--------+------+------+---------+

| Sonal |    1 |   11 | Pune |

+--------+------+------+---------+

1 row in set (0.00 sec)

mysql> select\* from stud where not  name='Sonal';

+--------+------+------+---------+

| name   | id   | age  | address |

+--------+------+------+---------+

| Sayali  |    2 |   12 | Mumbai  |

| Ratika|    3 |   13 | Delhi    |

| Kajal |    4 |   14| Nashik  |

+--------+------+------+---------+

3 rows in set (0.02 sec)