**DBMS SQL QUERIES**

**NAME: M.ABITHA**

**BATCH: 2022-7607**

**ENROLMENT NUMBER: EBEON0722634453**

**LOCATION: Chennai**

# **SQL Commands in DBMS**

**RDBMS:**

RDBMS stands for Relational Database Management System.

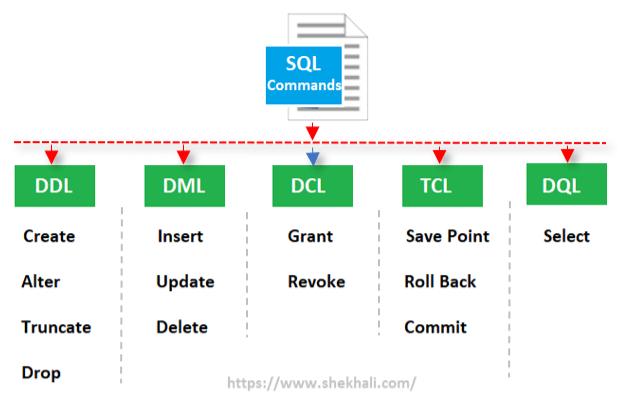
RDBMS is the basis for SQL, and for all modern database systems such as MS SQL Server, IBM DB2, Oracle, MySQL, and Microsoft Access.

The data in RDBMS is stored in database objects called tables. A table is a collection of related data entries and it consists of columns and rows.

**SQL :**

SQL is a standard language for accessing and manipulating databases.

* SQL stands for Structured Query Language
* SQL lets you access and manipulate databases
* SQL became a standard of the American National Standards Institute (ANSI) in 1986, and of the International Organization for Standardization (ISO) in 1987

.Some of The Most Important SQL Commands:

* SELECT – extracts data from a database
* UPDATE – updates data in a database
* DELETE – deletes data from a database
* INSERT INTO – inserts new data into a database
* CREATE DATABASE – creates a new database
* ALTER DATABASE – modifies a database
* CREATE TABLE – creates a new table
* ALTER TABLE – modifies a table
* DROP TABLE – deletes a table
* CREATE INDEX – creates an index (search key)
* DROP INDEX – deletes an index.

**SQL commends**

**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

**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);

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

**Syntax:**

* DROP TABLE table\_name;

**Example:**

* DROP TABLE EMPLOYEE;

**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.

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.

INSERT

UPDATE

DELETE

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

**For example:**

INSERT INTO Employee (Emp\_name, Subject) VALUES (“Sonoo”, “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 = ‘Sonoo’

WHERE Student\_Id = ‘3’ ;

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

**Syntax:**

DELETE FROM table\_name [WHERE condition];

**For example:**

1. DELETE FROM table\_name
2. WHERE Author="Sonoo";

**3.Data Control Language:**

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

* **Grant**
* **Revoke**

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

**Example**

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

**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.
* **Commit**
* **Rollback**
* **savepoint**

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

**Syntax:**

COMMIT;

**Example:**

DELETE FROM CUSTOMERS

WHERE AGE = 25;

COMMIT;

**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;

**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

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.

SELECT Syntax

SELECT column1, column2, …

FROM table\_name;

**Or**

SELECT expressions

FROM TABLES

WHERE conditions;

If you want to select all the fields available in the table, use the following:

**For**

**Example:**

**SELECT\* Employee**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | EMP\_NAME | CITY | E\_NO | SALARY |
|  | Abi | Chicago | 9737287378 | 150000 |
|  | Anuma | Austin | 9262738271 | 200000 |
|  | Angel | Denver | 9232673822 | 600000 |
|  | Roshni | Washington | 9367238263 | 350000 |
|  | Chritial | Los angels | 7253847382 | 260000 |

**Select statement:**

The SELECT statement is used to select data from a database.

**SQL Queries and the Select Sentence :**

**Output:**

If we select particular column only from the table means it will show only selected column:

SELECT EMP\_ID FROM EMPLOYEE;

|  |
| --- |
| **EMP\_ID** |
| 1 |
| 2 |
| 3 |
| 4 |
| 5 |

To fetch the EMP\_NAME and SALARY, use the following query:

SELECT EMP\_NAME, SALARY FROM EMPLOYEE;

|  |  |
| --- | --- |
| **EMP\_NAME** | **SALARY** |
| Kristen | 150000 |
| Russell | 200000 |
| Angelina | 600000 |
| Robert | 350000 |
| Christian | 260000 |

To fetch all the fields from the EMPLOYEE table, use the following query:

SELECT \* FROM EMPLOYEE

Output

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **EMP\_ID** | **EMP\_NAME** | **CITY** | **PHONE\_NO** | **SALARY** |
| 1 | Kristen | Chicago | 9737287378 | 150000 |
| 2 | Russell | Austin | 9262738271 | 200000 |
| 3 | Angelina | Denver | 9232673822 | 600000 |
| 4 | Robert | Washington | 9367238263 | 350000 |
| 5 | Christian | Los angels | 7253847382 | 260000 |

**SQL INSERT Statement:**

The SQL INSERT statement is used to insert a single or multiple data in a table. In SQL, You can insert the data in two ways:

* Without specifying column name
* By specifying column name

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| * **EMP\_ID** | **EMP\_NAME** | **CITY** | **SALARY** | **AGE** |
| 1 | Angelina | Chicago | 200000 | 30 |
| 2 | Robert | Austin | 300000 | 26 |
| 3 | Christian | Denver | 100000 | 42 |
| 4 | Kristen | Washington | 500000 | 29 |
| 5 | Russell | Los angels | 200000 | 36 |

### **Without Specified name:**

**Syntax:**

INSERT INTO TABLE\_NAME

VALUES (value1, value2, value 3, .... Value N);

**Query:**

INSERT INTO EMPLOYEE VALUES (6, 'Marry', 'Canada', 600000, 48); Output: After executing this query, the EMPLOYEE table will look like:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **EMP\_ID** | **EMP\_NAME** | **CITY** | **SALARY** | **AGE** |
| 1 | Angel | Chicago | 200000 | 30 |
| 2 | Roshni | Austin | 300000 | 26 |
| 3 | Chritial | Denver | 100000 | 42 |
| 4 | Anuma | Washington | 500000 | 29 |
| 5 | Abi | Los angels | 200000 | 36 |
| 6 | Meha | Canada | 600000 | 48 |

To insert **partial column values**, you must have to specify the column names.

**Syntax:**

1. INSERT INTO TABLE\_NAME
2. [(col1, col2, col3,.... col N)]
3. VALUES (value1, value2, value 3, .... Value N);

**Query:**

INSERT INTO EMPLOYEE (EMP\_ID, EMP\_NAME, AGE) VALUES (7, ‘Jack’, 40);

After executing this query, the table will look like:

**Output:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **EMP\_ID** | **EMP\_NAME** | **CITY** | **SALARY** | **AGE** |
| 1 | Angelina | Chicago | 200000 | 30 |
| 2 | Roshni | Austin | 300000 | 26 |
| 3 | Chritial | Denver | 100000 | 42 |
| 4 | Anuma | Washington | 500000 | 29 |
| 5 | Abi | Los angels | 200000 | 36 |
| 6 | Meha | Canada | 600000 | 48 |
| 7 | Jack | null | null | 40 |

**SQL Update Statement:**

Syntax

UPDATE table\_name

SET column1 = value1, column2 = value2, …

WHERE condition;

**Table Name**

**EMPLOYEE:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **EMP\_ID** | **EMP\_NAME** | **CITY** | **SALARY** | **AGE** |
| 1 | Angel | Chicago | 200000 | 30 |
| 2 | Roshni | Austin | 300000 | 26 |
| 3 | Chritial | Denver | 100000 | 42 |
| 4 | Anuma | Washington | 500000 | 29 |
| 5 | Abi | Los angels | 200000 | 36 |
| 6 | Meha | Canada | 600000 | 48 |

Update the column EMP\_NAME and set the value to 'Emma' in the row where SALARY is 500000.

Syntax

UPDATE table\_name

SET column\_name = value

WHERE condition;

**Query:**

UPDATE EMPLOYEE

SET EMP\_NAME = 'Anuma'

WHERE SALARY = 500000;

**Output:** After executing this query, the EMPLOYEE table will look like:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **EMP\_ID** | **EMP\_NAME** | **CITY** | **SALARY** | **AGE** |
| 1 | Angel | Chicago | 200000 | 30 |
| 2 | Roshni | Austin | 300000 | 26 |
| 3 | Chritial | Denver | 100000 | 42 |
| 4 | Anuma | Washington | 500000 | 29 |
| 5 | Abi | Los angels | 200000 | 36 |
| 6 | Meha | Canada | 600000 | 48 |

If you want to update multiple columns, you should separate each field assigned with a comma.

Like this:

**Query:**

UPDATE EMPLOYEE

SET EMP\_NAME = ‘Kevin’, City = ‘India’

WHERE EMP\_ID = 5;

Output

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **EMP\_ID** | **EMP\_NAME** | **CITY** | **SALARY** | **AGE** |
| 1 | Angel | Chicago | 200000 | 30 |
| 2 | Roshni | Austin | 300000 | 26 |
| 3 | Chritial | Denver | 100000 | 42 |
| 4 | Anuma | Washington | 500000 | 29 |
| 5 | Kevin | India | 200000 | 36 |
| 6 | Meha | Canada | 600000 | 48 |

**SQL DELETE Statement:**

The SQL DELETE statement is used to delete rows from a table. Generally, DELETE statement removes one or more records form a table.

**Syntax:**

DELETE FROM table\_name WHERE some\_condition;

DELETE FROM EMPLOYEE

WHERE EMP\_NAME = ‘Kristen’;

Output: After executing this query, the EMPLOYEE table will look like:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **EMP\_ID** | **EMP\_NAME** | **CITY** | **SALARY** | **AGE** |
| 1 | Angel | Chicago | 200000 | 30 |
| 2 | Roshni | Austin | 300000 | 26 |
| 3 | Chritial | Denver | 100000 | 42 |
| 5 | Abi | Los angels | 200000 | 36 |
| 6 | Meha | Canada | 600000 | 48 |

**Deleting Multiple Record:**

**Query:**

DELETE FROM EMPLOYEE WHERE AGE= 30;

Output: After executing this query, the EMPLOYEE table will look like:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **EMP\_ID** | **EMP\_NAME** | **CITY** | **SALARY** | **AGE** |
| 2 | Roshni | Austin | 300000 | 26 |
| 3 | Chritial | Denver | 100000 | 42 |
| 5 | Abi | Los angels | 200000 | 36 |
| 6 | Meha | Canada | 600000 | 48 |

Delete all the row from the EMPLOYEE table. After this, no records left to display. The EMPLOYEE table will become empty.

**Query :**

DELETE FROM EMPLOYEE;

Output: After executing this query, the EMPLOYEE table will look like:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **EMP\_ID** | **EMP\_NAME** | **CITY** | **SALARY** | **AGE** |

**1.SELECT DISTINCT:**

The select distinct statement is used to return only distinct diff values.

**Syntax:**

SELECT DISTINCT column1,column2,…

FROM table\_name;

For example

Select distinct country from customers;

**2.WHERE Clause :**

The where clause is used to filter records.

**Syntax:**

SELECT column1,column2,..

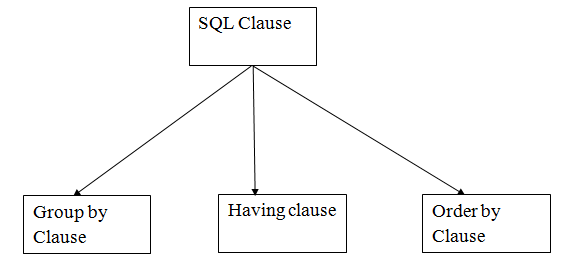
FROM table\_name

WHERE condition;

**For example :**

Select\*from customers where customer\_id=1;

**SQL Clauses:**

**. GROUP BY**

SQL GROUP BY statement is used to arrange identical data into groups. The GROUP BY statement is used with the SQL SELECT statement.

**Syntax :**

SELECT column

FROM table\_name

WHERE conditions

GROUP BY column

**Example:**

SELECT column\_name, COUNT(\*)

FROM table\_name

GROUP BY column\_name;

**HAVING**

HAVING clause is used to specify a search condition for a group or an aggregate.

**Syntax:**

SELECT column1, column2

FROM table\_name

WHERE conditions

GROUP BY column1, column2

HAVING conditions

ORDER BY column1, column2;

**Example:**

SELECT Column\_name, COUNT(\*)

FROM table\_name

GROUP BY column\_name

HAVING COUNT(\*)>2;

**3.ORDER BY:**

The order by is used to sort the result set in ascending or descending order.

**Syntax:**

Select column1,column2,…

From table\_name

Order by column1,column2,…

Asc or desc;

**ASC:** It is used to sort the result set in ascending order by expression.

**DESC:** It sorts the result set in descending order by expression.

**For example:**

Select \*from customers order by country desc;

**CUSTOMER:**

|  |  |  |
| --- | --- | --- |
| **CUSTOMER\_ID** | **NAME** | **ADDRESS** |
| 12 | Kathir | India |
| 23 | Divya | India |
| 34 | Abi | Dubai |
| 45 | John | UK |
| 56 | Harry | US |

Enter the following SQL statement:

SELECT \*  FROM CUSTOMER ORDER BY NAME;

**Output:**

|  |  |  |
| --- | --- | --- |
| **CUSTOMER\_ID** | **NAME** | **ADDRESS** |
| 34 | Abi | Dubai |
| 23 | Divya | India |
| 56 | Harry | US |
| 45 | John | UK |
| 12 | Kathir | India |

**Descending Order:**

Using the above CUSTOMER table

* SELECT \*
* FROM CUSTOMER
* ORDER BY NAME DESC;

**Output:**

|  |  |  |
| --- | --- | --- |
| **CUSTOMER\_ID** | **NAME** | **ADDRESS** |
| 12 | Kathir | India |
| 45 | John | UK |
| 56 | Harry | US |
| 23 | Divya | India |
| 34 | Abi | Dubai |

### **COUNT:**

COUNT function uses the COUNT(\*) that returns the count of all the rows in a specified table. COUNT(\*) considers duplicate and Null.

**Syntax:**

COUNT(\*)

Or

COUNT( [ALL|DISTINCT] expression )

**COUNT with WHERE:**

SELECT COUNT(\*)

FROM table\_name;

WHERE RATE>=20;

**Example: COUNT() with DISTINCT**

SELECT COUNT(DISTINCT Column\_name)

FROM table\_name;

**Example: COUNT() with GROUP BY:**

SELECT column\_name, COUNT(\*)

FROM table \_name

GROUP BY Column\_name;

**Example: COUNT() with HAVING:**

SELECT Column\_name, COUNT(\*)

FROM table \_name

GROUP BY Column\_name

HAVING COUNT(\*)>2;

**SUM Function:**

Sum function is used to calculate the sum of all selected columns. It works on numeric fields only.

**Syntax:**

SUM()

Or

SUM( [ALL|DISTINCT] expression )

**Example: SUM()**

SELECT SUM(salary)

FROM employee;

**Example: SUM() with WHERE**

* SELECT SUM(COST)
* FROM PRODUCT\_MAST
* WHERE QTY>3;

**Example: SUM() with GROUP BY:**

SELECT SUM(COST)

FROM table \_name

WHERE QTY>3

GROUP BY column\_name;

**SQL JOIN:**

As the name shows, JOIN means to combine something. In case of SQL, JOIN means “to combine two or more tables”.

**Types of SQL JOIN**

* INNER JOIN
* LEFT JOIN
* RIGHT JOIN
* FULL JOIN
* **INNER JOIN:**

In SQL, INNER JOIN selects records that have matching values in both tables as long as the condition is satisfied. It returns the combination of all rows from both the tables where the condition satisfies.

**Syntax:**

SELECT table1.column1, table1.column2, table2.column1,….

FROM table1

INNER JOIN table2

ON table1.matching\_column = table2.matching\_column;

* **LEFT JOIN:**

The SQL left join returns all the values from left table and the matching values from the right table. If there is no matching join value, it will return NULL.

**Syntax:**

SELECT table1.column1, table1.column2, table2.column1,….

FROM table1

LEFT JOIN table2

ON table1.matching\_column = table2.matching\_column;

* **RIGHT JOIN:**

In SQL, RIGHT JOIN returns all the values from the values from the rows of right table and the matched values from the left table. If there is no matching in both tables, it will return NULL.

**Syntax:**

SELECT table1.column1, table1.column2, table2.column1,….

FROM table1

RIGHT JOIN table2

ON table1.matching\_column = table2.matching\_column;

* **FULL JOIN:**

In SQL, FULL JOIN is the result of a combination of both left and right outer join. Join tables have all the records from both tables. It puts NULL on the place of matches not found.

**Syntax:**

SELECT table1.column1, table1.column2, table2.column1,….

FROM table1

FULL JOIN table2

ON table1.matching\_column = table2.matching\_column;

**Union:**

The SQL Union operation is used to combine the result of two or more SQL SELECT queries.

**Syntax:**

SELECT column\_name FROM table1

UNION

SELECT column\_name FROM table2;

**Conclusion**

So far discussed all the five categories of SQL Commands DCL, DML, DDL, TCL and DQL and it’s subtypes. We’ve gone through each command in detail with its syntax and example that will assist you in writing queries. The SQL commands’ allows you to construct and manipulate a wide range of database objects with the different commands. After going through this blog one can easily play with the SQL commands and interact with their database easily.