# SQL - WHERE Clause

The SQL **WHERE** clause is used to specify a condition while fetching the data from a single table or by joining with multiple tables. If the given condition is satisfied, then only it returns a specific value from the table. You should use the WHERE clause to filter the records and fetching only the necessary records.

The WHERE clause is not only used in the SELECT statement, but it is also used in the UPDATE, DELETE statement, etc., which we would examine in the subsequent chapters.

## Syntax

The basic syntax of the SELECT statement with the WHERE clause is as shown below.

SELECT column1, column2, columnN

FROM table\_name

WHERE [condition]

You can specify a condition using the [comparison or logical operators](https://www.tutorialspoint.com/sql/sql-operators.htm) like >, <, =, **LIKE, NOT**, etc. The following examples would make this concept clear.

## Example

Consider the CUSTOMERS table having the following records −

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

| ID | NAME | AGE | ADDRESS | SALARY |

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

| 1 | Ramesh | 32 | Ahmedabad | 2000.00 |

| 2 | Khilan | 25 | Delhi | 1500.00 |

| 3 | kaushik | 23 | Kota | 2000.00 |

| 4 | Chaitali | 25 | Mumbai | 6500.00 |

| 5 | Hardik | 27 | Bhopal | 8500.00 |

| 6 | Komal | 22 | MP | 4500.00 |

| 7 | Muffy | 24 | Indore | 10000.00 |

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

The following code is an example which would fetch the ID, Name and Salary fields from the CUSTOMERS table, where the salary is greater than 2000 −

SQL> SELECT ID, NAME, SALARY

FROM CUSTOMERS

WHERE SALARY > 2000;

This would produce the following result −

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

| ID | NAME | SALARY |

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

| 4 | Chaitali | 6500.00 |

| 5 | Hardik | 8500.00 |

| 6 | Komal | 4500.00 |

| 7 | Muffy | 10000.00 |

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

The following query is an example, which would fetch the ID, Name and Salary fields from the CUSTOMERS table for a customer with the name **Hardik**.

Here, it is important to note that all the strings should be given inside single quotes (''). Whereas, numeric values should be given without any quote as in the above example.

SQL> SELECT ID, NAME, SALARY

FROM CUSTOMERS

WHERE NAME = 'Hardik';

This would produce the following result −

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

| ID | NAME | SALARY |

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

| 5 | Hardik | 8500.00 |

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

# SQL - Distinct Keyword

The SQL **DISTINCT** keyword is used in conjunction with the SELECT statement to eliminate all the duplicate records and fetching only unique records002E

There may be a situation when you have multiple duplicate records in a table. While fetching such records, it makes more sense to fetch only those unique records instead of fetching duplicate records.

## Syntax

The basic syntax of DISTINCT keyword to eliminate the duplicate records is as follows −

SELECT DISTINCT column1, column2,.....columnN

FROM table\_name

WHERE [condition]

## Example

Consider the CUSTOMERS table having the following records −

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

| ID | NAME | AGE | ADDRESS | SALARY |

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

| 1 | Ramesh | 32 | Ahmedabad | 2000.00 |

| 2 | Khilan | 25 | Delhi | 1500.00 |

| 3 | kaushik | 23 | Kota | 2000.00 |

| 4 | Chaitali | 25 | Mumbai | 6500.00 |

| 5 | Hardik | 27 | Bhopal | 8500.00 |

| 6 | Komal | 22 | MP | 4500.00 |

| 7 | Muffy | 24 | Indore | 10000.00 |

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

First, let us see how the following SELECT query returns the duplicate salary records.

SQL> SELECT SALARY FROM CUSTOMERS

ORDER BY SALARY;

This would produce the following result, where the salary (2000) is coming twice which is a duplicate record from the original table.

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

| SALARY |

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

| 1500.00 |

| 2000.00 |

| 2000.00 |

| 4500.00 |

| 6500.00 |

| 8500.00 |

| 10000.00 |

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

Now, let us use the DISTINCT keyword with the above SELECT query and then see the result.

SQL> SELECT DISTINCT SALARY FROM CUSTOMERS

ORDER BY SALARY;

This would produce the following result where we do not have any duplicate entry.

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

| SALARY |

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

| 1500.00 |

| 2000.00 |

| 4500.00 |

| 6500.00 |

| 8500.00 |

| 10000.00 |

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

# SQL - ORDER BY Clause

The SQL **ORDER BY** clause is used to sort the data in ascending or descending order, based on one or more columns. Some databases sort the query results in an ascending order by default.

## Syntax

The basic syntax of the ORDER BY clause is as follows −

SELECT column-list

FROM table\_name

[WHERE condition]

[ORDER BY column1, column2, .. columnN] [ASC | DESC];

You can use more than one column in the ORDER BY clause. Make sure whatever column you are using to sort that column should be in the column-list.

## Example

Consider the CUSTOMERS table having the following records −

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

| ID | NAME | AGE | ADDRESS | SALARY |

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

| 1 | Ramesh | 32 | Ahmedabad | 2000.00 |

| 2 | Khilan | 25 | Delhi | 1500.00 |

| 3 | kaushik | 23 | Kota | 2000.00 |

| 4 | Chaitali | 25 | Mumbai | 6500.00 |

| 5 | Hardik | 27 | Bhopal | 8500.00 |

| 6 | Komal | 22 | MP | 4500.00 |

| 7 | Muffy | 24 | Indore | 10000.00 |

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

The following code block has an example, which would sort the result in an ascending order by the NAME and the SALARY −

SQL> SELECT \* FROM CUSTOMERS

ORDER BY NAME, SALARY;

This would produce the following result −

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

| ID | NAME | AGE | ADDRESS | SALARY |

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

| 4 | Chaitali | 25 | Mumbai | 6500.00 |

| 5 | Hardik | 27 | Bhopal | 8500.00 |

| 3 | kaushik | 23 | Kota | 2000.00 |

| 2 | Khilan | 25 | Delhi | 1500.00 |

| 6 | Komal | 22 | MP | 4500.00 |

| 7 | Muffy | 24 | Indore | 10000.00 |

| 1 | Ramesh | 32 | Ahmedabad | 2000.00 |

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

The following code block has an example, which would sort the result in the descending order by NAME.

SQL> SELECT \* FROM CUSTOMERS

ORDER BY NAME DESC;

This would produce the following result −

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

| ID | NAME | AGE | ADDRESS | SALARY |

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

| 1 | Ramesh | 32 | Ahmedabad | 2000.00 |

| 7 | Muffy | 24 | Indore | 10000.00 |

| 6 | Komal | 22 | MP | 4500.00 |

| 2 | Khilan | 25 | Delhi | 1500.00 |

| 3 | kaushik | 23 | Kota | 2000.00 |

| 5 | Hardik | 27 | Bhopal | 8500.00 |

| 4 | Chaitali | 25 | Mumbai | 6500.00 |

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

# SQL - TOP, LIMIT or ROWNUM Clause

The SQL **TOP** clause is used to fetch a TOP N number or X percent records from a table.

**Note** − All the databases do not support the TOP clause. For example MySQL supports the **LIMIT** clause to fetch limited number of records while Oracle uses the **ROWNUM** command to fetch a limited number of records.

## Syntax

The basic syntax of the TOP clause with a SELECT statement would be as follows.

SELECT TOP number|percent column\_name(s)

FROM table\_name

WHERE [condition]

## Example

Consider the CUSTOMERS table having the following records −

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

| ID | NAME | AGE | ADDRESS | SALARY |

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

| 1 | Ramesh | 32 | Ahmedabad | 2000.00 |

| 2 | Khilan | 25 | Delhi | 1500.00 |

| 3 | kaushik | 23 | Kota | 2000.00 |

| 4 | Chaitali | 25 | Mumbai | 6500.00 |

| 5 | Hardik | 27 | Bhopal | 8500.00 |

| 6 | Komal | 22 | MP | 4500.00 |

| 7 | Muffy | 24 | Indore | 10000.00 |

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

The following query is an example on the SQL server, which would fetch the top 3 records from the CUSTOMERS table.

SQL> SELECT TOP 3 \* FROM CUSTOMERS;

This would produce the following result −

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

| ID | NAME | AGE | ADDRESS | SALARY |

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

| 1 | Ramesh | 32 | Ahmedabad | 2000.00 |

| 2 | Khilan | 25 | Delhi | 1500.00 |

| 3 | kaushik | 23 | Kota | 2000.00 |

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

If you are using MySQL server, then here is an equivalent example −

SQL> SELECT \* FROM CUSTOMERS

LIMIT 3;

This would produce the following result −

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

| ID | NAME | AGE | ADDRESS | SALARY |

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

| 1 | Ramesh | 32 | Ahmedabad | 2000.00 |

| 2 | Khilan | 25 | Delhi | 1500.00 |

| 3 | kaushik | 23 | Kota | 2000.00 |

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

If you are using an Oracle server, then the following code block has an equivalent example.

SQL> SELECT \* FROM CUSTOMERS

WHERE ROWNUM <= 3;

This would produce the following result −

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

| ID | NAME | AGE | ADDRESS | SALARY |

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

| 1 | Ramesh | 32 | Ahmedabad | 2000.00 |

| 2 | Khilan | 25 | Delhi | 1500.00 |

| 3 | kaushik | 23 | Kota | 2000.00 |

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

# SQL - Group By

The SQL **GROUP BY** clause is used in collaboration with the SELECT statement to arrange identical data into groups. This GROUP BY clause follows the WHERE clause in a SELECT statement and precedes the ORDER BY clause.

## Syntax

The basic syntax of a GROUP BY clause is shown in the following code block. The GROUP BY clause must follow the conditions in the WHERE clause and must precede the ORDER BY clause if one is used.

SELECT column1, column2

FROM table\_name

WHERE [ conditions ]

GROUP BY column1, column2

ORDER BY column1, column2

## Example

Consider the CUSTOMERS table is having the following records −

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

| ID | NAME | AGE | ADDRESS | SALARY |

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

| 1 | Ramesh | 32 | Ahmedabad | 2000.00 |

| 2 | Khilan | 25 | Delhi | 1500.00 |

| 3 | kaushik | 23 | Kota | 2000.00 |

| 4 | Chaitali | 25 | Mumbai | 6500.00 |

| 5 | Hardik | 27 | Bhopal | 8500.00 |

| 6 | Komal | 22 | MP | 4500.00 |

| 7 | Muffy | 24 | Indore | 10000.00 |

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

If you want to know the total amount of the salary on each customer, then the GROUP BY query would be as follows.

SQL> SELECT NAME, SUM(SALARY) FROM CUSTOMERS

GROUP BY NAME;

This would produce the following result −

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

| NAME | SUM(SALARY) |

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

| Chaitali | 6500.00 |

| Hardik | 8500.00 |

| kaushik | 2000.00 |

| Khilan | 1500.00 |

| Komal | 4500.00 |

| Muffy | 10000.00 |

| Ramesh | 2000.00 |

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

Now, let us look at a table where the CUSTOMERS table has the following records with duplicate names −

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

| ID | NAME | AGE | ADDRESS | SALARY |

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

| 1 | Ramesh | 32 | Ahmedabad | 2000.00 |

| 2 | Ramesh | 25 | Delhi | 1500.00 |

| 3 | kaushik | 23 | Kota | 2000.00 |

| 4 | kaushik | 25 | Mumbai | 6500.00 |

| 5 | Hardik | 27 | Bhopal | 8500.00 |

| 6 | Komal | 22 | MP | 4500.00 |

| 7 | Muffy | 24 | Indore | 10000.00 |

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

Now again, if you want to know the total amount of salary on each customer, then the GROUP BY query would be as follows −

SQL> SELECT NAME, SUM(SALARY) FROM CUSTOMERS

GROUP BY NAME;

This would produce the following result −

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

| NAME | SUM(SALARY) |

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

| Hardik | 8500.00 |

| kaushik | 8500.00 |

| Komal | 4500.00 |

| Muffy | 10000.00 |

| Ramesh | 3500.00 |

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

# SQL - Having Clause

The **HAVING Clause** enables you to specify conditions that filter which group results appear in the results.

The WHERE clause places conditions on the selected columns, whereas the HAVING clause places conditions on groups created by the GROUP BY clause.

## Syntax

The following code block shows the position of the HAVING Clause in a query.

SELECT

FROM

WHERE

GROUP BY

HAVING

ORDER BY

The HAVING clause must follow the GROUP BY clause in a query and must also precede the ORDER BY clause if used. The following code block has the syntax of the SELECT statement including the HAVING clause −

SELECT column1, column2

FROM table1, table2

WHERE [ conditions ]

GROUP BY column1, column2

HAVING [ conditions ]

ORDER BY column1, column2

## Example

Consider the CUSTOMERS table having the following records.

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

| ID | NAME | AGE | ADDRESS | SALARY |

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

| 1 | Ramesh | 32 | Ahmedabad | 2000.00 |

| 2 | Khilan | 25 | Delhi | 1500.00 |

| 3 | kaushik | 23 | Kota | 2000.00 |

| 4 | Chaitali | 25 | Mumbai | 6500.00 |

| 5 | Hardik | 27 | Bhopal | 8500.00 |

| 6 | Komal | 22 | MP | 4500.00 |

| 7 | Muffy | 24 | Indore | 10000.00 |

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

Following is an example, which would display a record for a similar age count that would be more than or equal to 2.

SQL > SELECT ID, NAME, AGE, ADDRESS, SALARY

FROM CUSTOMERS

GROUP BY age

HAVING COUNT(age) >= 2;

This would produce the following result −

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

| ID | NAME | AGE | ADDRESS | SALARY |

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

| 2 | Khilan | 25 | Delhi | 1500.00 |

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

SQL - Clone Tables

There may be a situation when you need an exact copy of a table and the CREATE TABLE ... or the SELECT... commands does not suit your purposes because the copy must include the same indexes, default values and so forth.

If you are using MySQL RDBMS, you can handle this situation by adhering to the steps given below −

* Use SHOW CREATE TABLE command to get a CREATE TABLE statement that specifies the source table's structure, indexes and all.
* Modify the statement to change the table name to that of the clone table and execute the statement. This way you will have an exact clone table.
* Optionally, if you need the table contents copied as well, issue an INSERT INTO or a SELECT statement too.

## Example

Try out the following example to create a clone table for **TUTORIALS\_TBL**whose structure is as follows −

**Step 1** − Get the complete structure about the table.

SQL> SHOW CREATE TABLE TUTORIALS\_TBL \G;

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Table: TUTORIALS\_TBL

Create Table: CREATE TABLE 'TUTORIALS\_TBL' (

'tutorial\_id' int(11) NOT NULL auto\_increment,

'tutorial\_title' varchar(100) NOT NULL default '',

'tutorial\_author' varchar(40) NOT NULL default '',

'submission\_date' date default NULL,

PRIMARY KEY ('tutorial\_id'),

UNIQUE KEY 'AUTHOR\_INDEX' ('tutorial\_author')

) TYPE = MyISAM

1 row in set (0.00 sec)

**Step 2** − Rename this table and create another table.

SQL> CREATE TABLE `CLONE\_TBL` (

-> 'tutorial\_id' int(11) NOT NULL auto\_increment,

-> 'tutorial\_title' varchar(100) NOT NULL default '',

-> 'tutorial\_author' varchar(40) NOT NULL default '',

-> 'submission\_date' date default NULL,

-> PRIMARY KEY (`tutorial\_id'),

-> UNIQUE KEY 'AUTHOR\_INDEX' ('tutorial\_author')

-> ) TYPE = MyISAM;

Query OK, 0 rows affected (1.80 sec)

**Step 3** − After executing step 2, you will clone a table in your database. If you want to copy data from an old table, then you can do it by using the INSERT INTO... SELECT statement.

SQL> INSERT INTO CLONE\_TBL (tutorial\_id,

-> tutorial\_title,

-> tutorial\_author,

-> submission\_date)

-> SELECT tutorial\_id,tutorial\_title,

-> tutorial\_author,submission\_date,

-> FROM TUTORIALS\_TBL;

Query OK, 3 rows affected (0.07 sec)

Records: 3 Duplicates: 0 Warnings: 0

Finally, you will have an exact clone table as you wanted to have.