

**Insert Operation:**

The SQL **INSERT INTO** Statement is used to add new rows of data to a table in the database.

There are two basic syntaxes of the INSERT INTO statement which are shown below.

INSERT INTO TABLE\_NAME (column1, column2, column3,...columnN)

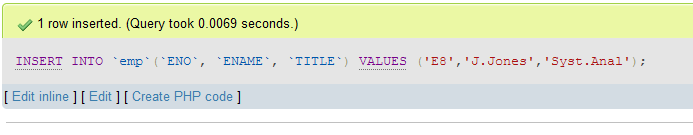
VALUES (value1, value2, value3,...valueN);

Here, column1, column2, column3,...columnN are the names of the columns in the table into which you want to insert the data.

You may not need to specify the column(s) name in the SQL query if you are adding values for all the columns of the table. But make sure the order of the values is in the same order as the columns in the table

**Example:**

INSERT INTO `emp`(`ENO`, `ENAME`, `TITLE`) VALUES ('E8','J.Jones','Syst.Anal');





**Update Operation:**

The SQL **UPDATE** Query is used to modify the existing records in a table. You can use the WHERE clause with the UPDATE query to update the selected rows, otherwise all the rows would be affected.

The basic syntax of the UPDATE query with a WHERE clause is as follows –

UPDATE table\_name

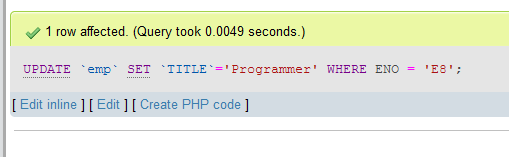
SET column1 = value1, column2 = value2...., columnN = valueN

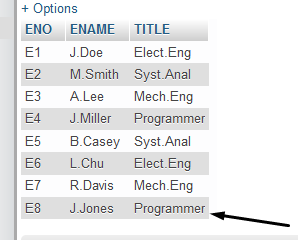
WHERE [condition];

You can combine N number of conditions using the AND or the OR operators.

**Example:**

UPDATE `emp` SET `TITLE`='Programmer' WHERE ENO = 'E8';





**Delete Operation:**

The SQL DELETE Query is used to delete the existing records from a table.

You can use the WHERE clause with a DELETE query to delete the selected rows, otherwise all the records would be deleted.

The basic syntax of the DELETE query with the WHERE clause is as follows –

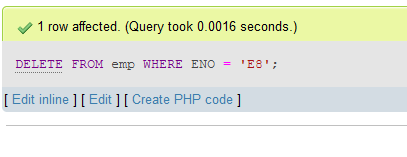
DELETE FROM table\_name

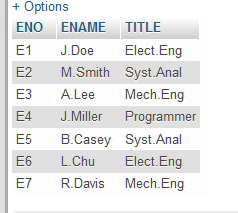
WHERE [condition];

You can combine N number of conditions using AND or OR operators.

**Example:**

DELETE FROM emp WHERE ENO = 'E8';





**Select Statement:**

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

The data returned is stored in a result table, called the result-set.

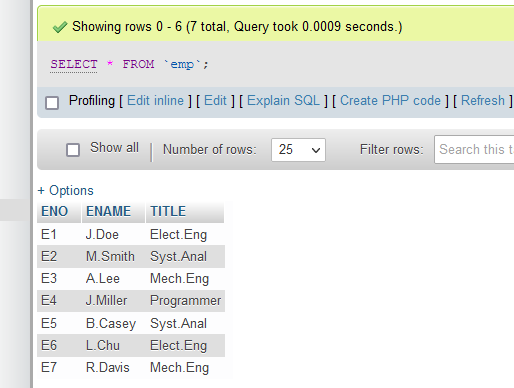
**Syntax:**

SELECT column1, column2, ...  
FROM table\_name;

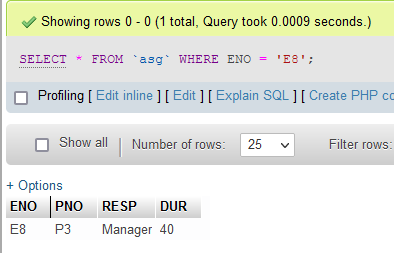
Here, column1, column2, ... are the field names of the table you want to select data from. If you want to select all the fields available in the table, use the following syntax:

SELECT \* FROM table\_name;

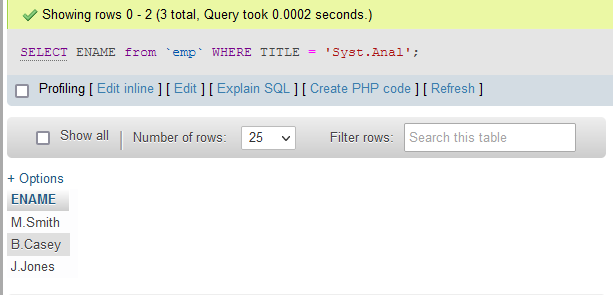
**Example:**



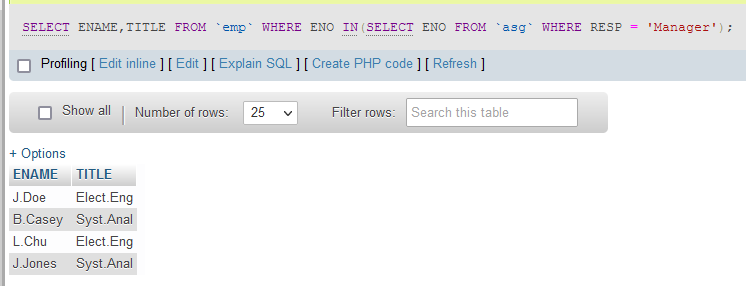
SELECT \* FROM `asg` WHERE ENO = 'E8';



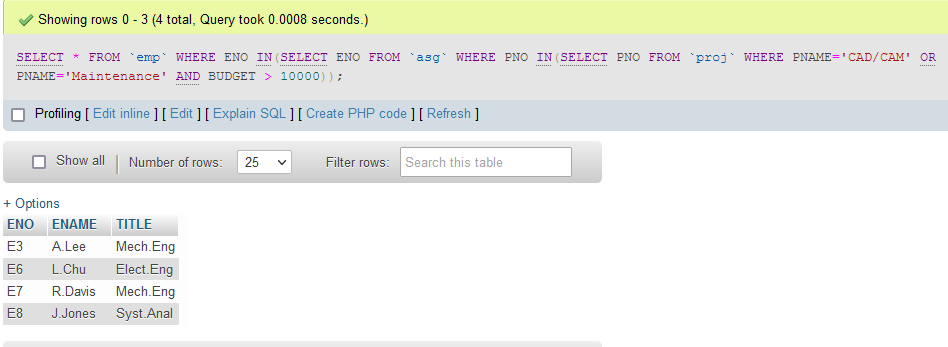
SELECT ENAME from `emp` WHERE TITLE = 'Syst.Anal';



SELECT ENAME,TITLE FROM `emp` WHERE ENO IN(SELECT ENO FROM `asg` WHERE RESP = 'Manager');



SELECT \* FROM `emp` WHERE ENO IN(SELECT ENO FROM `asg` WHERE PNO IN(SELECT PNO FROM `proj` WHERE PNAME='CAD/CAM' OR PNAME='Maintenance' AND BUDGET > 10000));



**ORDER BY:**

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.

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

SELECT column-list FROM table\_name

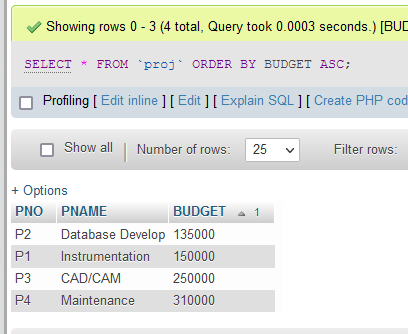
[WHERE condition]

[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:**

[SELECT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/8.0/en/select.html) \* FROM `proj` ORDER BY BUDGET ASC;



**Left Outer Join:**

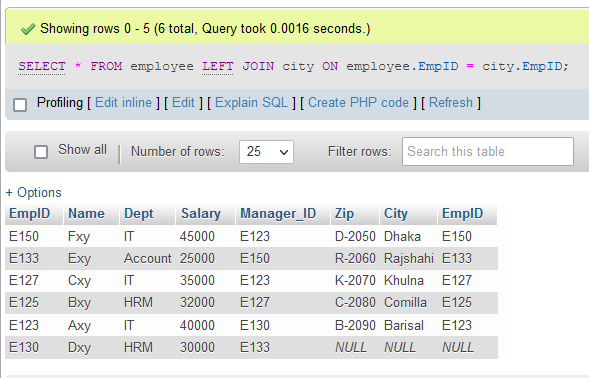
The LEFT JOIN keyword returns all records from the left table (table1), and the matching records from the right table (table2). The result is 0 records from the right side, if there is no match.

### **LEFT JOIN Syntax**

SELECT column\_name(s)  
FROM table1  
LEFT JOIN table2ON table1.column\_name = table2.column\_name;

**Example:**

SELECT \* FROM employee LEFT JOIN city ON employee.EmpID = city.EmpID;



**Right Outer Join:**

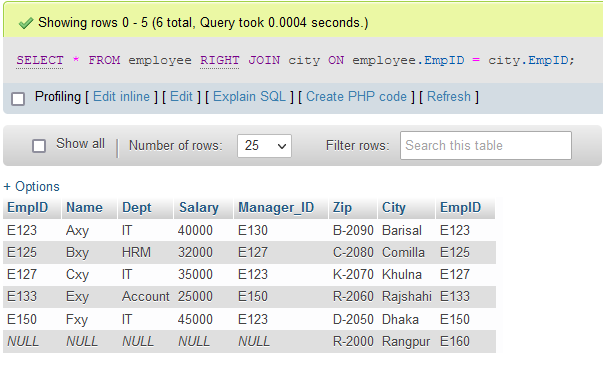
The RIGHT JOIN keyword returns all records from the right table (table2), and the matching records from the left table (table1). The result is 0 records from the left side, if there is no match.

### **RIGHT JOIN Syntax**

SELECT column\_name(s)  
FROM table1  
RIGHT JOIN table2ON table1.column\_name = table2.column\_name;

**Example:**

SELECT \* FROM employee RIGHT JOIN city ON employee.EmpID = city.EmpID;



**full outer Join:**

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

SELECT \* FROM employee LEFT JOIN city ON employee.EmpID = city.EmpID

UNION

SELECT \* FROM employee RIGHT JOIN city ON employee.EmpID = city.EmpID;



**View:**

In SQL, a view is a virtual table based on the result-set of an SQL statement.

A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database.

You can add SQL statements and functions to a view and present the data as if the data were coming from one single table.

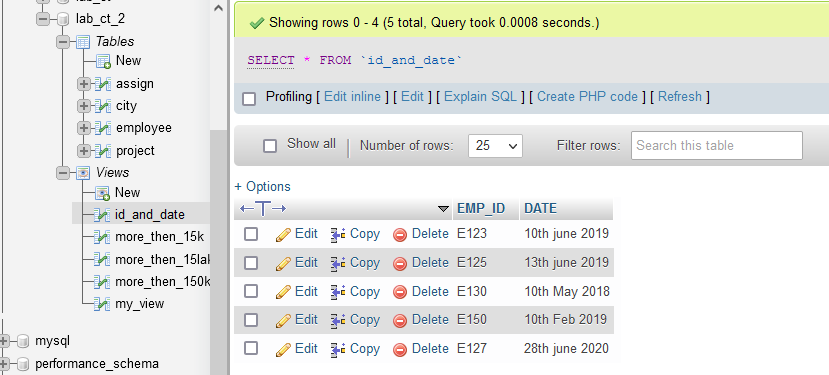
A view is created with the CREATE VIEW statement.

### **CREATE VIEW Syntax**

CREATE VIEW view\_name AS  
SELECT column1, column2, ...  
FROM table\_name  
WHERE condition;

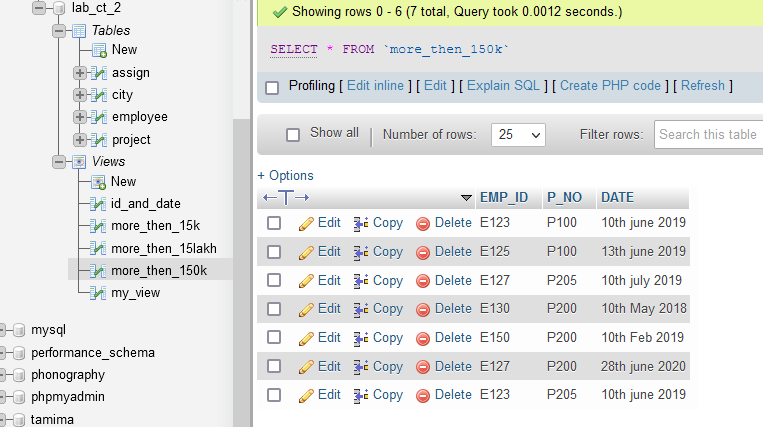
**Example:**

CREATE VIEW ID\_and\_Date as SELECT EMP\_ID, DATE FROM assign WHERE P\_NO IN(SELECT P\_NO FROM project WHERE P\_NAME = 'Instrumentation' OR P\_NAME = 'CAD');



**Example:**

CREATE VIEW more\_then\_150k AS SELECT \* FROM assign WHERE P\_NO IN(SELECT P\_NO FROM project WHERE BUDGET > 150000);

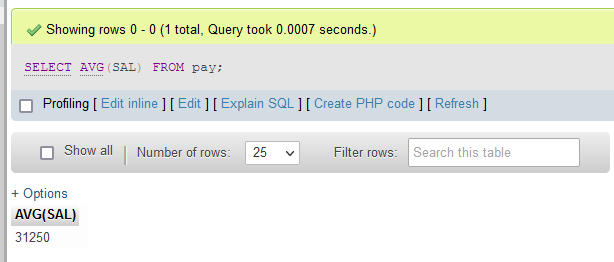


**Average:**

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

**Example:**

SELECT AVG(SAL) FROM pay;



**Example:**

SELECT AVG(DUR) FROM asg WHERE PNO = 'P2';

