



College Code : SDES

**SREE DATTHA INSTITUTE OF ENGINEERING & SCIENCE**  
(Autonomous, NAAC A+, NBA Accredited, Affiliated to JNTUH, Approved by AICTE, New Delhi)  
Nagarjuna Sagar Road, Sheriguda (V), Ibrahimpatnam (M), R.R. Dist., Greater Hyderabad, Telangana-501510.  
Website: [www.sreedattha.ac.in](http://www.sreedattha.ac.in) Email: [info@sreedattha.ac.in](mailto:info@sreedattha.ac.in)

## Department of Computer Science and Engineering



### LAB Notes

on

### Database Management System Laboratory

**2<sup>nd</sup> Year 2<sup>nd</sup> Semester**

by

**Prof. Puneet Shetteppanavar**

Assistant Professor, Computer Science and Engineering Department,  
Sree Dattha Institute of Engineering and Science, Sheriguda, Hyderabad - 501510

---

## DATABASE MANAGEMENT SYSTEMS LAB

B.Tech. II Year II Sem.

L	T	P	C
0	0	2	1

**Co-requisites:** "Database Management Systems"

### Course Objectives:

- Introduce ER data model, database design and normalization
- Learn SQL basics for data definition and data manipulation

### Course Outcomes:

- Design database schema for a given application and apply normalization
- Acquire skills in using SQL commands for data definition and data manipulation.
- Develop solutions for database applications using procedures, cursors and triggers

### List of Experiments:

1. Concept design with E-R Model
2. Relational Model
3. Normalization
4. Practicing DDL commands
5. Practicing DML commands
6. A. Querying (using ANY, ALL, UNION, INTERSECT, JOIN, Constraints etc.)  
B. **Nested, Correlated subqueries**
7. Queries using Aggregate functions, GROUP BY, HAVING and Creation and dropping of Views.
8. Triggers (Creation of insert trigger, delete trigger, update trigger)
9. Procedures
10. Usage of Cursors

### TEXT BOOKS:

1. Database Management Systems, Raghurama Krishnan, Johannes Gehrke, Tata Mc Graw Hill, 3<sup>rd</sup> Edition
2. Database System Concepts, Silberschatz, Korth, McGraw Hill, V edition.

### REFERENCE BOOKS:

1. Database Systems design, Implementation, and Management, Peter Rob & Carlos Coronel 7<sup>th</sup> Edition.
2. Fundamentals of Database Systems, Elmasri Navrate, Pearson Education
3. Introduction to Database Systems, C.J. Date, Pearson Education
4. Oracle for Professionals, The X Team, S. Shah and V. Shah, SPD.
5. Database Systems Using Oracle: A Simplified guide to SQL and PL/SQL, Shah, PHI.
6. Fundamentals of Database Management Systems, M. L. Gillenson, Wiley Student Edition.

Note: {Previous Hand Written Notes will be Added Soon}

# MySQL Aggregate Functions

MySQL's aggregate function is used to **perform calculations on multiple values and return the result in a single value like the average of all values**, the sum of all values, and maximum & minimum value among certain groups of values. We mostly use the aggregate functions with [SELECT statements](#) in the data query languages.

## Syntax:

The following are the syntax to use aggregate functions in MySQL:

function\_name (**DISTINCT** | ALL expression)

In the above syntax, we had used the following parameters:

- First, we need to specify the name of the aggregate function.
- Second, we use the **DISTINCT** modifier when we want to calculate the result based on distinct values or **ALL** modifiers when we calculate all values, including duplicates. The default is ALL.
- Third, we need to specify the expression that involves columns and arithmetic operators.

There are various aggregate functions available in [MySQL](#). Some of the most commonly used aggregate functions are summarised in the below table:

Aggregate Function	Descriptions
<a href="#">count()</a>	It returns the number of rows, including rows with NULL values in a group.
<a href="#">sum()</a>	It returns the total summed values (Non-NULL) in a set.
<a href="#">average()</a>	It returns the average value of an expression.
<a href="#">min()</a>	It returns the minimum (lowest) value in a set.
<a href="#">max()</a>	It returns the maximum (highest) value in a set.

<a href="#"><u>group_concat()</u></a>	It returns a concatenated string.
<a href="#"><u>first()</u></a>	It returns the first value of an expression.
<a href="#"><u>last()</u></a>	It returns the last value of an expression.

## Why we use aggregate functions?

We mainly use the aggregate functions in databases, spreadsheets and many other data manipulation software packages. In the context of business, different organization levels need different information such as top levels managers interested in knowing whole figures and not the individual details. These functions produce the summarised data from our database. Thus they are extensively used in economics and finance to represent the economic health or stock and sector performance.

Let us take an example of myflix (video streaming website which has huge collections of the movie) database, where management may require the following details:

- Most rented movies.
- Least rented movies.
- Average number that each movie is rented out in a month.

We can easily produce these details with the help of aggregate functions.

Let us discuss the most commonly used aggregate functions in detail. First, we will create a new table for the demonstration of all aggregate functions.

Execute the below statement to create an **employee** table:

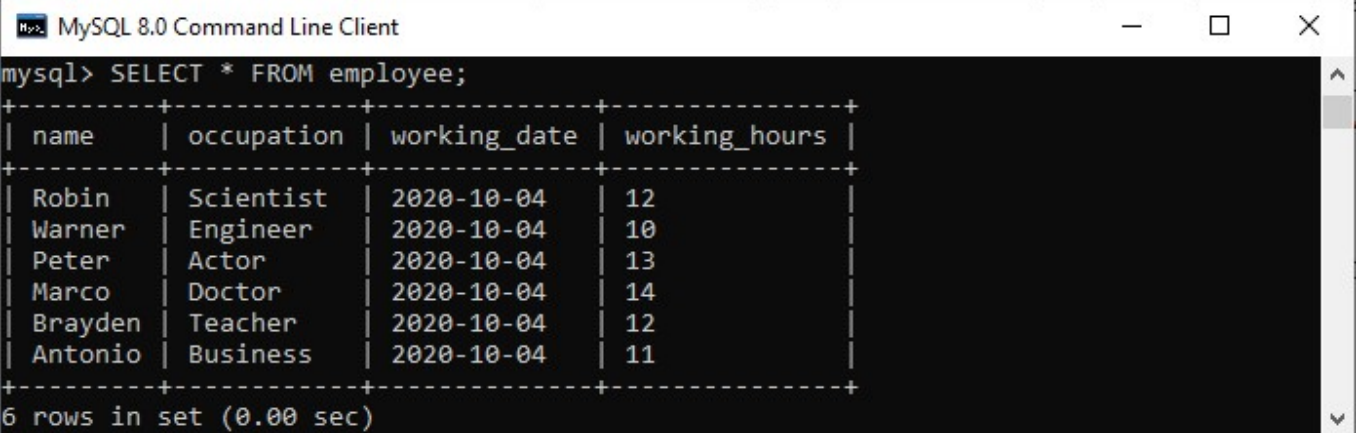
```
CREATE TABLE employee(  
  name varchar(45) NOT NULL,  
  occupation varchar(35) NOT NULL,  
  working_date date,  
  working_hours varchar(10)  
);
```

Execute the below statement to **insert the records** into the employee table:

### INSERT INTO employee VALUES

```
('Robin', 'Scientist', '2020-10-04', 12),  
('Warner', 'Engineer', '2020-10-04', 10),  
('Peter', 'Actor', '2020-10-04', 13),  
('Marco', 'Doctor', '2020-10-04', 14),  
('Brayden', 'Teacher', '2020-10-04', 12),  
('Antonio', 'Business', '2020-10-04', 11);
```

Now, execute the **SELECT statement** to show the record:



```
mysql> SELECT * FROM employee;  
+-----+-----+-----+-----+  
| name   | occupation | working_date | working_hours |  
+-----+-----+-----+-----+  
| Robin  | Scientist  | 2020-10-04   | 12            |  
| Warner | Engineer   | 2020-10-04   | 10            |  
| Peter  | Actor      | 2020-10-04   | 13            |  
| Marco  | Doctor     | 2020-10-04   | 14            |  
| Brayden | Teacher    | 2020-10-04   | 12            |  
| Antonio | Business   | 2020-10-04   | 11            |  
+-----+-----+-----+-----+  
6 rows in set (0.00 sec)
```

## Count() Function

MySQL count() function **returns the total number of values** in the expression. This function produces all rows or only some rows of the table based on a specified condition, and its return type is **BIGINT**. It returns zero if it does not find any matching rows. It can work with both numeric and non-numeric data types.

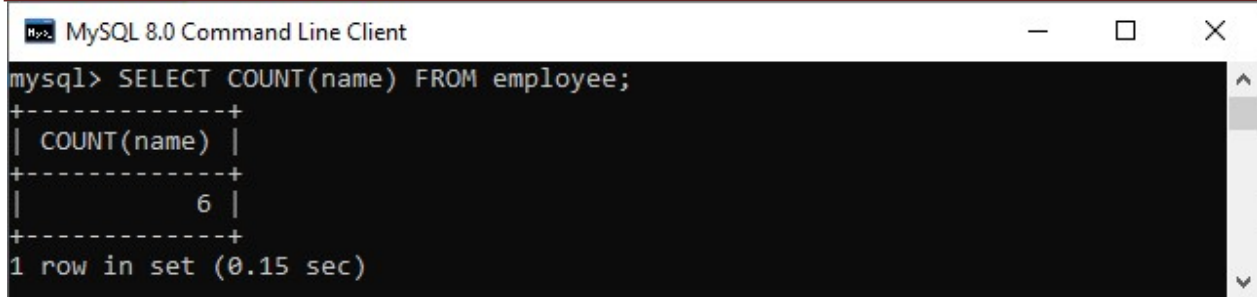
### Example

Suppose we want to get the total number of employees in the employee table, we need to use the count() function as shown in the following query:

```
mysql> SELECT COUNT(name) FROM employee;
```

### Output:

After execution, we can see that this table has six employees.



```
mysql> SELECT COUNT(name) FROM employee;
+-----+
| COUNT(name) |
+-----+
|          6 |
+-----+
1 row in set (0.15 sec)
```

## Sum() Function

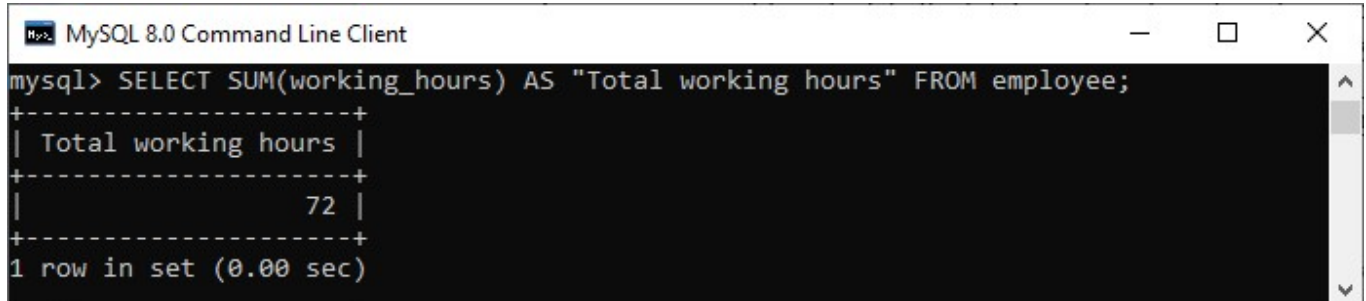
The MySQL sum() function **returns the total summed (non-NULL) value** of an expression. It returns NULL if the result set does not have any rows. It works with numeric data type only.

Suppose we want to calculate the total number of working hours of all employees in the table, we need to use the sum() function as shown in the following query:

```
mysql> SELECT SUM(working_hours) AS "Total working hours" FROM employee;
```

### Output:

After execution, we can see the total working hours of all employees in the table.



```
mysql> SELECT SUM(working_hours) AS "Total working hours" FROM employee;
+-----+
| Total working hours |
+-----+
|          72 |
+-----+
1 row in set (0.00 sec)
```

## AVG() Function

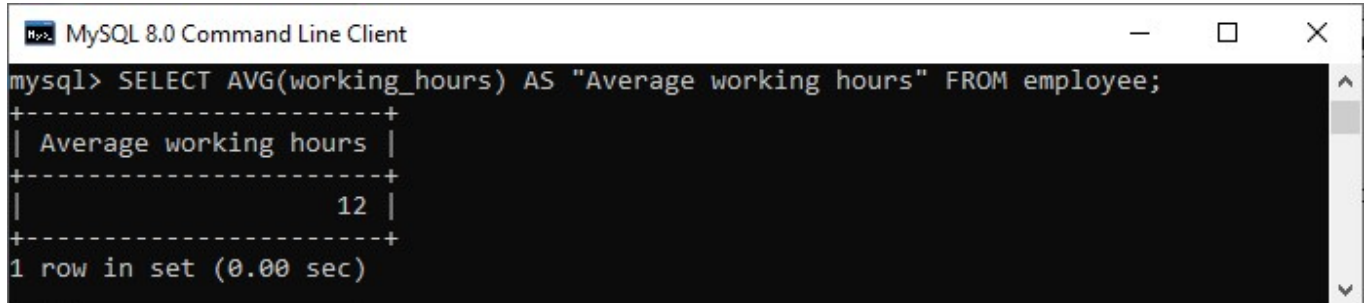
MySQL AVG() function **calculates the average of the values** specified in the column. Similar to the SUM() function, it also works with numeric data type only.

Suppose we want to get the average working hours of all employees in the table, we need to use the AVG() function as shown in the following query:

```
mysql> SELECT AVG(working_hours) AS "Average working hours" FROM employee;
```

### Output:

After execution, we can see that the average working hours of all employees in the organization:



```
mysql> SELECT AVG(working_hours) AS "Average working hours" FROM employee;
+-----+
| Average working hours |
+-----+
| 12 |
+-----+
1 row in set (0.00 sec)
```

### MIN() Function

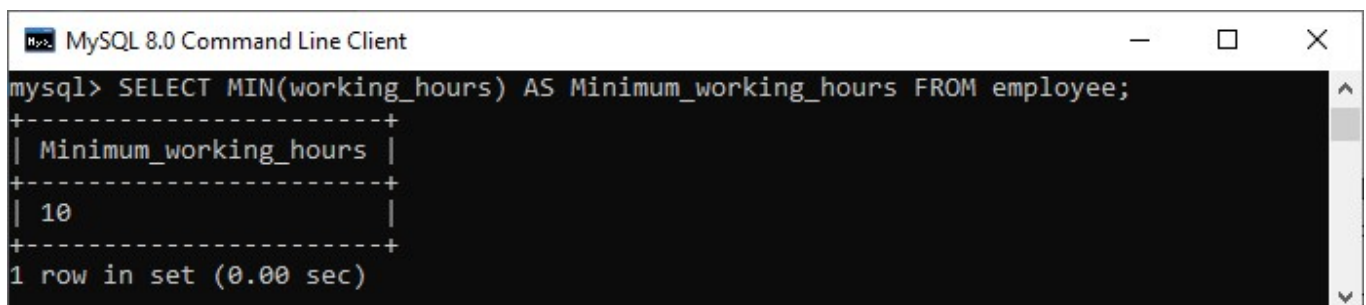
MySQL MIN() function **returns the minimum (lowest) value** of the specified column. It also works with numeric data type only.

Suppose we want to get minimum working hours of an employee available in the table, we need to use the MIN() function as shown in the following query:

```
mysql> SELECT MIN(working_hours) AS Minimum_working_hours FROM employee;
```

### Output:

After execution, we can see that the minimum working hours of an employee available in the table:



```
mysql> SELECT MIN(working_hours) AS Minimum_working_hours FROM employee;
+-----+
| Minimum_working_hours |
+-----+
| 10 |
+-----+
1 row in set (0.00 sec)
```



## MAX() Function

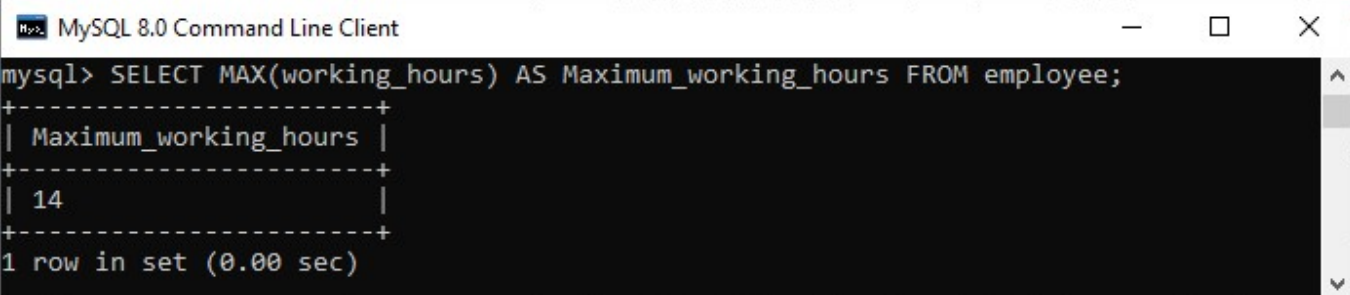
MySQL MAX() function **returns the maximum (highest) value** of the specified column. It also works with numeric data type only.

Suppose we want to get maximum working hours of an employee available in the table, we need to use the MAX() function as shown in the following query:

```
mysql> SELECT MAX(working_hours) AS Maximum_working_hours FROM employee;
```

### Output:

After execution, we can see that the maximum working hours of an employee available in the table:



```
mysql> SELECT MAX(working_hours) AS Maximum_working_hours FROM employee;
+-----+
| Maximum_working_hours |
+-----+
| 14                     |
+-----+
1 row in set (0.00 sec)
```

## FIRST() Function

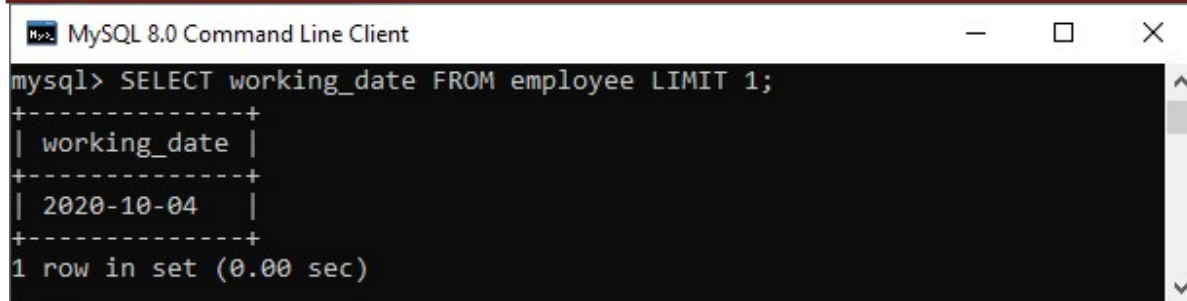
This function **returns the first value** of the specified column. To get the first value of the column, we must have to use the **LIMIT** clause. It is because FIRST() function only supports in MS Access.

Suppose we want to get the first working date of an employee available in the table, we need to use the following query:

```
mysql> SELECT working_date FROM employee LIMIT 1;
```

### Output:

After execution, we can see that the first working date of an employee available in the table:

A screenshot of the MySQL 8.0 Command Line Client window. The title bar reads "MySQL 8.0 Command Line Client". The command prompt shows the command: `mysql> SELECT working_date FROM employee LIMIT 1;`. The output is a table with one column, `working_date`, and one row containing the value `2020-10-04`. The output is formatted with a header row and a separator row. The status bar at the bottom indicates "1 row in set (0.00 sec)".

```
mysql> SELECT working_date FROM employee LIMIT 1;
+-----+
| working_date |
+-----+
| 2020-10-04   |
+-----+
1 row in set (0.00 sec)
```

## LAST() Function

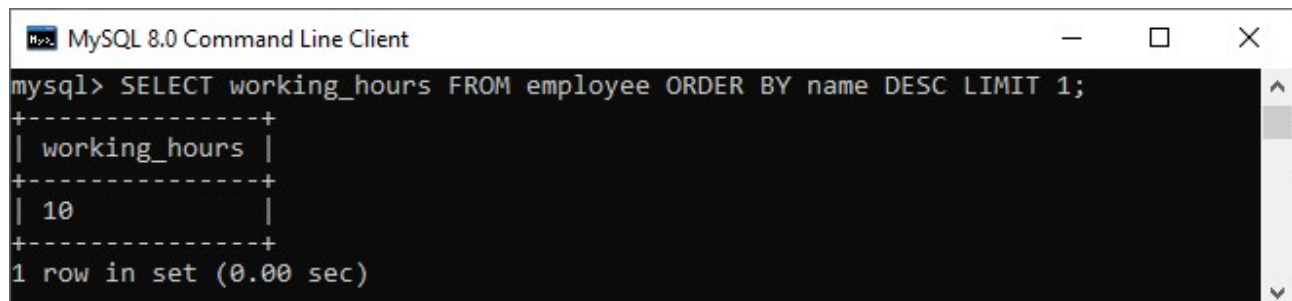
This function **returns the last value** of the specified column. To get the last value of the column, we must have to use the **ORDER BY** and **LIMIT** clause. It is because the LAST() function only supports in MS Access.

Suppose we want to get the last working hour of an employee available in the table, we need to use the following query:

```
mysql> SELECT working_hours FROM employee ORDER BY name DESC LIMIT 1;
```

### Output:

After execution, we can see that the last working hour of an employee available in the table:

A screenshot of the MySQL 8.0 Command Line Client window. The title bar reads "MySQL 8.0 Command Line Client". The command prompt shows the command: `mysql> SELECT working_hours FROM employee ORDER BY name DESC LIMIT 1;`. The output is a table with one column, `working_hours`, and one row containing the value `10`. The output is formatted with a header row and a separator row. The status bar at the bottom indicates "1 row in set (0.00 sec)".

```
mysql> SELECT working_hours FROM employee ORDER BY name DESC LIMIT 1;
+-----+
| working_hours |
+-----+
| 10            |
+-----+
1 row in set (0.00 sec)
```

## GROUP\_CONCAT() Function

The GROUP\_CONCAT() function **returns the concatenated string from multiple rows into a single string**. If the group contains at least one non-null value, it always returns a string value. Otherwise, we will get a null value.

Suppose we have another employee table as below:

emp_id	emp_fname	emp_lname	dept_id	designation
1	David	Miller	2	Engineer
2	Peter	Watson	3	Manager
3	Mark	Boucher	1	Scientist
2	Peter	Watson	3	BDE
1	David	Miller	2	Developer
4	Adam	Warner	4	Receptionist
3	Mark	Boucher	1	Engineer
4	Adam	Warner	4	Clerk

If we want to concatenate the designation of the same dept\_id on the employee table, we need to use the following query:

```
mysql> SELECT emp_id, emp_fname, emp_lname, dept_id,  
GROUP_CONCAT(designation) as "designation" FROM employee group by emp_id;
```

### Output:

After execution, we can see that the designation of the same dept\_id concatenated successfully:

emp_id	emp_fname	emp_lname	dept_id	designation
1	David	Miller	2	Engineer,Developer
2	Peter	Watson	3	Manager,BDE
3	Mark	Boucher	1	Scientist,Engineer
4	Adam	Warner	4	Receptionist,Clerk