

## 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:

1. 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>	returns the number of rows, including rows with NULL values in a group.
<a href="#">sum()</a>	returns the total summed values (Non-NULL) in a set.
<a href="#">average()</a>	returns the average value of an expression.
<a href="#">min()</a>	returns the minimum (lowest) value in a set.
<a href="#">max()</a>	returns the maximum (highest) value in a set.
<a href="#">group_concat()</a>	returns a concatenated string.
<a href="#">first()</a>	returns the first value of an expression.
<a href="#">last()</a>	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.

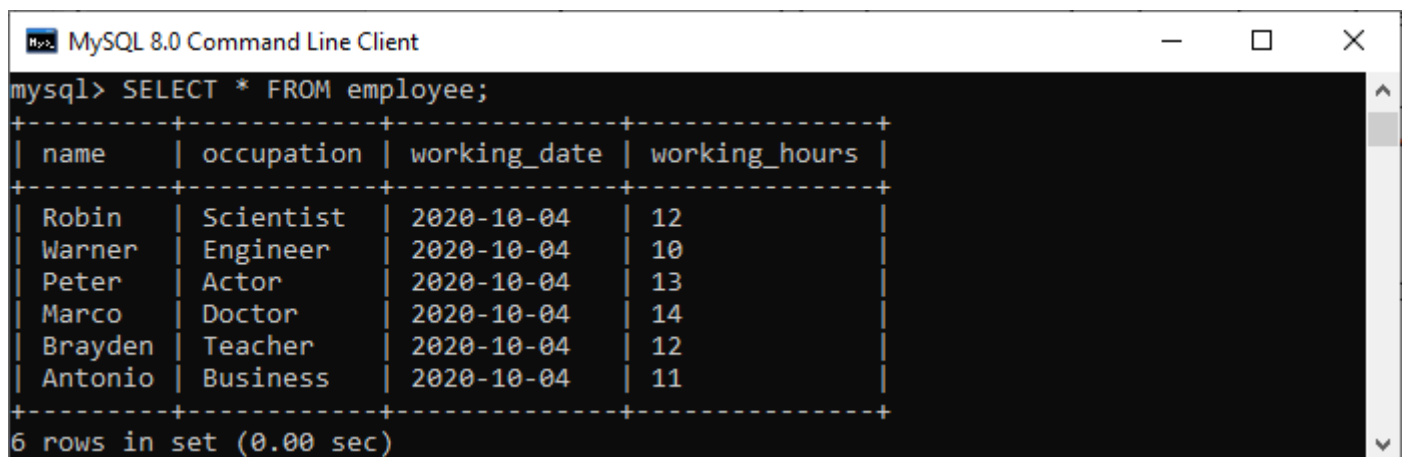
Execute the below statement to create an **employee** table:

1. **CREATE TABLE** employee(  
2.     **name** **varchar**(45) NOT NULL,  
3.     **occupation** **varchar**(35) NOT NULL,  
4.     **working\_date** **date**,  
5.     **working\_hours** **varchar**(10)  
6. );

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

1. **INSERT INTO** employee **VALUES**  
2. ('Robin', 'Scientist', '2020-10-04', 12),  
3. ('Warner', 'Engineer', '2020-10-04', 10),  
4. ('Peter', 'Actor', '2020-10-04', 13),  
5. ('Marco', 'Doctor', '2020-10-04', 14),  
6. ('Brayden', 'Teacher', '2020-10-04', 12),  
7. ('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.

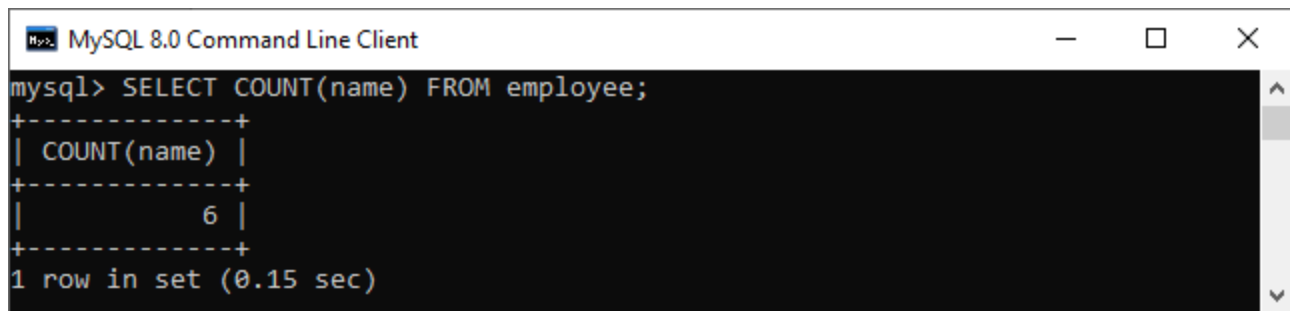
## Example

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:

1. 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

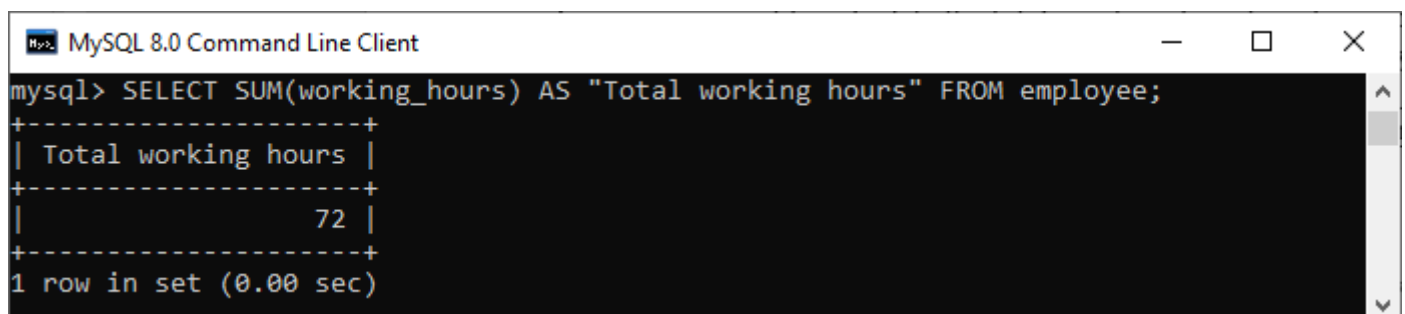
The MySQL sum() function **returns the total summed (non-NULL) value** of an expression. It returns NULL if the result set does not have any rows. It works with numeric data type only.

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:

1. 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:

1. 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 8.0 Command Line Client
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:

1. 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 8.0 Command Line Client
mysql> SELECT MIN(working_hours) AS Minimum_working_hours FROM employee;
+-----+
| Minimum_working_hours |
+-----+
| 10 |
+-----+
1 row in set (0.00 sec)
```

To read more information, [click here](#).

## 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:

1. 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 8.0 Command Line Client
mysql> SELECT MAX(working_hours) AS Maximum_working_hours FROM employee;
+-----+
| Maximum_working_hours |
+-----+
| 14 |
+-----+
1 row in set (0.00 sec)
```

To read more information, [click here](#).

### FIRST() Function

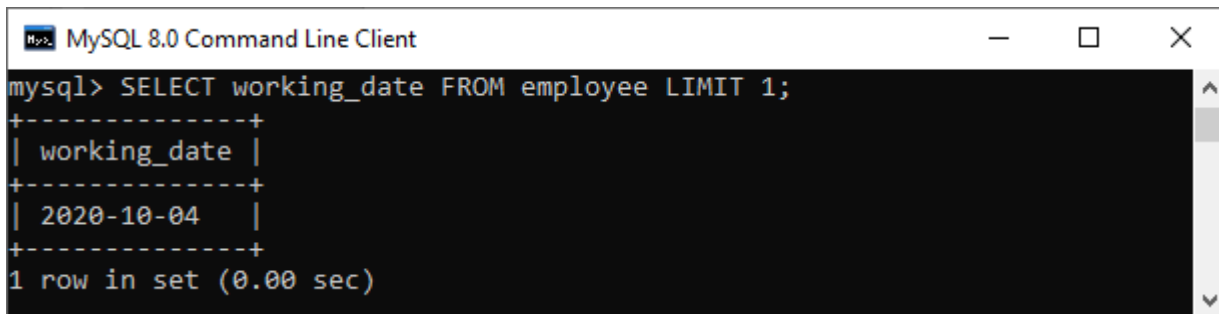
This function **returns the first value** of the specified column. To get the first value of the column, we must have to use the **LIMIT** clause. It is because FIRST() function only supports in MS Access.

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

1. 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:



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

To read more information, [click here](#).

### LAST() Function

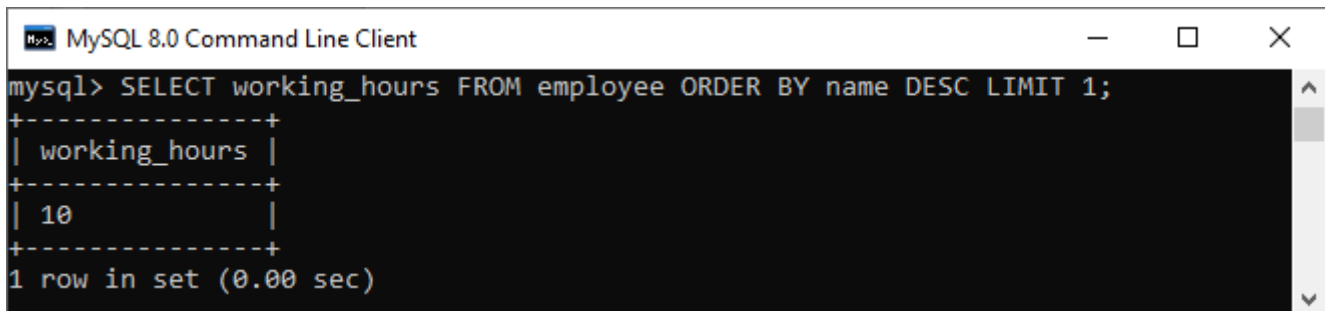
This function **returns the last value** of the specified column. To get the last value of the column, we must have to use the **ORDER BY** and **LIMIT** clause. It is because the LAST() function only supports in MS Access.

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

1. 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:



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

To read more information, [click here](#).

### 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:

1. mysql> **SELECT** emp\_id, emp\_fname, emp\_lname, dept\_id,
2. 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

## MySQL Count() Function

MySQL count() function is used to returns the count of an expression. It allows us to count all rows or only some rows of the table that matches a specified condition. It is a type of aggregate function whose return type is BIGINT. This function returns 0 if it does not find any matching rows.

We can use the count function in three forms, which are explained below:

- Count (\*)
- Count (expression)
- Count (distinct)

Let us discuss each in detail.

**COUNT(\*) Function:** This function uses the [SELECT statement](#) to returns the count of rows in a result set. The result set contains all Non-Null, Null, and duplicates rows.

**COUNT(expression) Function:** This function returns the result set without containing Null rows as the result of an expression.

**COUNT(distinct expression) Function:** This function returns the count of distinct rows without containing NULL values as the result of the expression.

### Syntax

The following are the syntax of the COUNT() function:

1. **SELECT COUNT** (aggregate\_expression)
2. **FROM** table\_name
3. [**WHERE** conditions];

#### Parameter explanation

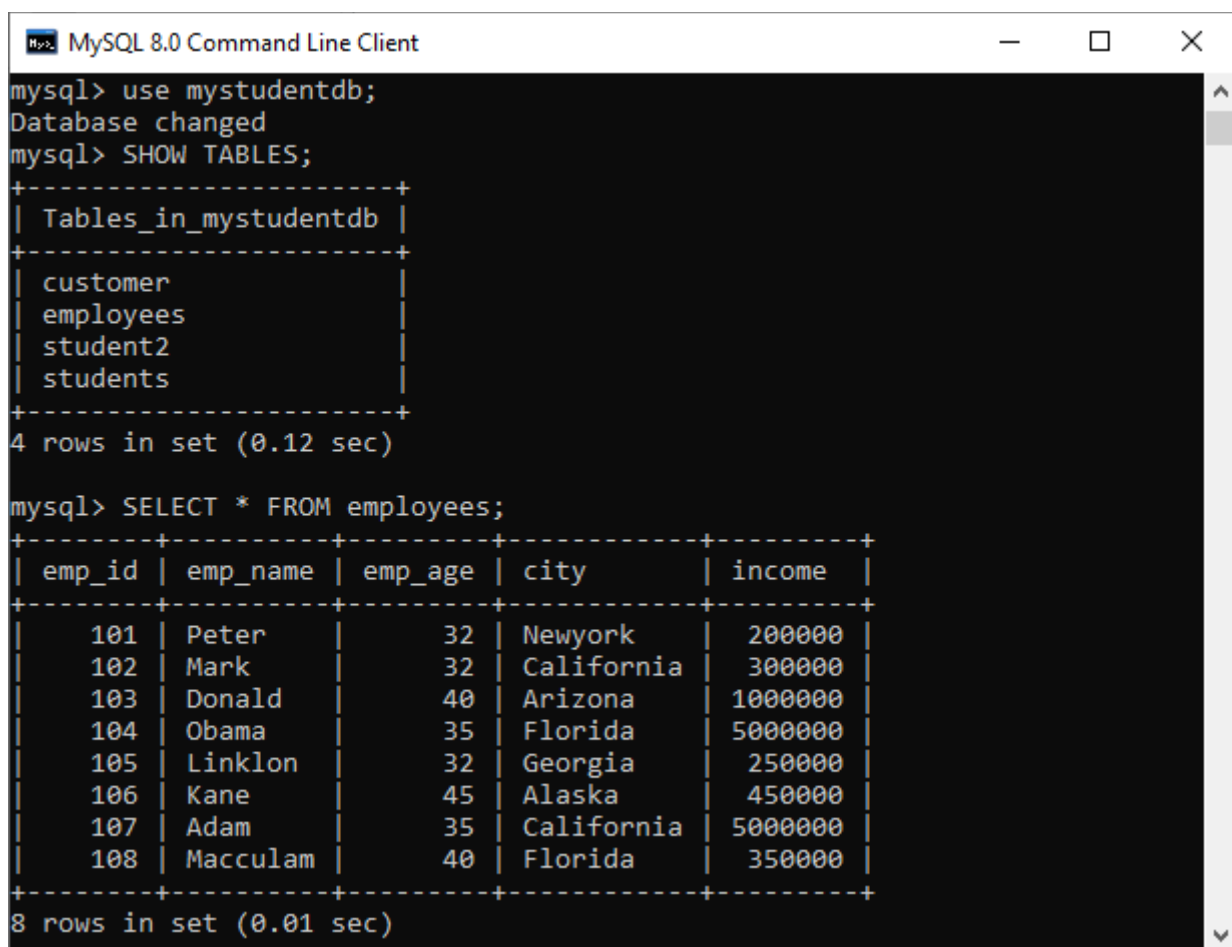
**aggregate\_expression:** It specifies the column or expression whose NON-NULL values will be counted.

**table\_name:** It specifies the tables from where you want to retrieve records. There must be at least one table listed in the [FROM clause](#).

**WHERE conditions:** It is optional. It specifies the conditions that must be fulfilled for the records to be selected.

#### MySQL count() function example

Consider a table named "employees" that contains the following data.



```
mysql> use mystudentdb;
Database changed
mysql> SHOW TABLES;
+-----+
| Tables_in_mystudentdb |
+-----+
| customer                |
| employees                |
| student2                |
| students                |
+-----+
4 rows in set (0.12 sec)

mysql> SELECT * FROM employees;
+-----+-----+-----+-----+-----+
| emp_id | emp_name | emp_age | city       | income |
+-----+-----+-----+-----+-----+
| 101    | Peter   | 32     | Newyork    | 200000 |
| 102    | Mark    | 32     | California | 300000 |
| 103    | Donald  | 40     | Arizona    | 1000000 |
| 104    | Obama   | 35     | Florida    | 5000000 |
| 105    | Linklon | 32     | Georgia    | 250000 |
| 106    | Kane    | 45     | Alaska     | 450000 |
| 107    | Adam    | 35     | California | 5000000 |
| 108    | Macculam | 40     | Florida    | 350000 |
+-----+-----+-----+-----+-----+
8 rows in set (0.01 sec)
```

Let us understand how count() functions work in [MySQL](#).

#### Example1

Execute the following query that uses the COUNT(expression) function to calculates the total number of employees name available in the table:

1. mysql> **SELECT COUNT**(emp\_name) **FROM** employees;

**Output:**

```
MySQL 8.0 Command Line Client

mysql> SELECT COUNT(emp_name) FROM employees;
+-----+
| COUNT(emp_name) |
+-----+
|                8 |
+-----+
1 row in set (0.00 sec)
```

### Example2

Execute the following statement that returns all rows from the employee table and [WHERE clause](#) specifies the rows whose value in the column emp\_age is greater than 32:

1. mysql> **SELECT COUNT(\*) FROM employees WHERE emp\_age>32;**

Output:

```
MySQL 8.0 Command Line Client

mysql> SELECT COUNT(*) FROM employees WHERE emp_age>32;
+-----+
| COUNT(*) |
+-----+
|         5 |
+-----+
1 row in set (0.00 sec)
```

### Example3

This statement uses the COUNT(distinct expression) function that counts the Non-Null and distinct rows in the column emp\_age:

1. mysql> **SELECT COUNT(DISTINCT emp\_age) FROM employees;**

Output:

```
MySQL 8.0 Command Line Client

mysql> SELECT COUNT(DISTINCT emp_age) FROM employees;
+-----+
| COUNT(DISTINCT emp_age) |
+-----+
|                        4 |
+-----+
1 row in set (0.00 sec)
```

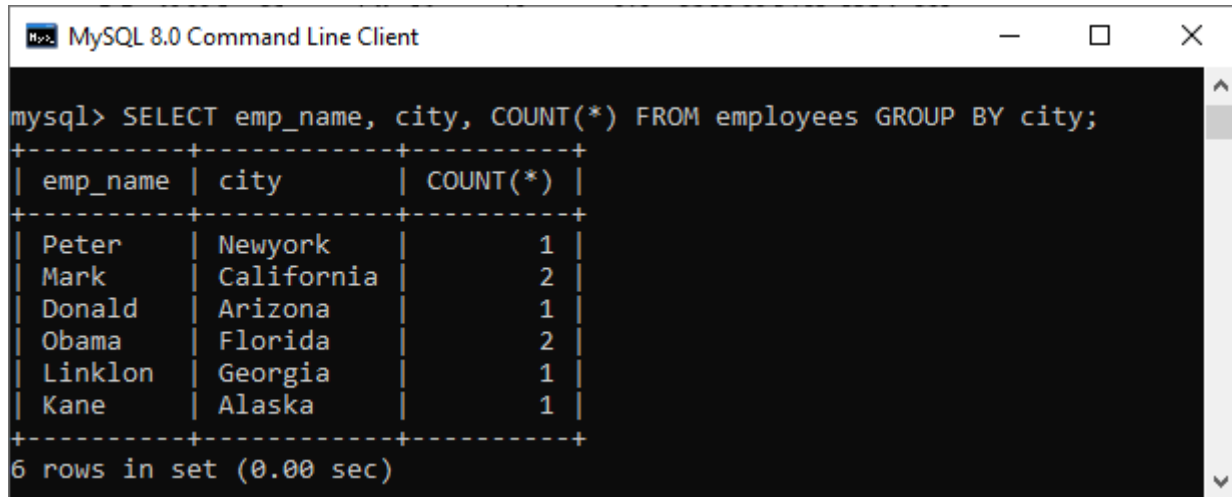
### MySQL Count() Function with GROUP BY Clause

We can also use the count() function with the GROUP BY clause that returns the count of the element in each group. For example, the following statement returns the number of employee in each city:

1. mysql> **SELECT emp\_name, city, COUNT(\*) FROM employees GROUP BY city;**



After the successful execution, we will get the result as below:



```
mysql> SELECT emp_name, city, COUNT(*) FROM employees GROUP BY city;
```

emp_name	city	COUNT(*)
Peter	Newyork	1
Mark	California	2
Donald	Arizona	1
Obama	Florida	2
Linklon	Georgia	1
Kane	Alaska	1

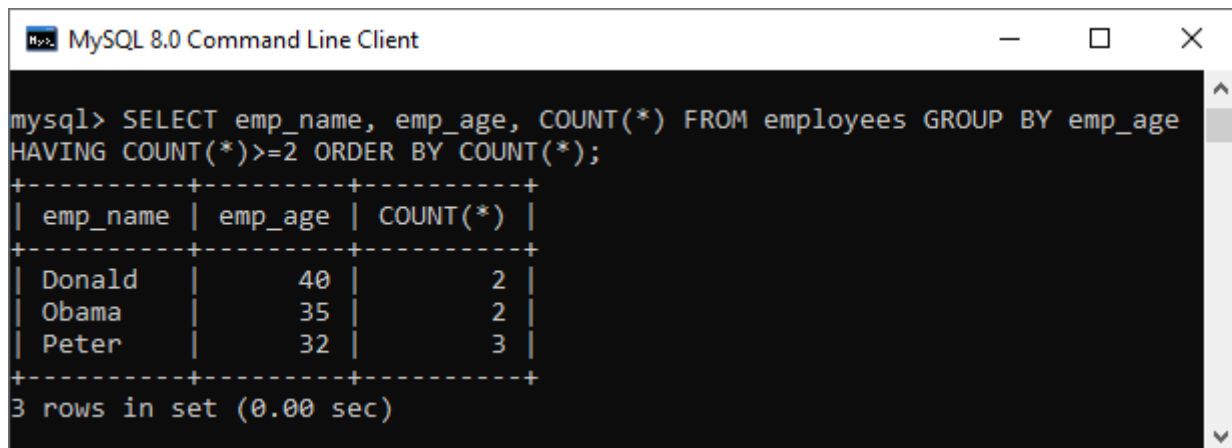
6 rows in set (0.00 sec)

### MySQL Count() Function with HAVING and ORDER BY Clause

Let us see another clause that uses ORDER BY and Having clause with the count() function. Execute the following statement that gives the employee name who has at least two age same and sorts them based on the count result:

1. mysql> **SELECT** emp\_name, emp\_age, **COUNT**(\*) **FROM** employees
2. **GROUP BY** emp\_age
3. **HAVING COUNT**(\*)>=2
4. **ORDER BY COUNT**(\*);

This statement will give the output as below:



```
mysql> SELECT emp_name, emp_age, COUNT(*) FROM employees GROUP BY emp_age
HAVING COUNT(*)>=2 ORDER BY COUNT(*);
```

emp_name	emp_age	COUNT(*)
Donald	40	2
Obama	35	2
Peter	32	3

3 rows in set (0.00 sec)

### MySQL sum() function

The MySQL sum() function is used to return the total summed value of an expression. It returns **NULL** if the result set does not have any rows. It is one of the kinds of aggregate functions in MySQL.

#### Syntax

Following are the syntax of sum() function in MySQL:

1. **SELECT SUM**(aggregate\_expression)
2. **FROM** tables
3. [**WHERE** conditions];

## Parameter Explanation

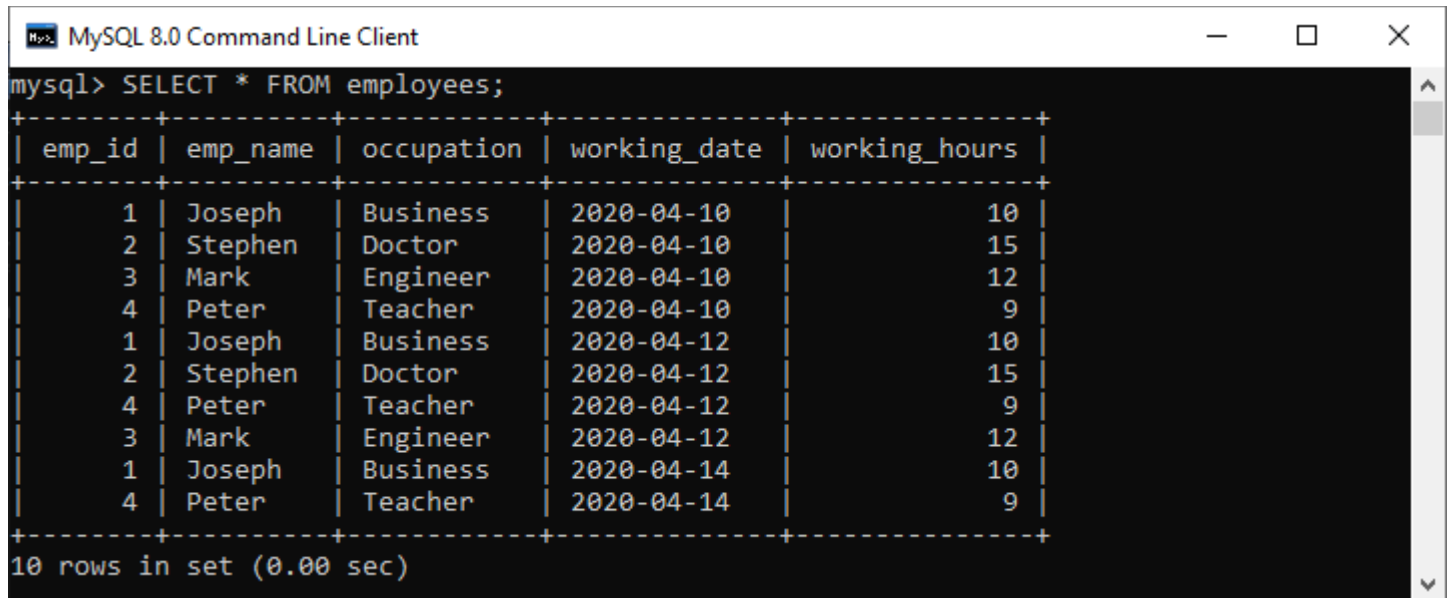
**aggregate\_expression:** It specifies the column or expression that we are going to calculate the sum.

**table\_name:** It specifies the tables from where we want to retrieve records. There must be at least one table listed in the FROM clause.

**WHERE conditions:** It is optional. It specifies the conditions that must be fulfilled for the records to be selected.

## MySQL sum() function example

Consider our database has a table named **employees**, having the following data. Now, we are going to understand this function with various examples:



```
mysql> SELECT * FROM employees;
```

emp_id	emp_name	occupation	working_date	working_hours
1	Joseph	Business	2020-04-10	10
2	Stephen	Doctor	2020-04-10	15
3	Mark	Engineer	2020-04-10	12
4	Peter	Teacher	2020-04-10	9
1	Joseph	Business	2020-04-12	10
2	Stephen	Doctor	2020-04-12	15
4	Peter	Teacher	2020-04-12	9
3	Mark	Engineer	2020-04-12	12
1	Joseph	Business	2020-04-14	10
4	Peter	Teacher	2020-04-14	9

10 rows in set (0.00 sec)

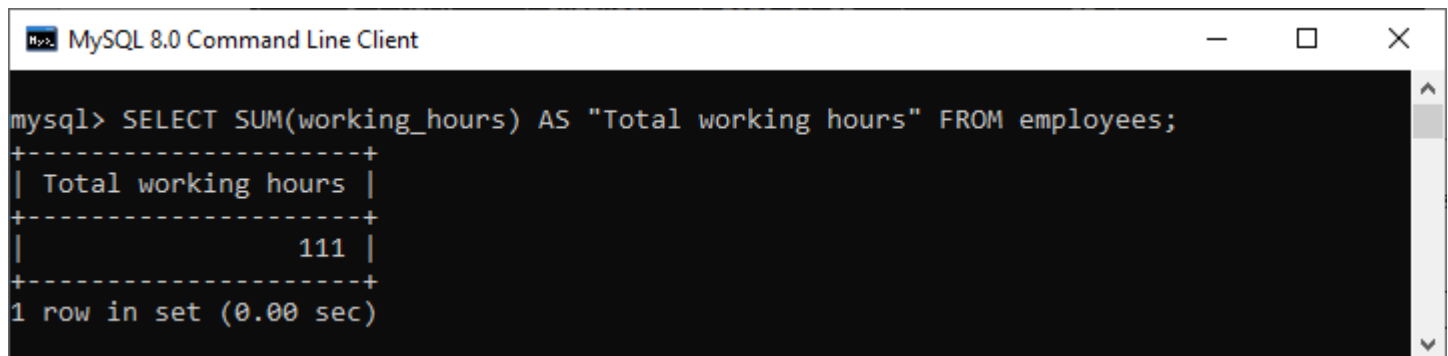
### 1. Basic Example

Execute the following query that calculates the total number of working hours of all employees in the table:

```
1. mysql> SELECT SUM(working_hours) AS "Total working hours" FROM employees;
```

**Output:**

We will get the result as below:



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

Total working hours
111

1 row in set (0.00 sec)

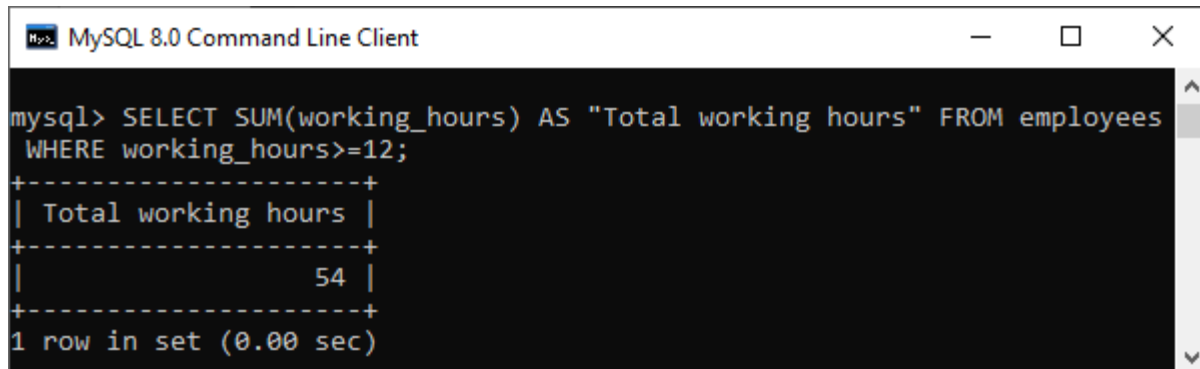
### 2. MySQL sum() function with WHERE clause

This example is used to return the result based on the condition specified in the WHERE clause. Execute the following query to calculate the total working hours of employees whose **working\_hours**  $\geq$  12.

1. mysql> **SELECT SUM**(working\_hours) **AS** "Total working hours" **FROM** employees **WHERE** working\_hour s>=12;

#### Output:

This statement will give the output as below:



```
mysql> SELECT SUM(working_hours) AS "Total working hours" FROM employees
WHERE working_hours>=12;
+-----+
| Total working hours |
+-----+
|                54 |
+-----+
1 row in set (0.00 sec)
```

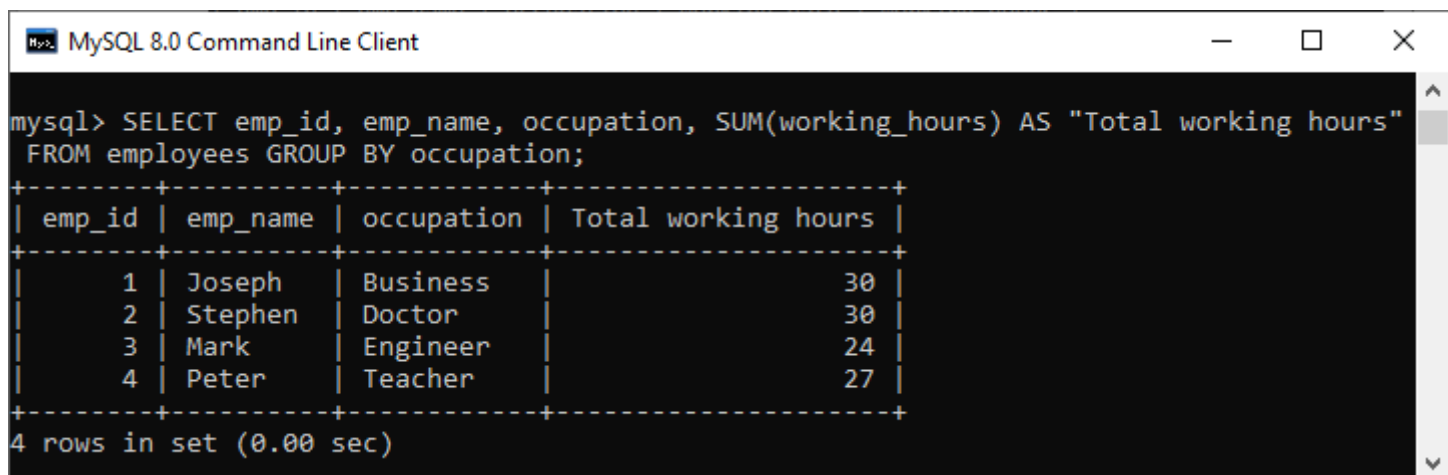
### 3. MySQL sum() function with GROUP BY clause

We can also use the SUM() function with the GROUP BY clause to return the total summed value for each group. For example, this statement calculates the total working hours of each employee by using the SUM() function with the GROUP BY clause, as shown in the following query:

1. mysql> **SELECT** emp\_id, emp\_name, occupation, **SUM**(working\_hours) **AS** "Total working hours" **FROM** employees **GROUP BY** occupation;

#### Output:

Here, we can see that the total working hours of each employee calculates by grouping them based on their occupation.



```
mysql> SELECT emp_id, emp_name, occupation, SUM(working_hours) AS "Total working hours"
FROM employees GROUP BY occupation;
+-----+-----+-----+-----+
| emp_id | emp_name | occupation | Total working hours |
+-----+-----+-----+-----+
| 1      | Joseph   | Business   | 30                  |
| 2      | Stephen  | Doctor     | 30                  |
| 3      | Mark     | Engineer   | 24                  |
| 4      | Peter    | Teacher    | 27                  |
+-----+-----+-----+-----+
4 rows in set (0.00 sec)
```

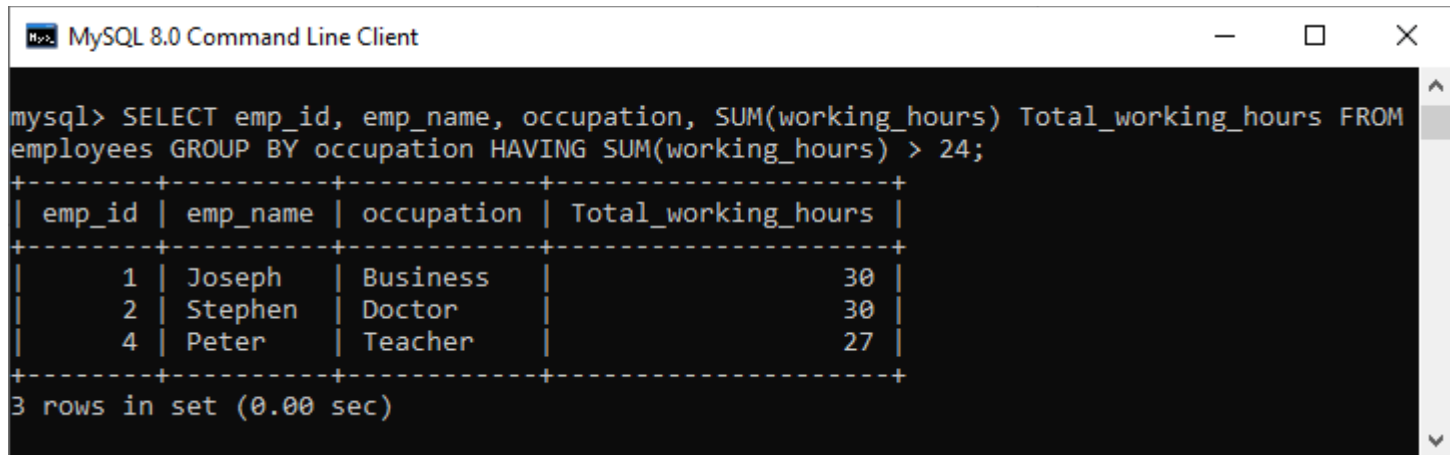
### 4. MySQL sum() function with HAVING clause

The HAVING clause is used to **filter** the group with the sum() function in MySQL. Execute the following statement that calculates the working hours of all employees, grouping them based on their occupation and returns the result whose Total\_working\_hours>24.

1. mysql> **SELECT** emp\_id, emp\_name, occupation,
2. **SUM**(working\_hours) Total\_working\_hours
3. **FROM** employees
4. **GROUP BY** occupation

5. **HAVING SUM**(working\_hours)>24;

Output:



```
mysql> SELECT emp_id, emp_name, occupation, SUM(working_hours) Total_working_hours FROM
employees GROUP BY occupation HAVING SUM(working_hours) > 24;
+-----+-----+-----+-----+
| emp_id | emp_name | occupation | Total_working_hours |
+-----+-----+-----+-----+
| 1      | Joseph  | Business  | 30                  |
| 2      | Stephen | Doctor    | 30                  |
| 4      | Peter   | Teacher   | 27                  |
+-----+-----+-----+-----+
3 rows in set (0.00 sec)
```

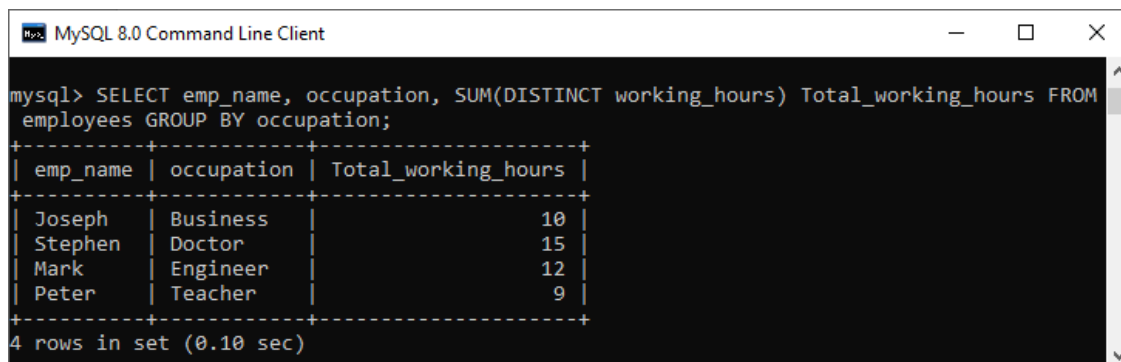
## 5. MySQL sum() function with DISTINCT clause

MySQL uses the DISTINCT keyword to remove the **duplicate** rows from the column name. This clause can also be used with sum() function to return the total summed value of a Unique number of records present in the table.

Execute the following query that removes the duplicate records in the working\_hours column of the employee table and then calculates the sum:

1. mysql> **SELECT** emp\_name, occupation,
2. **SUM**(**DISTINCT** working\_hours) Total\_working\_hours
3. **FROM** employees
4. **GROUP BY** occupation;

Output:



```
mysql> SELECT emp_name, occupation, SUM(DISTINCT working_hours) Total_working_hours FROM
employees GROUP BY occupation;
+-----+-----+-----+
| emp_name | occupation | Total_working_hours |
+-----+-----+-----+
| Joseph   | Business  | 10                  |
| Stephen  | Doctor    | 15                  |
| Mark     | Engineer  | 12                  |
| Peter    | Teacher   | 9                   |
+-----+-----+-----+
4 rows in set (0.10 sec)
```

## MySQL avg() function

The MySQL avg() is an aggregate function used to return the average value of an expression in various records.

### Syntax

The following are the basic syntax an avg() function in MySQL:

1. **SELECT** **AVG**(aggregate\_expression)
2. **FROM** tables

### 3. [WHERE conditions];

#### Parameter explanation

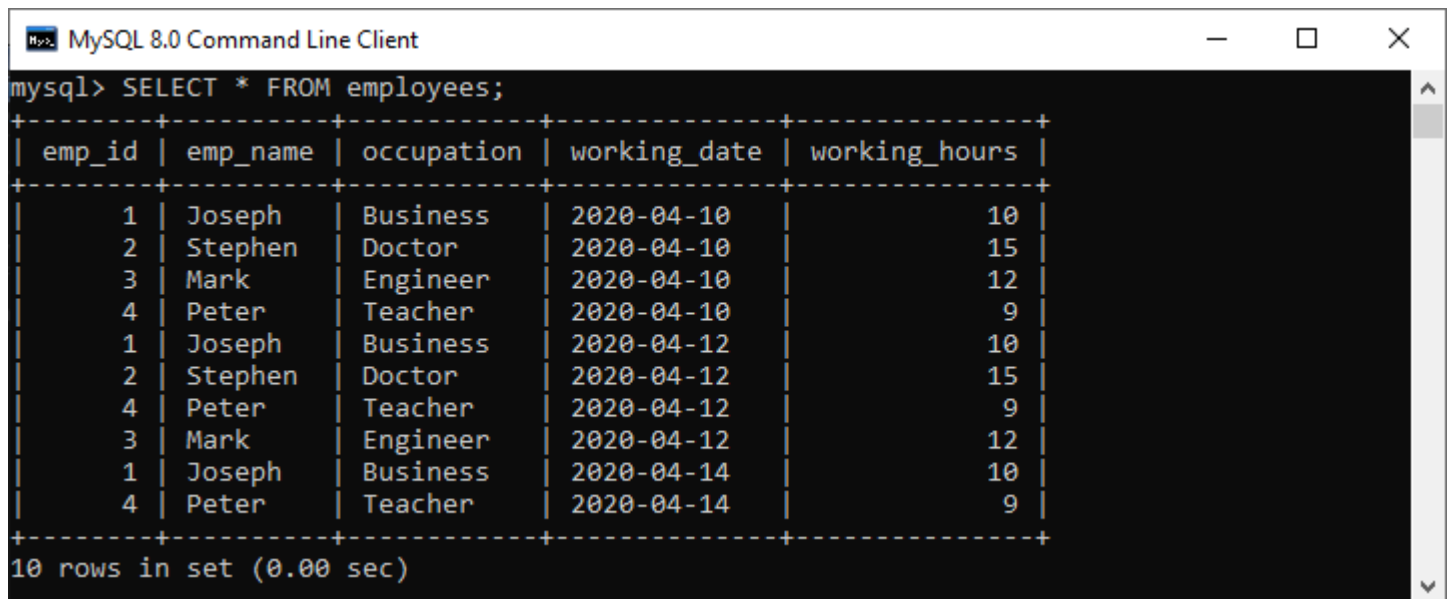
**aggregate\_expression:** It specifies the column or expression that we are going to find the average result.

**table\_name:** It specifies the tables from where we want to retrieve records. There must be at least one table listed in the FROM clause.

**WHERE conditions:** It is optional. It specifies the conditions that must be fulfilled for the records to be selected.

#### MySQL avg() function example

Consider our database has a table named **employees**, having the following data. Now, we are going to understand this function with various examples:



```
mysql> SELECT * FROM employees;
```

emp_id	emp_name	occupation	working_date	working_hours
1	Joseph	Business	2020-04-10	10
2	Stephen	Doctor	2020-04-10	15
3	Mark	Engineer	2020-04-10	12
4	Peter	Teacher	2020-04-10	9
1	Joseph	Business	2020-04-12	10
2	Stephen	Doctor	2020-04-12	15
4	Peter	Teacher	2020-04-12	9
3	Mark	Engineer	2020-04-12	12
1	Joseph	Business	2020-04-14	10
4	Peter	Teacher	2020-04-14	9

```
10 rows in set (0.00 sec)
```

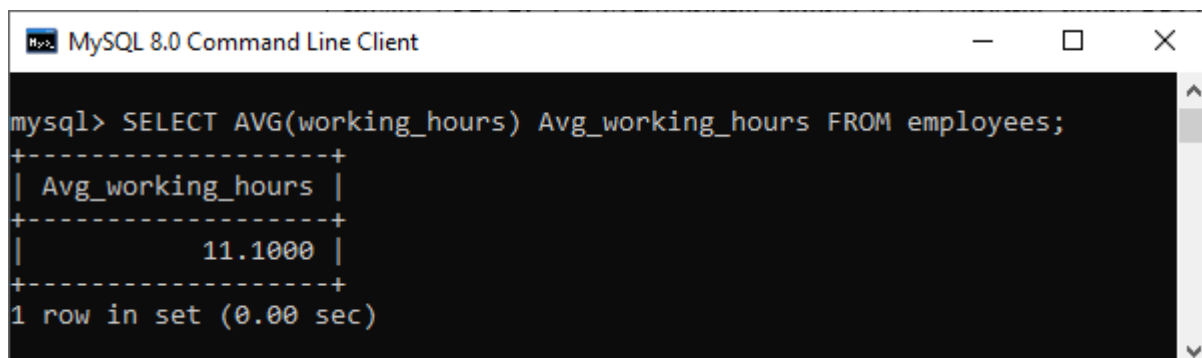
#### 1. Basic Example

Execute the following query that calculates the **average working hours** of all employees in the table:

1. mysql> **SELECT** **AVG**(working\_hours) Avg\_working\_hours **FROM** employees;

**Output:**

We will get the result as below:



```
mysql> SELECT AVG(working_hours) Avg_working_hours FROM employees;
```

Avg_working_hours
11.1000

```
1 row in set (0.00 sec)
```

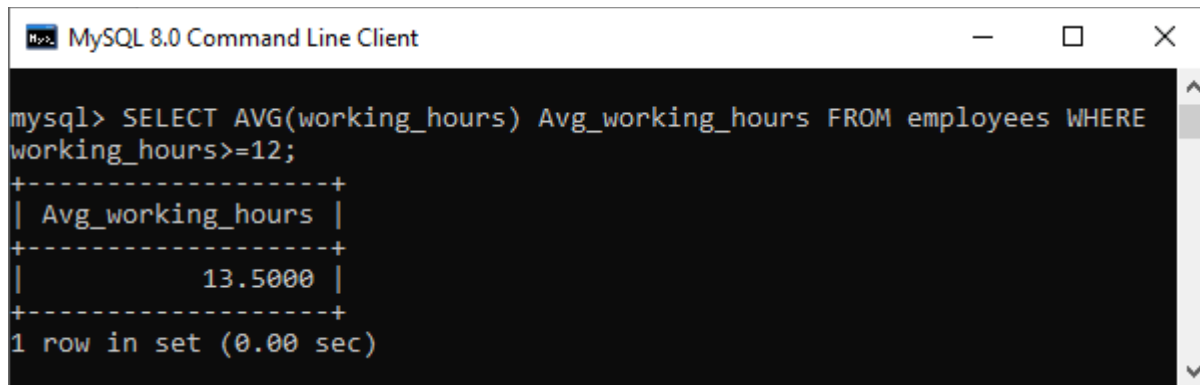
#### 2. MySQL AVG() function with WHERE clause

The WHERE clause specifies the conditions that must be fulfilled for the selected records. Execute the following query to calculate the total average working hours of employees whose **working\_hours** **>= 12**.

1. mysql> **SELECT** **AVG**(working\_hours) Avg\_working\_hours **FROM** employees **WHERE** working\_hours>=12 ;

**Output:**

It will give the following output:



```
mysql> SELECT AVG(working_hours) Avg_working_hours FROM employees WHERE
working_hours>=12;
+-----+
| Avg_working_hours |
+-----+
|          13.5000 |
+-----+
1 row in set (0.00 sec)
```

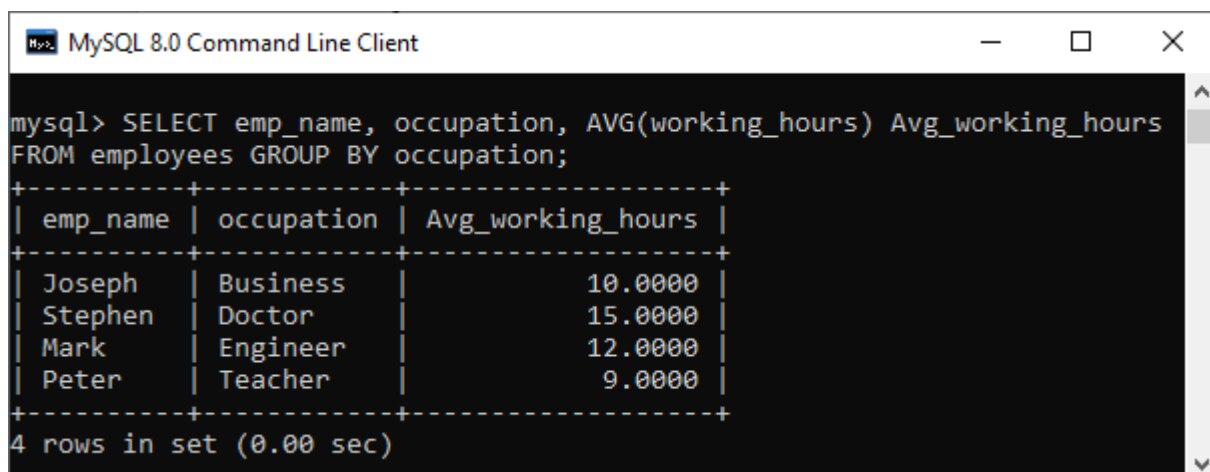
### 3. MySQL AVG() function with GROUP BY clause

The GROUP BY clause is used to return the result for each group by one or more columns. For example, this statement calculates the average working hours of each employee using the AVG() function and then group the result with the GROUP BY clause:

1. mysql> **SELECT** emp\_name, occupation, **AVG**(working\_hours) Avg\_working\_hours **FROM** employees **GROUP BY** occupation;

**Output:**

Here, we can see that the total working hours of each employee calculates by grouping them based on their **occupation**.



```
mysql> SELECT emp_name, occupation, AVG(working_hours) Avg_working_hours
FROM employees GROUP BY occupation;
+-----+-----+-----+
| emp_name | occupation | Avg_working_hours |
+-----+-----+-----+
| Joseph   | Business   |          10.0000 |
| Stephen  | Doctor     |          15.0000 |
| Mark     | Engineer   |          12.0000 |
| Peter    | Teacher    |           9.0000 |
+-----+-----+-----+
4 rows in set (0.00 sec)
```

### 4. MySQL AVG() function with HAVING clause

The HAVING clause is used to **filter** the average values of the groups in MySQL. Execute the following statement that calculates the average working hours of all employees, grouping them based on their occupation and returns the result whose **Avg\_working\_hours**>9.

1. mysql> **SELECT** emp\_name, occupation,

2. **AVG**(working\_hours) Avg\_working\_hours
3. **FROM** employees
4. **GROUP BY** occupation
5. **HAVING AVG**(working\_hours)>9;

**Output:**

```
mysql> SELECT emp_name, occupation, AVG(working_hours) Avg_working_hours FROM
employees GROUP BY occupation HAVING AVG(working_hours)>9;
```

emp_name	occupation	Avg_working_hours
Joseph	Business	10.0000
Stephen	Doctor	15.0000
Mark	Engineer	12.0000

3 rows in set (0.00 sec)

## 5. MySQL AVG() function with DISTINCT clause

MySQL uses the DISTINCT keyword to remove the **duplicate** rows from the column name. This clause is used with this avg() function to return the average value of a unique number of records present in the table.

Execute the following query that removes the duplicate records in the working\_hours column of the employee table and then returns the average value:

1. mysql> **SELECT** emp\_name, occupation,
2. **AVG**(**DISTINCT** working\_hours) Avg\_working\_hours
3. **FROM** employees
4. **GROUP BY** occupation;

**Output:**

```
mysql> SELECT emp_name, occupation, AVG(DISTINCT working_hours) Avg_working_hours FROM
employees GROUP BY occupation;
```

emp_name	occupation	Avg_working_hours
Joseph	Business	10.0000
Stephen	Doctor	15.0000
Mark	Engineer	12.0000
Peter	Teacher	9.0000

4 rows in set (0.00 sec)

## MySQL MIN() Function

The MIN() function in MySQL is used to return the **minimum value** in a set of values from the table. It is an aggregate function that is useful when we need to find the smallest number, selecting the least expensive product, etc.

### Syntax

The following is the basic syntax of MIN() [function in MySQL](#):

1. **SELECT MIN** ( **DISTINCT** aggregate\_expression)
2. **FROM** table\_name(s)
3. [**WHERE** conditions];

### Parameter explanation

This function uses the following parameters:

**aggregate\_expression:** It is the required expression. It specifies the column or expression name from which the minimum value will be returned.

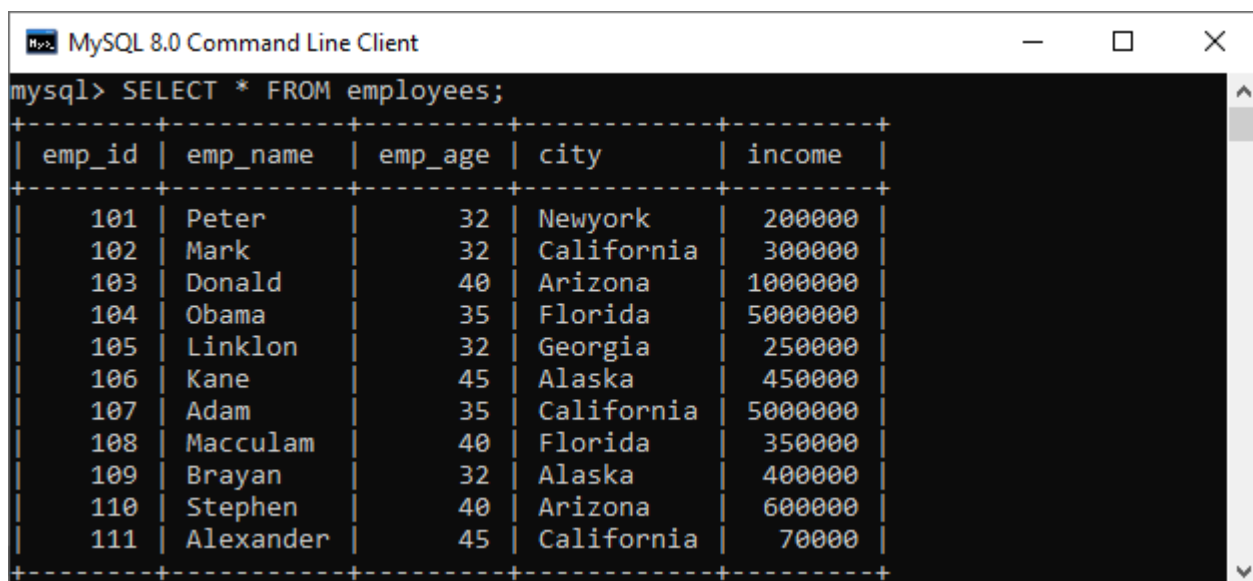
**Table\_name(s):** It specifies the tables from where we want to retrieve records. There must be at least one table listed in the FROM clause.

**WHERE conditions:** It is optional. It specifies the conditions that must be fulfilled for the records to be selected.

**DISTINCT:** It allows us to return the minimum of the distinct values in the expression. However, it does not affect the MIN() function and produces the same result without using this keyword.

### MySQL MIN() Function Example

Let us understand how MIN function works in [MySQL](#) with the help of various examples. Consider our database has a table named "**employees**" that contains the following data.



```
mysql> SELECT * FROM employees;
```

emp_id	emp_name	emp_age	city	income
101	Peter	32	Newyork	200000
102	Mark	32	California	300000
103	Donald	40	Arizona	1000000
104	Obama	35	Florida	5000000
105	Linklon	32	Georgia	250000
106	Kane	45	Alaska	450000
107	Adam	35	California	5000000
108	Macculam	40	Florida	350000
109	Brayan	32	Alaska	400000
110	Stephen	40	Arizona	600000
111	Alexander	45	California	70000

#### 1. Basic Example

Execute the following query that uses the MIN function to find the **minimum income** of the employee available in the table:

1. mysql> **SELECT MIN**(income) **AS** Minimum\_Income **FROM** employees;

### Output

The above query produces the result of minimum values in all rows. After execution, we will get the output as below:



```
MySQL 8.0 Command Line Client
mysql> SELECT MIN(income) AS Minimum_Income FROM employees;
+-----+
| Minimum_Income |
+-----+
|          70000 |
+-----+
1 row in set (0.00 sec)
```

## 2. MySQL MIN() Function with WHERE Clause

The **WHERE clause** allows us to filter the result from the selected records. The following statement finds the minimum income in all rows from the employee table and WHERE clause specifies all those rows whose **emp\_age column** is greater than or equal to 32 and less than or equal to 40.

1. mysql> **SELECT MIN**(income) **AS** Minimum\_Income
2. **FROM** employees
3. **WHERE** emp\_age >= 32 **AND** emp\_age <= 40;

### Output

The above statement will get the output as below:

```
MySQL 8.0 Command Line Client
mysql> SELECT MIN(income) AS Minimum_Income
-> FROM employees
-> WHERE emp_age>=32 AND emp_age<=40;
+-----+
| Minimum_Income |
+-----+
|          200000 |
+-----+
```

## 3. MySQL MIN() Function with GROUP BY Clause

The GROUP BY clause allows us to collect data from multiple rows and group it based on one or more columns. For example, the following statement uses the MIN() function with the GROUP BY clause to find the minimum income in all rows from the employee table for each emp\_age group.

1. mysql> **SELECT** emp\_age, **MIN**(income) **AS** Minimum\_Income
2. **FROM** employees
3. **GROUP BY** emp\_age;

### Output

After the successful execution, we can see that the income of each employee returns by grouping them based on their age:

```
MySQL 8.0 Command Line Client
mysql> SELECT emp_age, MIN(income) AS Minimum_Income
-> FROM employees
-> GROUP BY emp_age;
+-----+-----+
| emp_age | Minimum_Income |
+-----+-----+
| 32      | 200000          |
| 40      | 350000          |
| 35      | 500000          |
| 45      | 70000           |
+-----+-----+
4 rows in set (0.00 sec)
```

#### 4. MySQL MIN() Function with HAVING Clause

The **HAVING clause** is always used with the GROUP BY clause to filter the records from the table. For example, the below statement returns the minimum income of all employees, grouping them based on their city and returns the result whose MIN(income)>150000.

1. mysql> **SELECT** city, **MIN**(income) **AS** Minimum\_Income
2. **FROM** employees
3. **GROUP BY** city
4. **HAVING MIN**(income) > 150000;

#### Output

This statement will return the output as below:

```
MySQL 8.0 Command Line Client
mysql> SELECT city, MIN(income) AS Minimum_Income
-> FROM employees
-> GROUP BY city
-> HAVING MIN(income)>150000;
+-----+-----+
| city   | Minimum_Income |
+-----+-----+
| Newyork | 200000          |
| Arizona | 600000          |
| Florida | 350000          |
| Georgia | 250000          |
| Alaska  | 400000          |
+-----+-----+
```

#### 5. MySQL MIN() Function with DISTINCT Clause

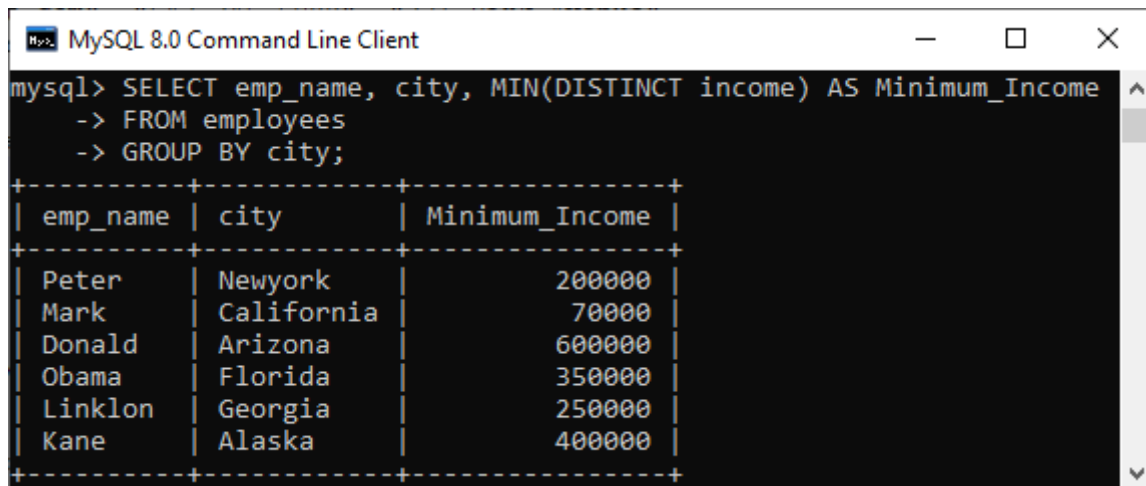
MySQL uses the **DISTINCT** keyword to remove the duplicate rows from the column name. We can also use this clause with MIN() function to return the minimum income value of a unique number of records present in the table.

Execute the following query that removes the duplicate records in the income column of the employee table, group by city, and then returns the minimum value:

1. mysql> **SELECT** emp\_name, city, **MIN**(**DISTINCT** income) **AS** Minimum\_Income
2. **FROM** employees
3. **GROUP BY** city;

## Output

This statement will give the output as below:



```
mysql> SELECT emp_name, city, MIN(DISTINCT income) AS Minimum_Income
-> FROM employees
-> GROUP BY city;
```

emp_name	city	Minimum_Income
Peter	Newyork	200000
Mark	California	70000
Donald	Arizona	600000
Obama	Florida	350000
Linklon	Georgia	250000
Kane	Alaska	400000

## MySQL MAX() Function

The MySQL MAX() function is used to return the maximum value in a set of values of an expression. This aggregate function is useful when we need to find the maximum number, selecting the most expensive product, or getting the largest payment to the customer from your table.

### Syntax

The following is the basic syntax of MAX() function in MySQL:

1. **SELECT MAX(DISTINCT** aggregate\_expression)
2. **FROM** table\_name(s)
3. [**WHERE** conditions];

### Parameter Explanation

This function uses the following parameters:

**aggregate\_expression:** It is the required expression. It specifies the column, expression, or formula from which the maximum value will be returned.

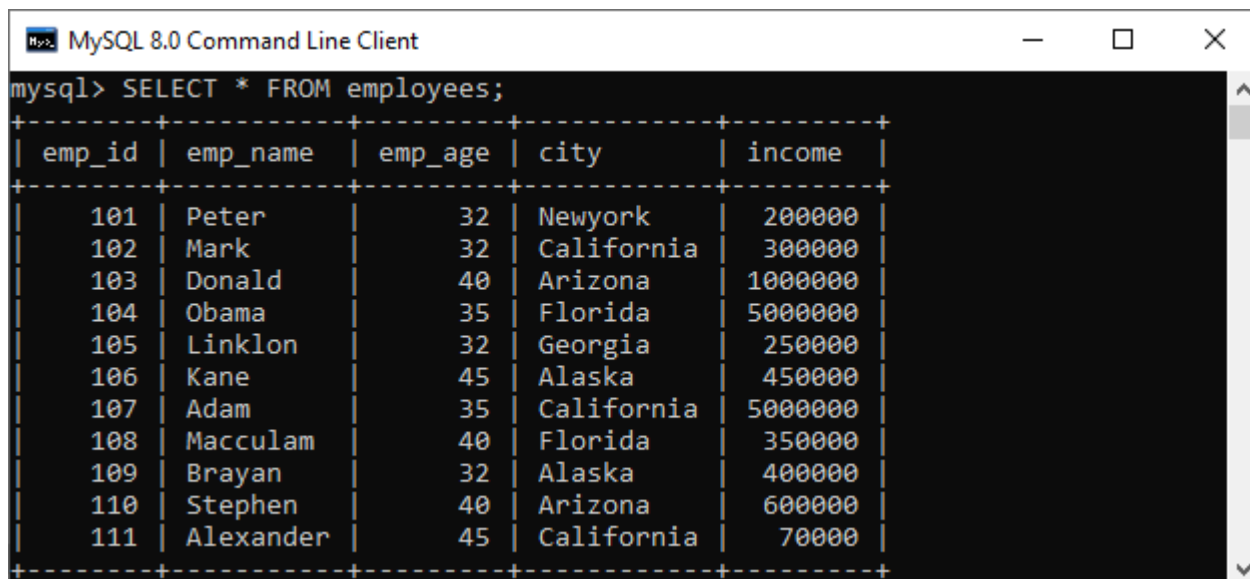
**table\_name(s):** It specifies the tables from where we want to retrieve records. There must be at least one table listed in the FROM clause.

**WHERE conditions:** It is optional. It specifies the conditions that must be fulfilled for the records to be selected.

**DISTINCT:** It allows us to return the maximum of the distinct values in the expression. However, it does not affect the MAX() function and produces the same result without using this keyword.

### MySQL MAX() Function Example

Let us understand how the MAX function works in [MySQL](#) with the help of various examples. Consider our database has a table named "**employees**" that contains the following data.



```
mysql> SELECT * FROM employees;
```

emp_id	emp_name	emp_age	city	income
101	Peter	32	Newyork	200000
102	Mark	32	California	300000
103	Donald	40	Arizona	1000000
104	Obama	35	Florida	5000000
105	Linklon	32	Georgia	250000
106	Kane	45	Alaska	450000
107	Adam	35	California	5000000
108	Macculam	40	Florida	350000
109	Brayan	32	Alaska	400000
110	Stephen	40	Arizona	600000
111	Alexander	45	California	70000

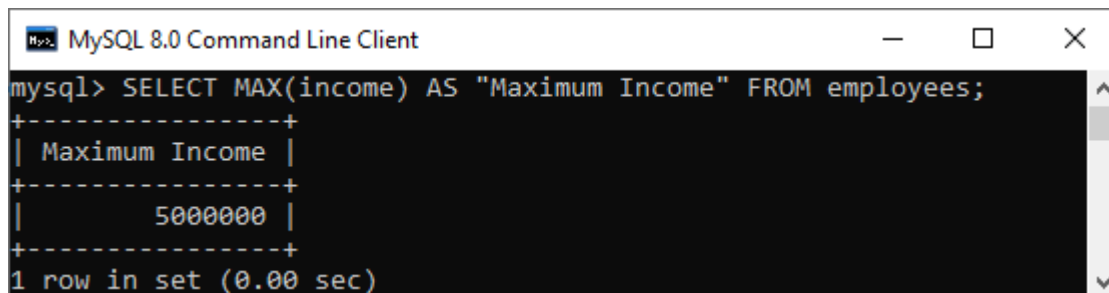
## 1. Basic Example

Execute the following query that uses the MAX function to find the **maximum income** of the employee available in the table:

1. mysql> **SELECT MAX**(income) **AS** "Maximum Income" **FROM** employees;

### Output

The above query produces the result of maximum values in all rows. After execution, we will get the output as below:



```
mysql> SELECT MAX(income) AS "Maximum Income" FROM employees;
```

Maximum Income
5000000

1 row in set (0.00 sec)

## 2. MySQL MAX() Function with WHERE Clause

The [WHERE clause](#) allows us to filter the result from the selected records. The following statement finds the maximum income in all rows from the employee table. The WHERE clause specifies all those rows whose **emp\_age column** is greater than 35.

1. mysql> **SELECT MAX**(income) **AS** "Maximum\_Income"
2. **FROM** employees
3. **WHERE** emp\_age > 35;

### Output

The above statement will get the output as below:

```
MySQL 8.0 Command Line Client
mysql> SELECT MAX(income) AS "Maximum_Income"
-> FROM employees
-> WHERE emp_age > 35;
+-----+
| Maximum_Income |
+-----+
|          1000000 |
+-----+
```

### 3. MySQL MAX() Function with GROUP BY Clause

The [GROUP BY clause](#) allows us to collect data from multiple rows and group it based on one or more columns. For example, the following statement uses the MAX() function with the GROUP BY clause to find the maximum income in all rows from the employee table for each emp\_age group.

1. mysql> **SELECT** emp\_age, **MAX**(income) **AS** "Maximum Income"
2. **FROM** employees
3. **GROUP BY** emp\_age;

#### Output

After the successful execution, we can see that the maximum income of the employee returns by grouping them based on their age:

```
MySQL 8.0 Command Line Client
mysql> SELECT emp_age, MAX(income) AS "Maximum Income"
-> FROM employees
-> GROUP BY emp_age;
+-----+-----+
| emp_age | Maximum Income |
+-----+-----+
|      32 |          400000 |
|      40 |         1000000 |
|      35 |         5000000 |
|      45 |          450000 |
+-----+-----+
```

### 3. MySQL MAX() Function with HAVING Clause

The [HAVING clause](#) is always used with the GROUP BY clause to filter the records from the table. For example, the following statement returns the maximum income among all employees, grouping them based on their city and returns the result whose MAX(income) >= 200000.

1. mysql> **SELECT** city, **MAX**(income) **AS** "Maximum Income"
2. **FROM** employees
3. **GROUP BY** city
4. **HAVING MAX**(income) >= 200000;

#### Output

This statement will return the output as below:

```
MySQL 8.0 Command Line Client
mysql> SELECT city, MAX(income) AS "Maximum Income"
-> FROM employees
-> GROUP BY city
-> HAVING MAX(income) >= 200000;
+-----+-----+
| city      | Maximum Income |
+-----+-----+
| Newyork   | 200000         |
| California| 5000000        |
| Arizona   | 1000000        |
| Florida   | 5000000        |
| Georgia   | 250000         |
| Alaska    | 450000         |
+-----+-----+
```

## 5. MySQL MAX() Function with DISTINCT Clause

MySQL uses the **DISTINCT** keyword to remove the duplicate rows from the column name. We can also use this clause with MAX() function to return the maximum income value of a unique number of records present in the table.

Execute the following query that removes the duplicate records in the employee table's income column, group by city, and then returns the maximum value:

1. mysql> **SELECT** city, **MAX(DISTINCT income)** AS "Maximum Income"
2. **FROM** employees
3. **GROUP BY** city;

### Output

This statement will give the output as below:

```
MySQL 8.0 Command Line Client
mysql> SELECT city, MAX(DISTINCT income) AS "Maximum Income"
-> FROM employees
-> GROUP BY city;
+-----+-----+
| city      | Maximum Income |
+-----+-----+
| Newyork   | 200000         |
| California| 5000000        |
| Arizona   | 1000000        |
| Florida   | 5000000        |
| Georgia   | 250000         |
| Alaska    | 450000         |
+-----+-----+
```

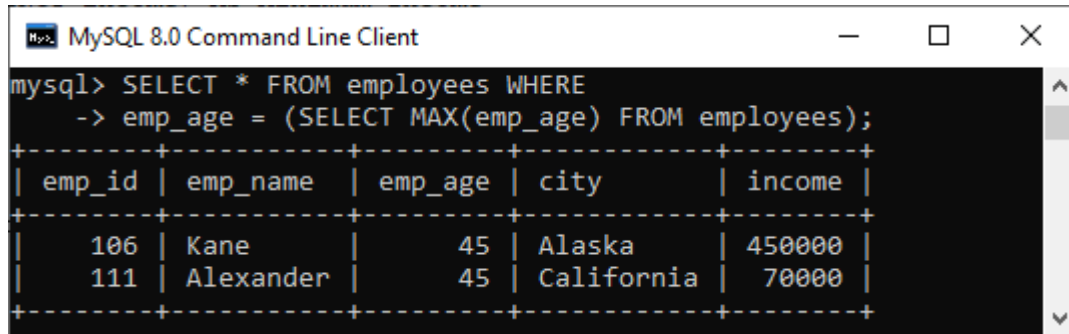
## 6. MySQL MAX() Function in Subquery Example

Sometimes it is required to use the subquery for returning the maximum value in the table. In that case, we use the following query:

1. mysql> **SELECT \* FROM** employees **WHERE**
2. emp\_age = (**SELECT MAX(emp\_age) FROM** employees);

The subquery first finds the maximum age of employees from the table. Then, the main query (outer query) returns the result of age being equal to the maximum age returned from the subquery and other information.

## Output



```
mysql> SELECT * FROM employees WHERE  
-> emp_age = (SELECT MAX(emp_age) FROM employees);
```

emp_id	emp_name	emp_age	city	income
106	Kane	45	Alaska	450000
111	Alexander	45	California	70000

### MySQL first function

The MySQL first function is used to return the first value of the selected column. Here, we use limit clause to select first record or more.

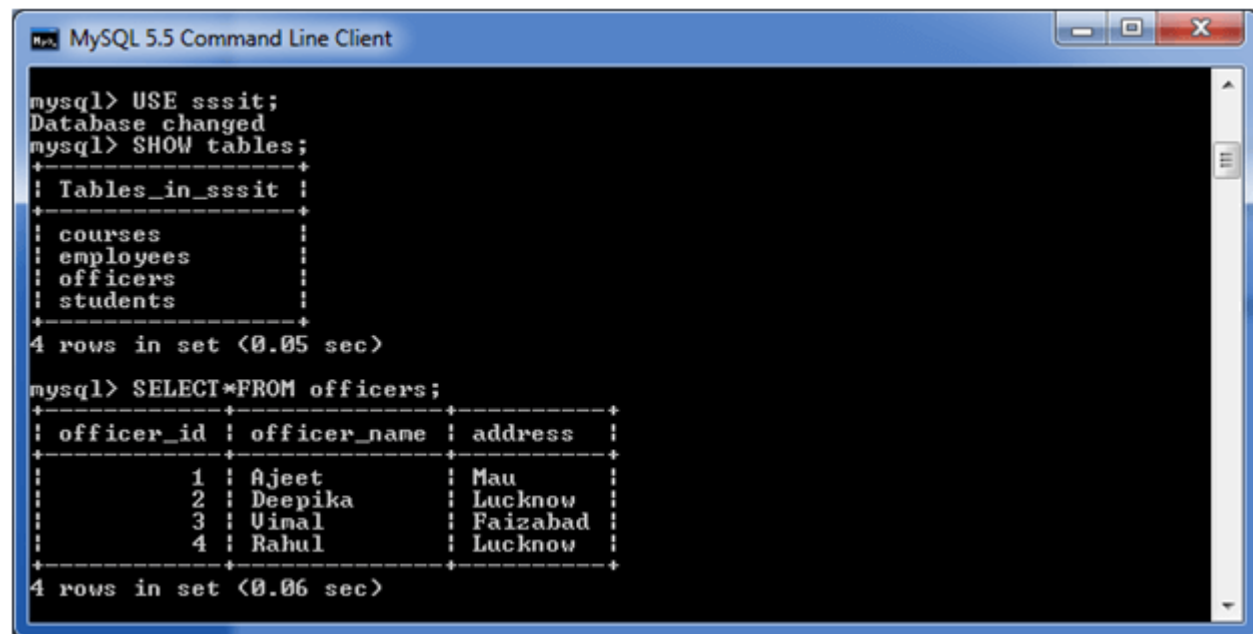
#### Syntax:

1. **SELECT** column\_name
2. **FROM** table\_name
3. **LIMIT** 1;

### MySQL first function example

#### To SELECT FIRST element:

Consider a table named "officers", having the following data.



```
mysql> USE sssit;  
Database changed  
mysql> SHOW tables;  
+-----+  
| Tables_in_sssit |  
+-----+  
| courses          |  
| employees        |  
| officers          |  
| students         |  
+-----+  
4 rows in set (0.05 sec)
```

```
mysql> SELECT * FROM officers;
```

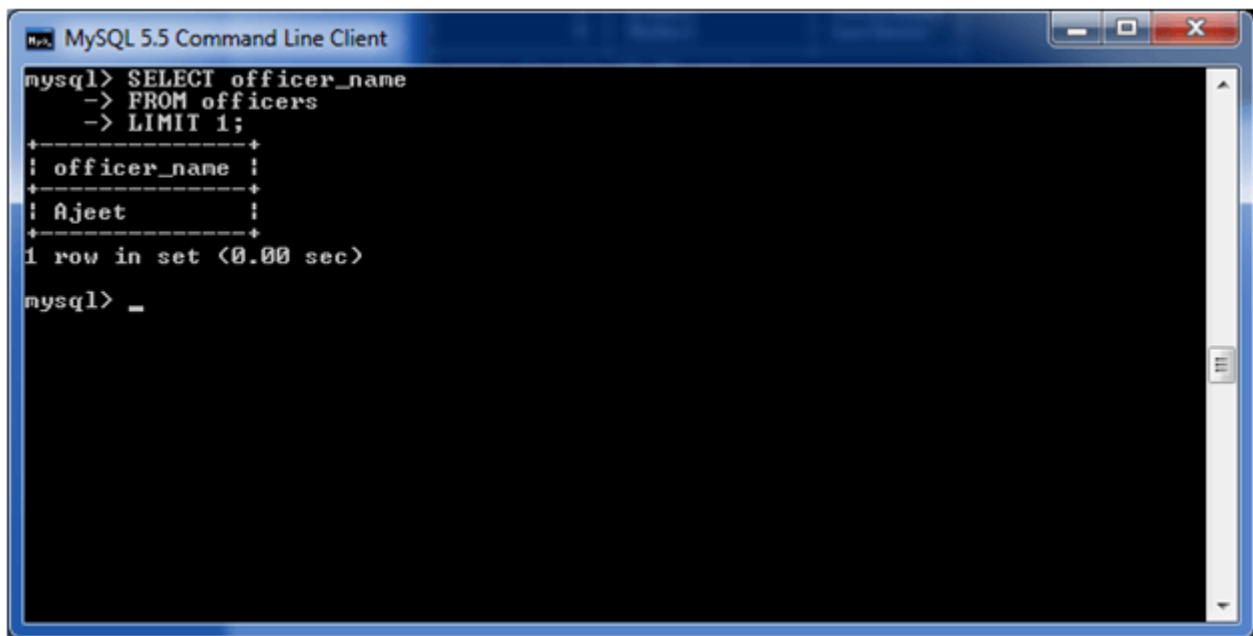
officer_id	officer_name	address
1	Ajeet	Mau
2	Deepika	Lucknow
3	Uimal	Faizabad
4	Rahul	Lucknow

```
4 rows in set (0.06 sec)
```

Execute the following query:

1. **SELECT** officer\_name
2. **FROM** officers
3. **LIMIT** 1;

#### Output:



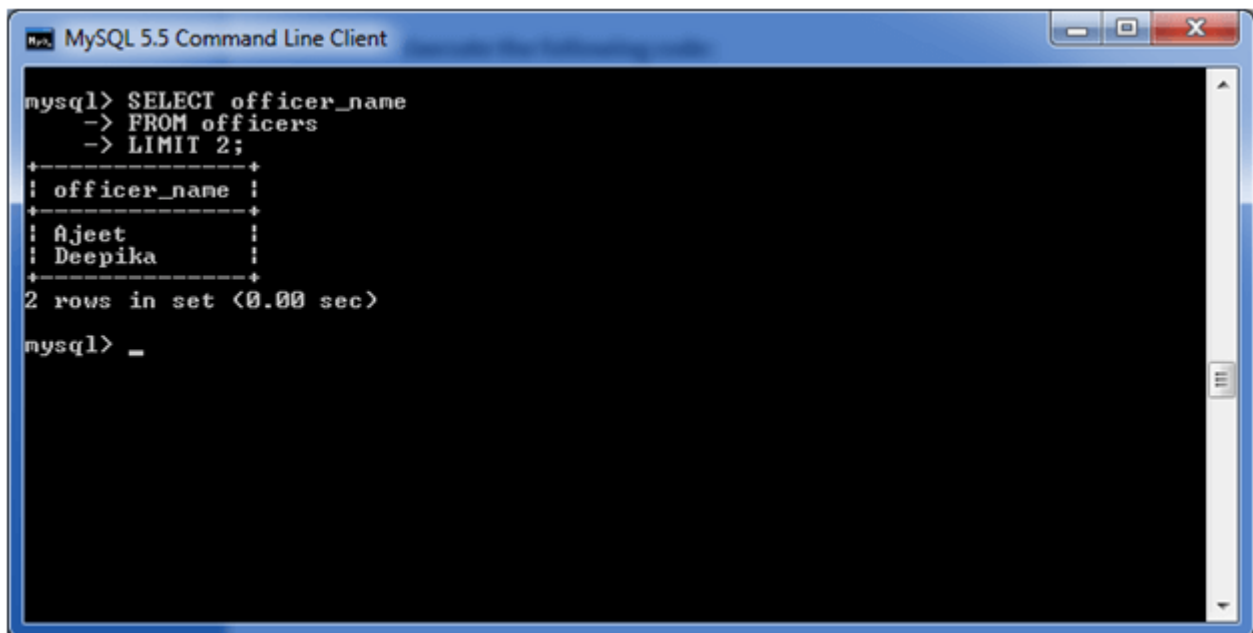
```
mysql> SELECT officer_name
-> FROM officers
-> LIMIT 1;
+-----+
| officer_name |
+-----+
| Ajeet        |
+-----+
1 row in set (0.00 sec)

mysql> _
```

To SELECT FIRST two records

1. **SELECT** officer\_name
2. **FROM** officers
3. **LIMIT** 2;

Output:



```
mysql> SELECT officer_name
-> FROM officers
-> LIMIT 2;
+-----+
| officer_name |
+-----+
| Ajeet        |
| Deepika      |
+-----+
2 rows in set (0.00 sec)

mysql> _
```

MySQL last function

MySQL last function is used to return the last value of the selected column.

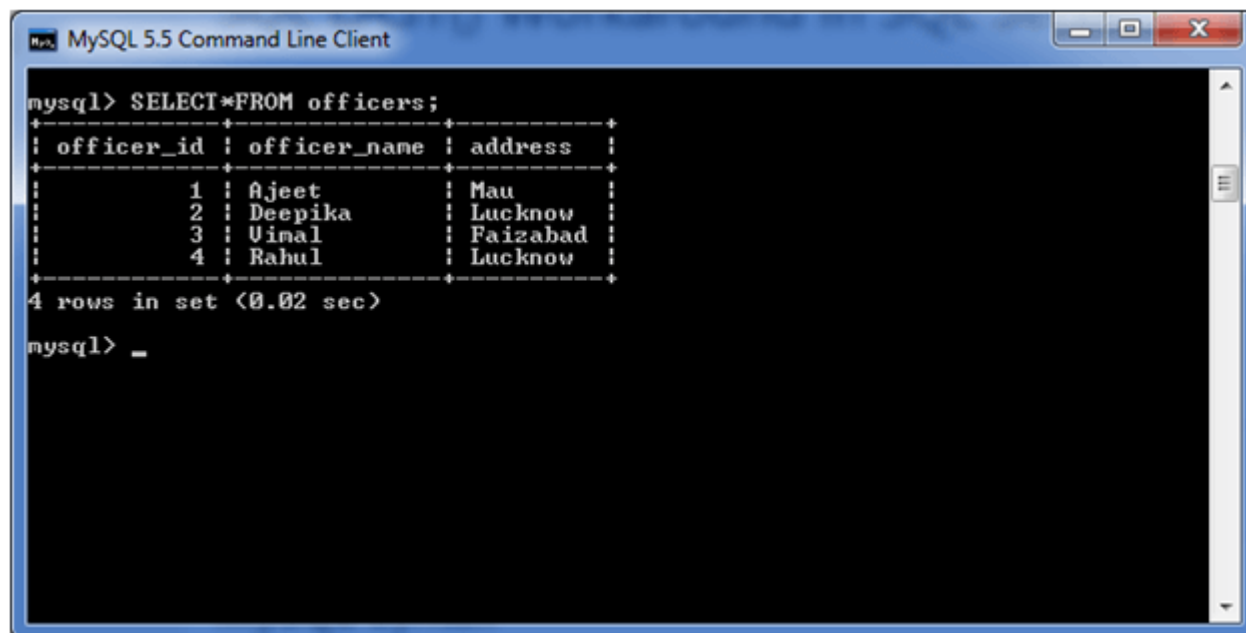
Syntax:

1. **SELECT** column\_name
2. **FROM** table\_name
3. **ORDER BY** column\_name **DESC**
4. **LIMIT** 1;



## MySQL last function example

Consider a table "officers" having the following data.



```
mysql> SELECT * FROM officers;
+-----+-----+-----+
| officer_id | officer_name | address |
+-----+-----+-----+
| 1 | Ajeet | Mau |
| 2 | Deepika | Lucknow |
| 3 | Vimal | Faizabad |
| 4 | Rahul | Lucknow |
+-----+-----+-----+
4 rows in set (0.02 sec)

mysql> _
```

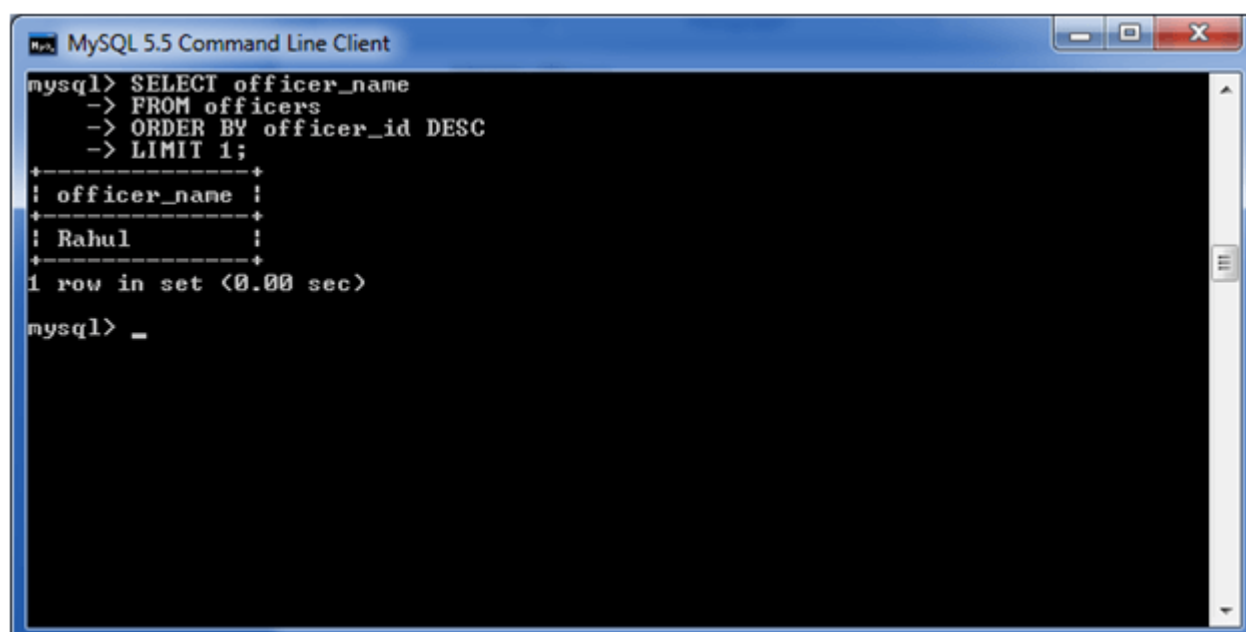
Execute the following query:

Keep Watching

1. **SELECT** officer\_name
2. **FROM** officers
3. **ORDER BY** officer\_id **DESC**
4. **LIMIT 1;**

This query will return the last officer\_name ordering by officer\_id.

Output:



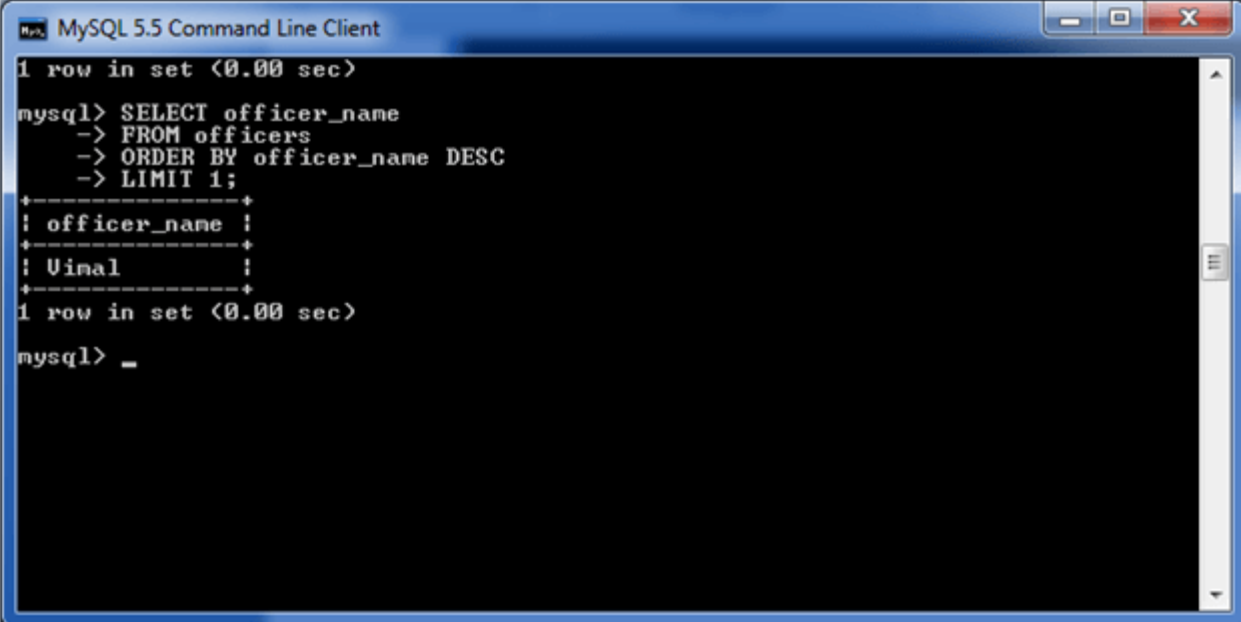
```
mysql> SELECT officer_name
-> FROM officers
-> ORDER BY officer_id DESC
-> LIMIT 1;
+-----+
| officer_name |
+-----+
| Rahul |
+-----+
1 row in set (0.00 sec)

mysql> _
```

Return the last officer\_name ordering by officer\_name:

1. **SELECT** officer\_name
2. **FROM** officers
3. **ORDER BY** officer\_name **DESC**
4. **LIMIT 1;**

Output:



```
MySQL 5.5 Command Line Client
1 row in set (0.00 sec)
mysql> SELECT officer_name
-> FROM officers
-> ORDER BY officer_name DESC
-> LIMIT 1;
+-----+
| officer_name |
+-----+
| Uinal        |
+-----+
1 row in set (0.00 sec)
mysql> _
```