

In this module we will learn the following topics in SQL

1. Order By Clause
2. Types of SQL Functions – Scalar and Aggregate
3. Aggregate Functions – Count(), Sum(), Avg(), Min() and Max()
4. Group By Clause
5. Having Clause
6. Joins – EQUI and NATURAL

ORDER BY Clause

The ORDER BY Clause is used to sort the records in ascending or descending order.

Syntax:

```
SELECT expressions  
FROM tables  
[WHERE conditions]  
ORDER BY expression [ ASC | DESC ];
```

expressions: It specifies the columns that you want to retrieve.

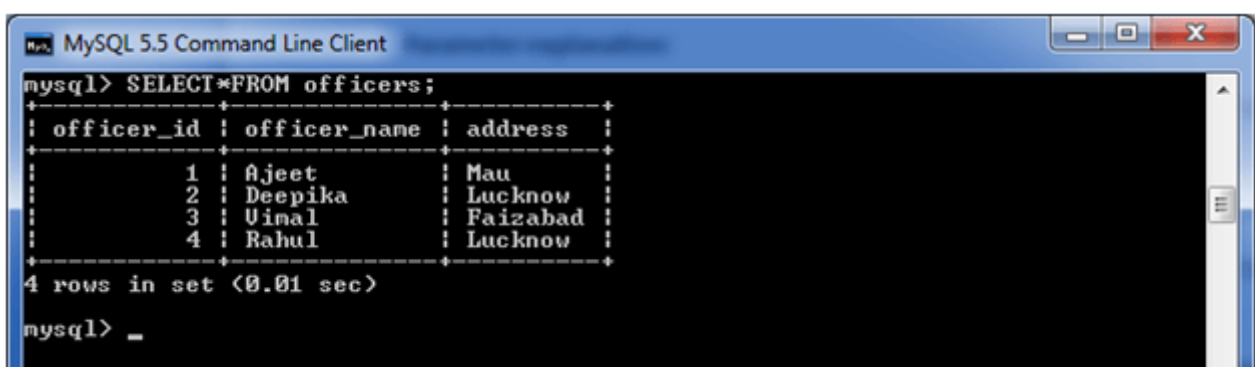
tables: It specifies the tables, from where you want to retrieve records. There must be at least one table listed in the FROM clause.

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

ASC: It is optional. It sorts the result set in ascending order by expression (default, if no modifier is provided).

DESC: It is also optional. It sorts the result set in descending order by expression.

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



```
mysql> SELECT * FROM officers;  
+-----+-----+-----+  
| officer_id | officer_name | address |  
+-----+-----+-----+  
| 1 | Ajeet | Mau |  
| 2 | Deepika | Lucknow |  
| 3 | Uinal | Faizabad |  
| 4 | Rahul | Lucknow |  
+-----+-----+-----+  
4 rows in set (0.01 sec)  
mysql> _
```

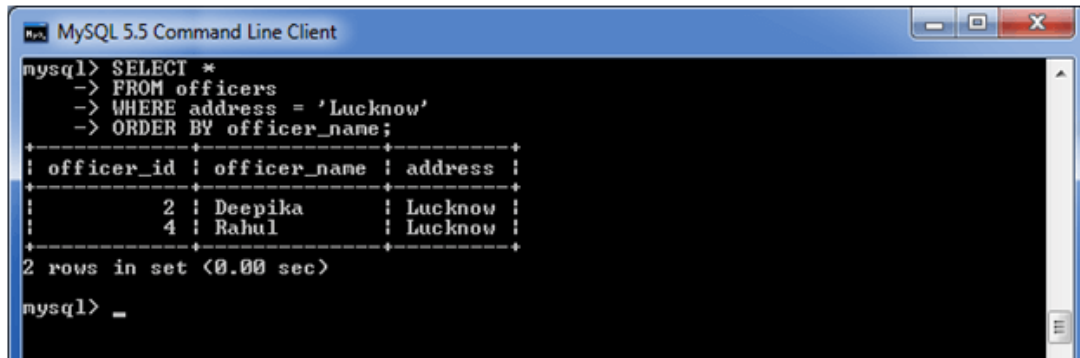
1. ORDER BY: without using ASC/DESC attribute

If you use ORDER BY clause without specifying the ASC and DESC modifier then by default you will get the result in ascending order.

Execute the following query:

```
SELECT *  
FROM officers  
WHERE address = 'Lucknow'  
ORDER BY officer_name;
```

Output:



```
mysql> SELECT *  
-> FROM officers  
-> WHERE address = 'Lucknow'  
-> ORDER BY officer_name;  
+-----+-----+-----+  
| officer_id | officer_name | address |  
+-----+-----+-----+  
|          2 | Deepika      | Lucknow |  
|          4 | Rahul        | Lucknow |  
+-----+-----+-----+  
2 rows in set (0.00 sec)  
mysql> _
```

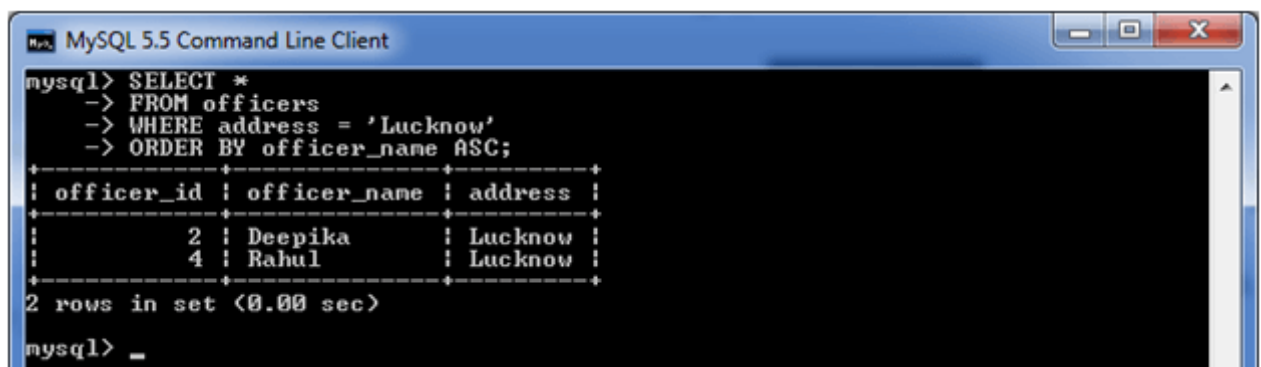
2. ORDER BY: with ASC attribute

Let's take an example to retrieve the data in ascending order.

Execute the following query:

```
SELECT *  
FROM officers  
WHERE address = 'Lucknow'  
ORDER BY officer_name ASC;
```

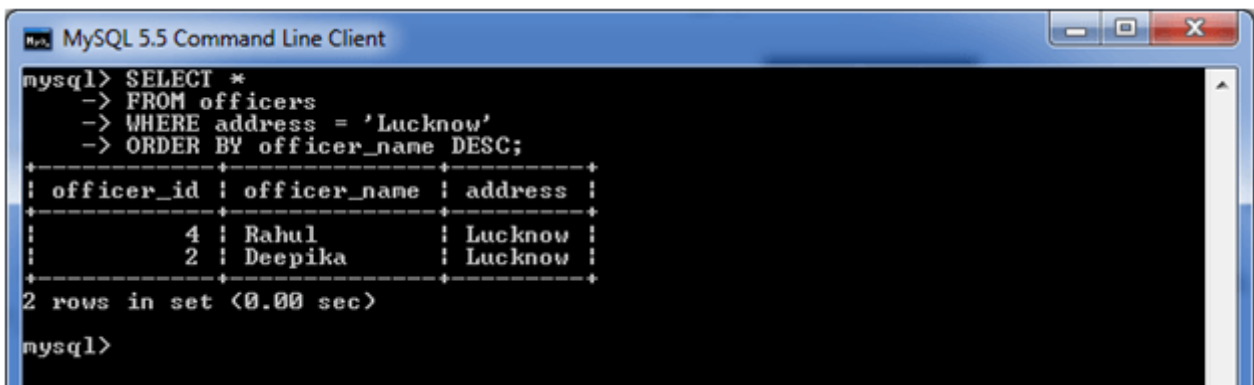
Output:



```
mysql> SELECT *  
-> FROM officers  
-> WHERE address = 'Lucknow'  
-> ORDER BY officer_name ASC;  
+-----+-----+-----+  
| officer_id | officer_name | address |  
+-----+-----+-----+  
|          2 | Deepika      | Lucknow |  
|          4 | Rahul        | Lucknow |  
+-----+-----+-----+  
2 rows in set (0.00 sec)  
mysql> _
```

3. ORDER BY: with DESC attribute

```
SELECT *  
FROM officers  
WHERE address = 'Lucknow'  
ORDER BY officer_name DESC;
```



```
mysql> SELECT *
-> FROM officers
-> WHERE address = 'Lucknow'
-> ORDER BY officer_name DESC;
+-----+-----+-----+
| officer_id | officer_name | address |
+-----+-----+-----+
|          4 | Rahul        | Lucknow |
|          2 | Deepika      | Lucknow |
+-----+-----+-----+
2 rows in set (0.00 sec)

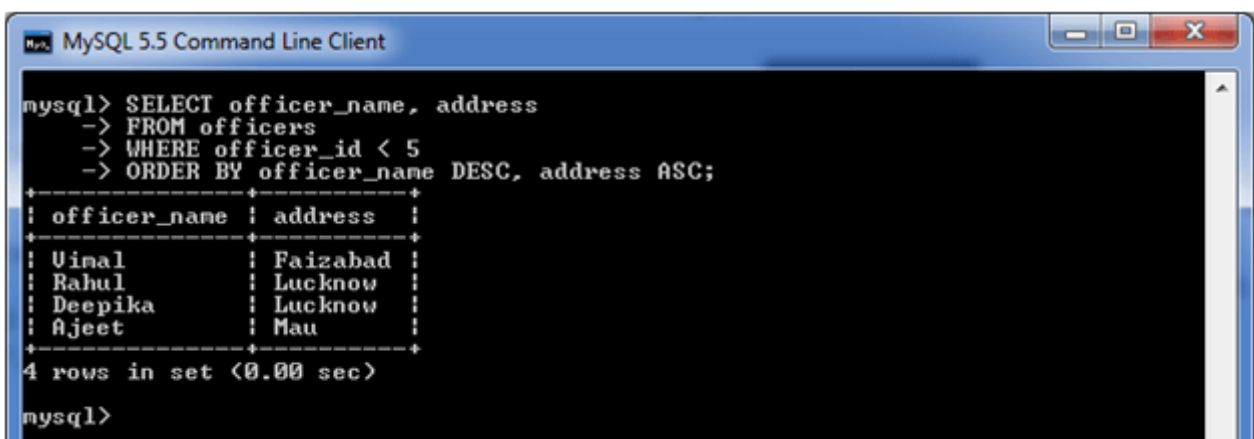
mysql>
```

4. ORDER BY: using both ASC and DESC attributes

Execute the following query:

```
SELECT officer_name, address
FROM officers
WHERE officer_id < 5
ORDER BY officer_name DESC, address ASC;
```

Output:



```
mysql> SELECT officer_name, address
-> FROM officers
-> WHERE officer_id < 5
-> ORDER BY officer_name DESC, address ASC;
+-----+-----+
| officer_name | address |
+-----+-----+
| Uinal        | Faizabad |
| Rahul        | Lucknow  |
| Deepika      | Lucknow  |
| Ajeet        | Mau      |
+-----+-----+
4 rows in set (0.00 sec)

mysql>
```

Types of SQL Functions:

1. Single Row(or scalar) Functions: This type of functions work with a single row at a time. A single row function returns a result for every row of a queried table.
2. Multiple Row(or Group or Aggregate)Functions: This type of functions are the group functions that works with data of multiple rows at a time and returns a single result for that group.

Aggregate Functions

Aggregate functions work upon group of rows rather than single rows. That is why, these functions are sometimes called as multiple row functions. Some of aggregate functions are:

- 1) Count() 2)Sum() 3)Avg() 4)Min() 5)Max()

Consider a table named "employees" that contains the following data for all the types of aggregate functions given above.

```
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)
```

1) Count() Function

Count() function is used to return the number of rows in a given column or expression.

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

COUNT ({*[DISTINCT | ALL]expr})

- Returns the number of rows in the query.
- If you specify argument expr, this function returns rows where expr is not null. You can count either all rows, or only distinct values of expr.
- If you specify the asterisk (*), this function returns all rows, including duplicates and nulls.

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

Example1

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

```
mysql> SELECT COUNT(emp_name) FROM employees;
```

Output:

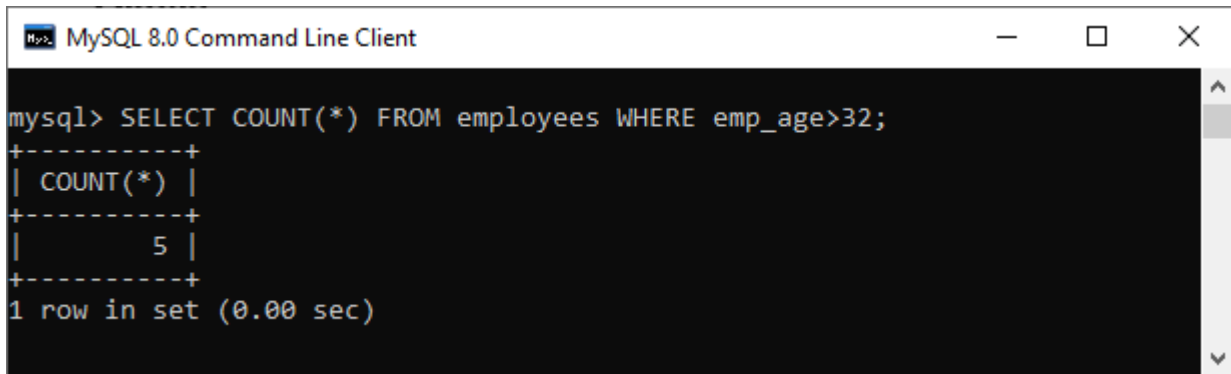
```
mysql> SELECT COUNT(emp_name) FROM employees;
+-----+
| COUNT(emp_name) |
+-----+
| 8                |
+-----+
1 row in set (0.00 sec)
```

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:

```
mysql> SELECT COUNT(*) FROM employees WHERE emp_age>32;
```

Output:



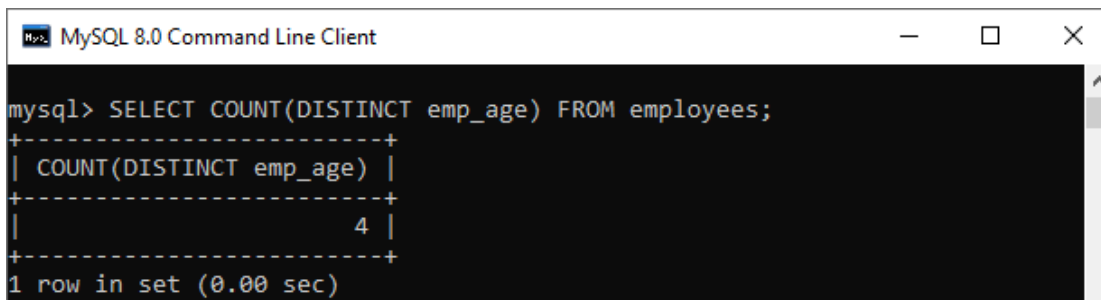
```
mysql> SELECT COUNT(*) FROM employees WHERE emp_age>32;
+-----+
| COUNT(*) |
+-----+
|         5 |
+-----+
1 row in set (0.00 sec)
```

Example3

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

```
mysql> SELECT COUNT(DISTINCT emp_age) FROM employees;
```

Output:



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

2) Sum() function

The 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.

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

SUM([DISTINCT |ALL]n)

- Returns the sum of values of n

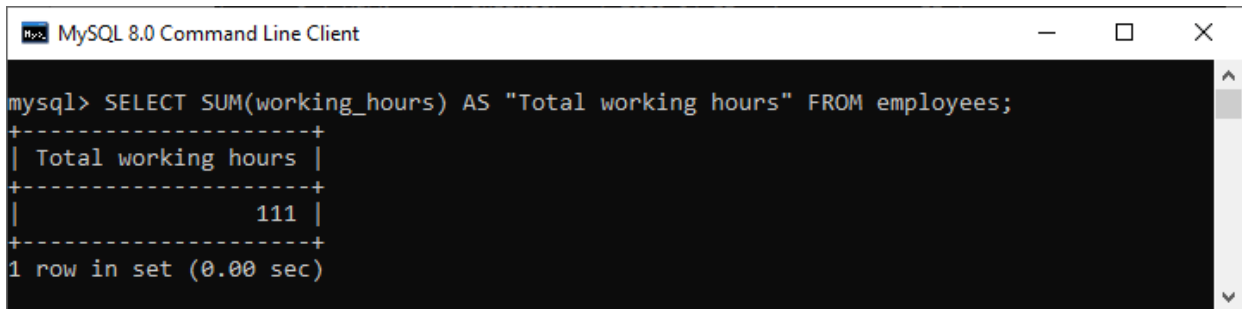
Consider our database has a table named **employees**, having the following data.

1. Basic Example

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

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

Output:



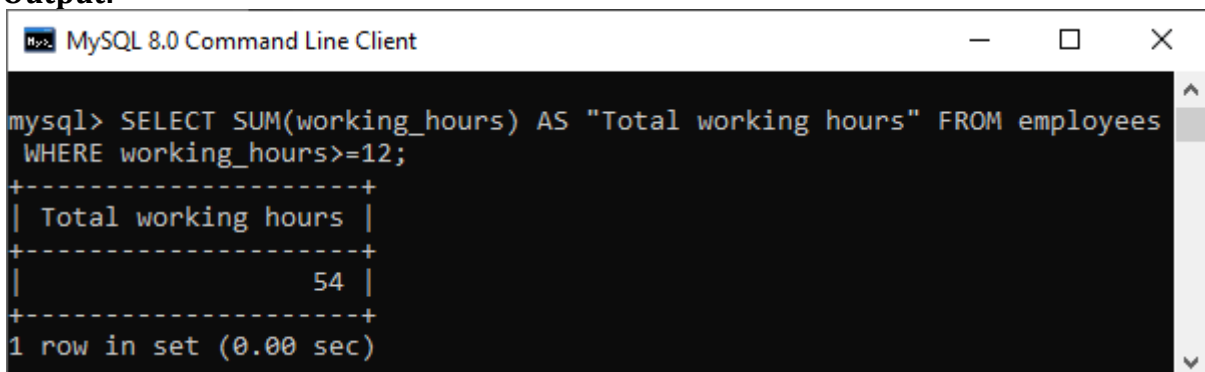
```
mysql> SELECT SUM(working_hours) AS "Total working hours" FROM employees;
+-----+
| Total working hours |
+-----+
|                111 |
+-----+
1 row in set (0.00 sec)
```

2. 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** >= 12.

```
mysql> SELECT SUM(working_hours) AS "Total working hours" FROM employees WHERE working_hours>=12;
```

Output:



```
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) Avg() function

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

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

AVG([DISTINCT|ALL]n)

- Returns average value of parameter(s) n.

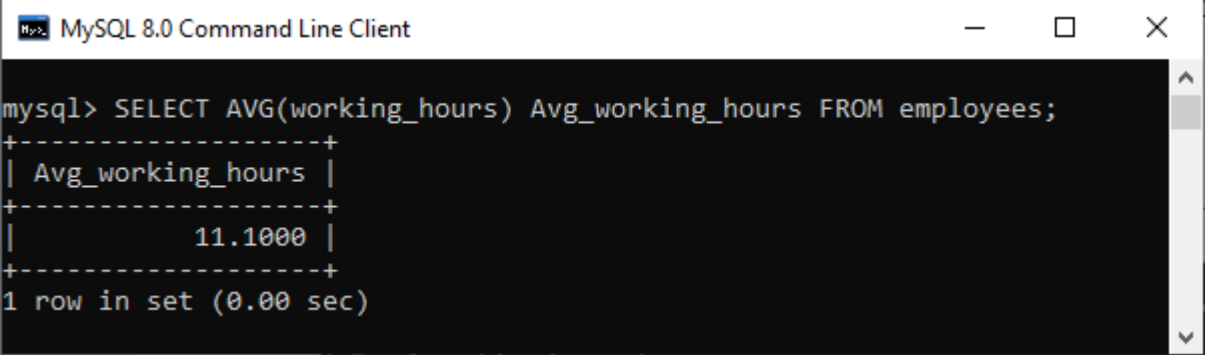
Consider our database has a table named **employees**, having the following data.

1. Basic Example

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

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

Output:



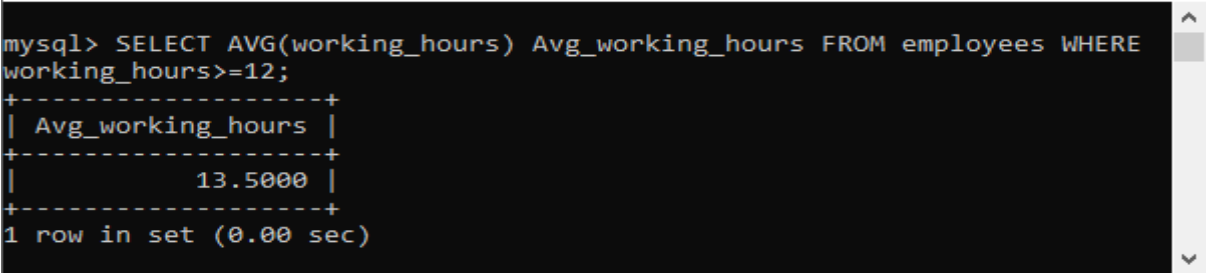
```
mysql> SELECT AVG(working_hours) Avg_working_hours FROM employees;
+-----+
| Avg_working_hours |
+-----+
|          11.1000 |
+-----+
1 row in set (0.00 sec)
```

2. 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** \geq 12.

```
mysql> SELECT AVG(working_hours) Avg_working_hours FROM employees
WHERE working_hours>=12;
```

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)
```

4) 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.

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

MIN ([DISTINCT|ALL]expr)

- Returns the minimum value of expr.

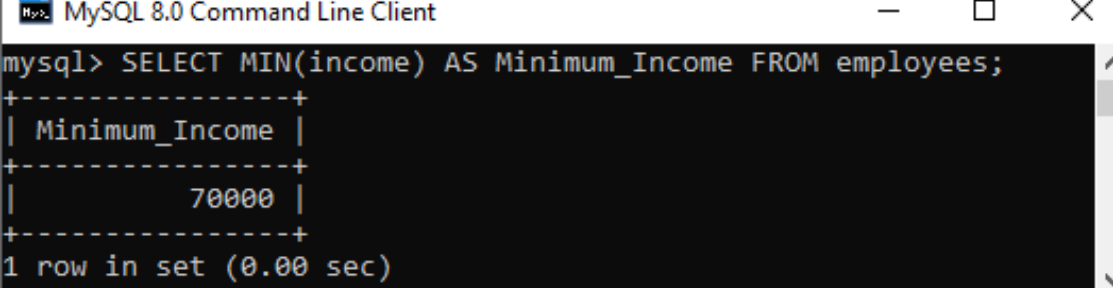
Let us understand how MIN function works in **MySQL** with the help of various examples. Consider our database has a table named "**employees**" that contains the following data.

1. Basic Example

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

```
mysql> SELECT MIN(income) AS Minimum_Income FROM employees;
```

Output



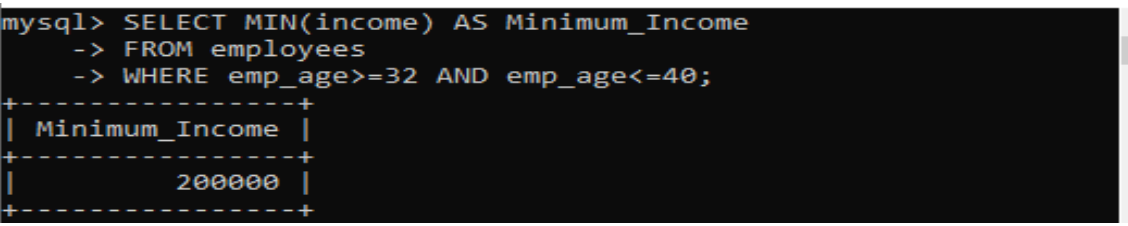
```
mysql> SELECT MIN(income) AS Minimum_Income FROM employees;
+-----+
| Minimum_Income |
+-----+
|          70000 |
+-----+
1 row in set (0.00 sec)
```

2. 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.

```
mysql> SELECT MIN(income) AS Minimum_Income
FROM employees
WHERE emp_age >= 32 AND emp_age <= 40;
```

Output



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

5) MAX() Function

The 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.

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

MAX([**DISTINCT** |**ALL**]expr)

- Returns the maximum value of argument expr.

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.

1. Basic Example

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

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

Output

```
mysql> SELECT MAX(income) AS "Maximum Income" FROM employees;
+-----+
| Maximum Income |
+-----+
|          5000000 |
+-----+
1 row in set (0.00 sec)
```

2. 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.

```
mysql> SELECT MAX(income) AS "Maximum_Income"
FROM employees
WHERE emp_age > 35;
```

Output

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

GROUP BY Clause

The GROUP BY Clause is used to collect data from multiple records and group the result by one or more column. It is generally used in a SELECT statement.

You can also use some aggregate functions like COUNT, SUM, MIN, MAX, AVG etc. on the grouped column.

Syntax:

```
SELECT expression1, expression2, ... expression_n,
aggregate_function (expression)
```

FROM tables
[**WHERE** conditions]
GROUP BY expression1, expression2, ... expression_n;

expression1, expression2, ... expression_n: It specifies the expressions that are not encapsulated within an aggregate function and must be included in the GROUP BY clause.

aggregate_function: It specifies a function such as SUM, COUNT, MIN, MAX, or AVG etc.
tables: It specifies the tables, from where you want to retrieve the 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.

(i) GROUP BY Clause with COUNT function

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



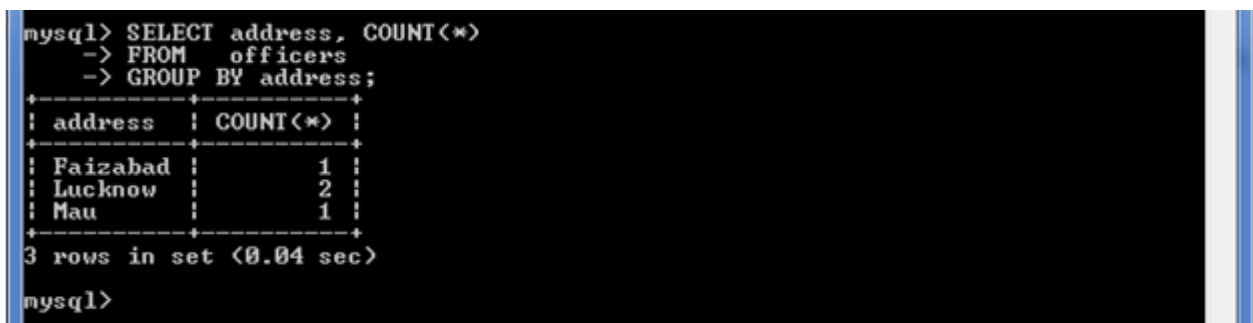
```
mysql> SELECT * FROM officers;
+----+-----+-----+
| officer_id | officer_name | address |
+----+-----+-----+
| 1 | Ajeet | Mau |
| 2 | Deepika | Lucknow |
| 3 | Uinal | Faizabad |
| 4 | Rahul | Lucknow |
+----+-----+-----+
4 rows in set (0.01 sec)
```

Now, let's count repetitive number of cities in the column address.

Execute the following query:

```
SELECT address, COUNT(*)
FROM officers
GROUP BY address;
```

Output:



```
mysql> SELECT address, COUNT(*)
-> FROM officers
-> GROUP BY address;
+-----+-----+
| address | COUNT(*) |
+-----+-----+
| Faizabad | 1 |
| Lucknow | 2 |
| Mau | 1 |
+-----+-----+
3 rows in set (0.04 sec)
mysql>
```

(ii) GROUP BY Clause with SUM function

Let's take a table "employees" table, having the following data.

```
mysql> SELECT * FROM employees;
+-----+-----+-----+-----+
| emp_id | emp_name | working_date | working_hours |
+-----+-----+-----+-----+
| 1      | Ajeet   | 2015-01-24  | 12            |
| 2      | Ayan    | 2015-01-24  | 10            |
| 3      | Milan   | 2015-01-24  | 9             |
| 4      | Ruchi   | 2015-01-24  | 6             |
| 1      | Ajeet   | 2015-01-25  | 12            |
| 2      | Ayan    | 2015-01-25  | 10            |
| 4      | Ruchi   | 2015-01-25  | 6             |
| 3      | Milan   | 2015-01-25  | 9             |
| 1      | Ajeet   | 2015-01-26  | 12            |
| 3      | Milan   | 2015-01-26  | 9             |
+-----+-----+-----+-----+
10 rows in set (0.00 sec)
```

Now, the following query will GROUP BY the example using the SUM function and return the emp_name and total working hours of each employee.

Execute the following query:

```
SELECT emp_name, SUM(working_hours) AS "Total working hours"
FROM employees
GROUP BY emp_name;
```

```
mysql>
mysql> SELECT emp_name, SUM(working_hours) AS "Total working hours"
-> FROM employees
-> GROUP BY emp_name;
+-----+-----+
| emp_name | Total working hours |
+-----+-----+
| Ajeet    | 36                  |
| Ayan     | 20                  |
| Milan    | 27                  |
| Ruchi    | 12                  |
+-----+-----+
4 rows in set (0.00 sec)
```

(iii) GROUP BY Clause with MIN function

The following example specifies the minimum working hours of the employees from the table "employees".

Execute the following query:

```
SELECT emp_name, MIN(working_hours) AS "Minimum working hour"
FROM employees
GROUP BY emp_name;
```

Output:

```
mysql> SELECT emp_name, MIN(working_hours) AS "Minimum working hour"
-> FROM employees
-> GROUP BY emp_name;
+-----+-----+
| emp_name | Minimum working hour |
+-----+-----+
| Ajeet    | 12                   |
| Ayan     | 10                   |
| Milan    | 9                    |
| Ruchi    | 6                    |
+-----+-----+
4 rows in set (0.00 sec)
```

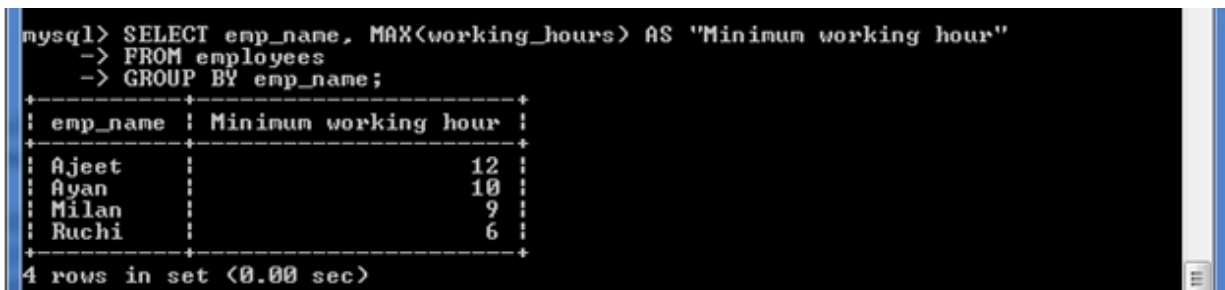
(iv) GROUP BY Clause with MAX function

The following example specifies the maximum working hours of the employees from the table "employees".

Execute the following query:

```
SELECT emp_name, MAX(working_hours) AS "Maximum working hour"  
FROM employees  
GROUP BY emp_name;
```

Output:



```
mysql> SELECT emp_name, MAX(working_hours) AS "Minimum working hour"  
-> FROM employees  
-> GROUP BY emp_name;  
+-----+-----+  
| emp_name | Minimum working hour |  
+-----+-----+  
| Ajeet   | 12                   |  
| Ayan    | 10                   |  
| Milan   | 9                    |  
| Ruchi   | 6                    |  
+-----+-----+  
4 rows in set (0.00 sec)
```

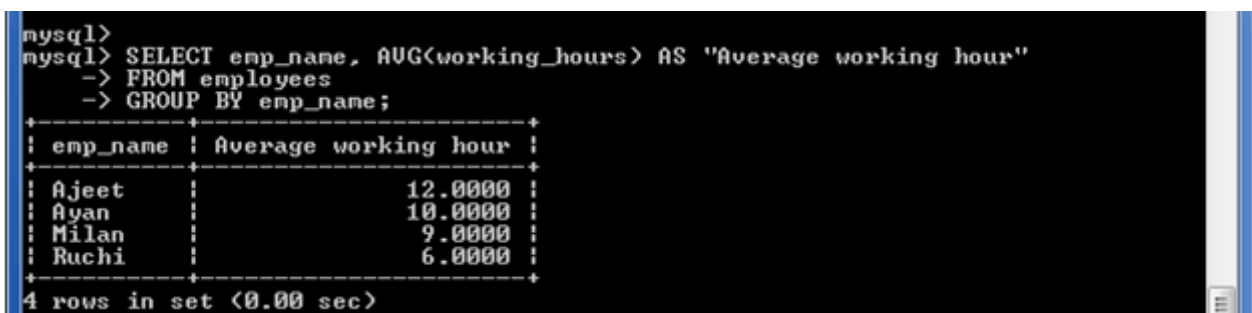
(v) GROUP BY Clause with AVG function

The following example specifies the average working hours of the employees from the table "employees".

Execute the following query:

```
SELECT emp_name, AVG(working_hours) AS "Average working hour"  
FROM employees  
GROUP BY emp_name;
```

Output:



```
mysql>  
mysql> SELECT emp_name, AVG(working_hours) AS "Average working hour"  
-> FROM employees  
-> GROUP BY emp_name;  
+-----+-----+  
| emp_name | Average working hour |  
+-----+-----+  
| Ajeet   | 12.0000              |  
| Ayan    | 10.0000              |  
| Milan   | 9.0000               |  
| Ruchi   | 6.0000               |  
+-----+-----+  
4 rows in set (0.00 sec)
```

HAVING Clause

HAVING Clause is used with GROUP BY clause. It always returns the rows where condition is TRUE.

Syntax:

```
SELECT expression1, expression2, ... expression_n,  
aggregate_function (expression)  
FROM tables  
[WHERE conditions]  
GROUP BY expression1, expression2, ... expression_n  
HAVING condition;
```

aggregate_function: It specifies any one of the aggregate function such as SUM, COUNT, MIN, MAX, or AVG.

expression1, expression2, ... expression_n: It specifies the expressions that are not encapsulated within an aggregate function and must be included in the GROUP BY clause.

WHERE conditions: It is optional. It specifies the conditions for the records to be selected.

HAVING condition: It is used to restrict the groups of returned rows. It shows only those groups in result set whose conditions are TRUE.

HAVING Clause with SUM function

Consider a table "employees" table having the following data.

```
mysql>  
mysql> SELECT * FROM employees;  
+----+-----+-----+-----+  
| emp_id | emp_name | working_date | working_hours |  
+----+-----+-----+-----+  
| 1 | Ajeet | 2015-01-24 | 12 |  
| 2 | Ayan | 2015-01-24 | 10 |  
| 3 | Milan | 2015-01-24 | 9 |  
| 4 | Ruchi | 2015-01-24 | 6 |  
| 1 | Ajeet | 2015-01-25 | 12 |  
| 2 | Ayan | 2015-01-25 | 10 |  
| 4 | Ruchi | 2015-01-25 | 6 |  
| 3 | Milan | 2015-01-25 | 9 |  
| 1 | Ajeet | 2015-01-26 | 12 |  
| 3 | Milan | 2015-01-26 | 9 |  
+----+-----+-----+-----+  
10 rows in set (0.00 sec)
```

Here, we use the SUM function with the HAVING Clause to return the emp_name and sum of their working hours. It can also be used with COUNT, MIN, MAX and AVG functions.

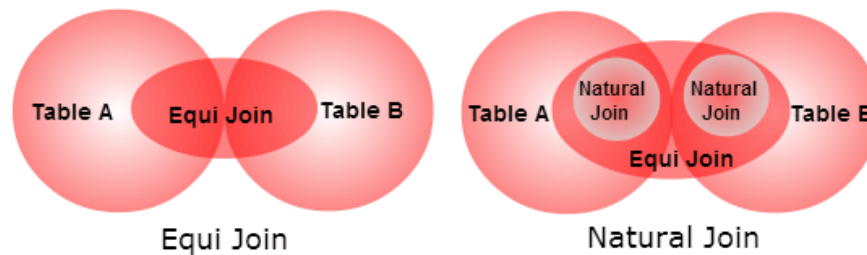
Execute the following query:

```
SELECT emp_name, SUM(working_hours) AS "Total working hours"  
FROM employees  
GROUP BY emp_name  
HAVING SUM(working_hours) > 5;
```

```
mysql> SELECT emp_name, SUM(working_hours) AS "Total working hours"  
-> FROM employees  
-> GROUP BY emp_name  
-> HAVING SUM(working_hours) > 5;  
+----+-----+-----+  
| emp_name | Total working hours |  
+----+-----+-----+  
| Ajeet | 36 |  
| Ayan | 20 |  
| Milan | 27 |  
| Ruchi | 12 |  
+----+-----+-----+
```

JOINS

A join is a query that combines rows from two or more tables. In join query, more than one tables are listed in FROM Clause. Some of the types of joins are:



1. EQUI-JOIN

SQL EQUI JOIN performs a JOIN against equality or matching column(s) values of the associated tables. An equal sign (=) is used as comparison operator in the where clause to refer equality.

You may also perform EQUI JOIN by using JOIN keyword by ON keyword and then specifying names of the columns along with their associated tables to check equality.

Syntax:

```
SELECT column_list  
FROM table1,table2.....WHERE  
table1.column_name=table2.column_name;
```

OR

```
SELECT * FROM table1 JOIN table2  
[ON(join_condition)];
```

Let us Consider two tables "Emp" and "Dept" having the following data.

```
mysql> Select * from Emp;
```

Eno	Ename	DOB	Salary	Commission	Gender	dno
101	Praveen	1989-08-06	50000	890.96	M	10
102	Sushil	1989-09-26	45000	NULL	M	20
103	Pavitra	1994-08-09	45000	100.96	M	10
105	Nisha	1990-03-23	65000	999.25	F	30
106	Neha	1990-03-08	55000	989.25	F	40
107	Shekhar	1985-12-25	60000	79.25	M	20
108	Shashank	1988-11-25	46000	709.25	M	20
109	Anamika	1988-08-14	56000	509.25	F	10
110	Kanishka	1991-02-24	42000	58.25	F	20
111	Sushil	1982-05-06	35000	89.25	M	20

```
10 rows in set (0.03 sec)
```

```
mysql> Select * from Dept;
+-----+-----+
| dno | dname |
+-----+-----+
| 10 | Purchase |
| 20 | Sales |
| 30 | Finance |
| 40 | Marketing |
+-----+-----+
4 rows in set (0.00 sec)
```

Execute the following query:

```
SELECT *
FROM Emp ,Dept
where Emp.Dno=Dept.Dno;
```

```
mysql> Select * from Emp,Dept where Emp.Dno=Dept.Dno;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| Eno | Ename | DOB | Salary | Commission | Gender | dno | dno | dname |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 101 | Praveen | 1989-08-06 | 50000 | 890.96 | M | 10 | 10 | Purchase |
| 102 | Sushil | 1989-09-26 | 45000 | NULL | M | 20 | 20 | Sales |
| 103 | Pavitra | 1994-08-09 | 45000 | 100.96 | M | 10 | 10 | Purchase |
| 105 | Nisha | 1990-03-23 | 65000 | 999.25 | F | 30 | 30 | Finance |
| 106 | Neha | 1990-03-08 | 55000 | 989.25 | F | 40 | 40 | Marketing |
| 107 | Shekhar | 1985-12-25 | 60000 | 79.25 | M | 20 | 20 | Sales |
| 108 | Shashank | 1988-11-25 | 46000 | 709.25 | M | 20 | 20 | Sales |
| 109 | Anamika | 1988-08-14 | 56000 | 509.25 | F | 10 | 10 | Purchase |
| 110 | Kanishka | 1991-02-24 | 42000 | 58.25 | F | 20 | 20 | Sales |
| 111 | Sushil | 1982-05-06 | 35000 | 89.25 | M | 20 | 20 | Sales |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
10 rows in set (0.00 sec)
```

Here EQUI JOIN performs a JOIN against equality or matching column(s) values of the associated tables Emp and Dept.

2. NATURAL-JOIN

The SQL NATURAL JOIN is a type of EQUI JOIN and is structured in such a way that, columns with the same name of associated tables will appear only once.

OR

A SQL NATURAL JOIN is a type of EQUI JOIN which occurs implicitly by comparing all the same names columns in both tables. The join result has only one column for each pair of equally named columns.

Syntax:

```
SELECT * FROM table1
NATUTAL JOIN table 2 ;
```

Let us Consider two tables "Emp" and "Dept" having the following data.

```
mysql> Select * from Emp;
```

Eno	Ename	DOB	Salary	Commission	Gender	dno
101	Praveen	1989-08-06	50000	890.96	M	10
102	Sushil	1989-09-26	45000	NULL	M	20
103	Pavitra	1994-08-09	45000	100.96	M	10
105	Nisha	1990-03-23	65000	999.25	F	30
106	Neha	1990-03-08	55000	989.25	F	40
107	Shekhar	1985-12-25	60000	79.25	M	20
108	Shashank	1988-11-25	46000	709.25	M	20
109	Anamika	1988-08-14	56000	509.25	F	10
110	Kanishka	1991-02-24	42000	58.25	F	20
111	Sushil	1982-05-06	35000	89.25	M	20

```
10 rows in set (0.03 sec)
```

```
mysql> Select * from Dept;
```

dno	dname
10	Purchase
20	Sales
30	Finance
40	Marketing

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

Execute the following query:

SELECT * FROM Emp Natural join Dept;

```
mysql> Select * from Emp Natural Join Dept;
```

dno	Eno	Ename	DOB	Salary	Commission	Gender	dname
10	101	Praveen	1989-08-06	50000	890.96	M	Purchase
20	102	Sushil	1989-09-26	45000	NULL	M	Sales
10	103	Pavitra	1994-08-09	45000	100.96	M	Purchase
30	105	Nisha	1990-03-23	65000	999.25	F	Finance
40	106	Neha	1990-03-08	55000	989.25	F	Marketing
20	107	Shekhar	1985-12-25	60000	79.25	M	Sales
20	108	Shashank	1988-11-25	46000	709.25	M	Sales
10	109	Anamika	1988-08-14	56000	509.25	F	Purchase
20	110	Kanishka	1991-02-24	42000	58.25	F	Sales
20	111	Sushil	1982-05-06	35000	89.25	M	Sales

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

Here in natural join, column dno of Emp and Dept tables will appear only once. Since * has been given in the query, all the columns of both the tables will displayed.

References/Acknowledgement:

1. Materials from Wikipedia.
2. Self generated SQL outputs.