# 25 Essential MySQL Select Command Examples

If you are using MySQL database, it is essential that you become comfortable with mysql command line.

In this tutorial we'll explain how to use the MySQL select command with several practical examples.

## 1. Basic Select command Example

First, to connect to MySQL command line, do the following from your operating system prompt.

```
# mysql -u root -p
Password:
```

Next, view all available databases.

```
mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| thegeekstuff       |
| crm                |
| bugzilla           |
+--------------------+
8 rows in set (0.00 sec)
```

Use the database where you want to work on. In this example, I'm selecting "thegeekstuff" database, where the "employee" table is located, which is used as an example for all the select commands explained in this article.

```
mysql> USE thegeekstuff;
Database changed

mysql> DESC employee;
```

The basic usage of select command is to view rows from a table. The following select command example will display all the rows from the "employee" table.

```
mysql> SELECT * FROM employee;
+-----+--------+------------+--------+
| id  | name   | dept       | salary |
+-----+--------+------------+--------+
| 100 | Thomas | Sales      |   5000 |
| 200 | Jason  | Technology |   5500 |
| 300 | Sanjay | Technology |   7000 |
| 400 | Nisha  | Marketing  |   9500 |
| 500 | Randy  | Technology |   6000 |
| 501 | Ritu   | Accounting |   NULL |
+-----+--------+------------+--------+
6 rows in set (0.01 sec)
```

Or, select specific columns by specifying the column names (Instead of * which will give all columns).

```
mysql> SELECT name, salary FROM employee;
+--------+--------+
| name   | salary |
+--------+--------+
| Thomas |   5000 |
| Jason  |   5500 |
| Sanjay |   7000 |
| Nisha  |   9500 |
| Randy  |   6000 |
| Ritu   |   NULL |
+--------+--------+
6 rows in set (0.00 sec)
```

**Note:** If you are new to MySQL, read our previous article on how to create a MySQL database and table before you continue this tutorial.

## 2. Select from Dual – Virtual Table

dual is a virtual table. This really doesn't exist. But, you can use this table to perform some non-table activities.

For example, you can use select on dual table to perform arithmetic operations as shown below.

```
mysql> SELECT 2+3 FROM DUAL;
+-----+
| 2+3 |
+-----+
|   5 |
+-----+
1 row in set (0.00 sec)
```

You can also use dual table to view the current date and time. The now() function in MySQL is similar to the sysdate() function in Oracle database.

```
mysql> SELECT NOW() FROM DUAL;
+---------------------+
| now()               |
+---------------------+
| 2013-09-14 09:15:35 |
+---------------------+
1 row in set (0.00 sec)
```

When you don't specify any table, MySQL will assume that you want to use dual. The following example are exactly same as the above. Just to avoid confusion, I recommend that you use "from dual" in these situation for better readability and clarity.

```
mysql> SELECT 2+3;
+-----+
| 2+3 |
+-----+
|   5 |
+-----+
1 row in set (0.00 sec)

mysql> SELECT NOW();
```

```
+---------------------+
| now()               |
+---------------------+
| 2013-09-14 09:16:45 |
+---------------------+
1 row in set (0.00 sec)
```

## 3. Basic WHERE Condition to Restrict Records

Instead of display all the records from a table, you can also use WHERE condition to view only recrods that matches a specific condition as shown below.

```
mysql> SELECT * FROM employee WHERE salary > 6000;
+-----+--------+------------+--------+
| id  | name   | dept       | salary |
+-----+--------+------------+--------+
| 300 | Sanjay | Technology |   7000 |
| 400 | Nisha  | Marketing  |   9500 |
+-----+--------+------------+--------+
2 rows in set (0.00 sec)
```

Similar to “greater than >” you can also use “less than=”, “not equal to !=” as shown below.

```
mysql> SELECT * FROM employee WHERE salary < 6000; mysql> SELECT * FROM employee
WHERE salary  SELECT * FROM employee WHERE salary >= 6000;

mysql> SELECT * FROM employee WHERE salary = 6000;

mysql> SELECT * FROM employee WHERE salary != 6000;
```

## 4. Match Strings in WHERE Condition

The previous example displays how to restrict records based on numerical conditions. This example explains how to restrict records based on string values.

The exact match of strings works like numeric match using “equal to =” as shown below. This example will display all employees who belong to Technology department.

```
mysql> SELECT * FROM employee WHERE dept = 'Technology';
+-----+--------+------------+--------+
| id  | name   | dept       | salary |
+-----+--------+------------+--------+
| 200 | Jason  | Technology |   5500 |
| 300 | Sanjay | Technology |   7000 |
| 500 | Randy  | Technology |   6000 |
+-----+--------+------------+--------+
3 rows in set (0.00 sec)
```

Please note that this is case insensitive comparison. So, the following is exactly the same as above select command.

```
mysql> SELECT * FROM employee WHERE dept = 'TECHNOLOGY';
```

You can also use != to display all the employee who does not belong to Technology department as shown below.

```
mysql> SELECT * FROM employee WHERE dept != 'TECHNOLOGY';
```

You can also perform partial string match using % in the keywords. The following will display all employees whos last name begins with “John”.

```
mysql> SELECT * FROM employee WHERE name LIKE 'JOHN%';
```

The following will display all employees whos name ends with “Smith”.

```
mysql> SELECT * FROM employee WHERE name LIKE '%SMITH';
```

You can also give % at both beginning and end. In which case, it will search for the given keyword anywhere in the string. The following will display all employees who contain “John” in their name anywhere.

```
mysql> SELECT * FROM employee WHERE name LIKE '%JOHN%';
```

## 5. Combine WHERE Conditions Using OR, AND

You can also use OR, AND, NOT in WHERE condition to combine multiple conditions. The following example displays all employees who are in “Technology” department AND with salary >= 6000. This will display records only when both the conditions are met.

```
mysql> SELECT * FROM employee WHERE dept = 'TECHNOLOGY' AND salary >= 6000;
+-----+--------+------------+--------+
| id  | name   | dept       | salary |
+-----+--------+------------+--------+
| 300 | Sanjay | Technology |   7000 |
| 500 | Randy  | Technology |   6000 |
+-----+--------+------------+--------+
2 rows in set (0.00 sec)
```

The following is same as above, but uses OR condition. So, this will display records as long as any one of the condition matches.

```
mysql> SELECT * FROM employee WHERE dept = 'TECHNOLOGY' OR salary >= 6000;
+-----+--------+------------+--------+
| id  | name   | dept       | salary |
+-----+--------+------------+--------+
| 200 | Jason  | Technology |   5500 |
| 300 | Sanjay | Technology |   7000 |
| 400 | Nisha  | Marketing  |   9500 |
| 500 | Randy  | Technology |   6000 |
+-----+--------+------------+--------+
4 rows in set (0.00 sec)
```

## 6. Combine column values using CONCAT in select

You can use CONCAT function in select commanda to combine values from multiple columns and display it. The following example combines name and department field (for display only) as shown below.

```
mysql> SELECT ID, CONCAT(NAME, ' FROM ', DEPT) AS NAME, SALARY FROM employee;
+-----+-----------------------+--------+
| id  | name                  | salary |
+-----+-----------------------+--------+
| 100 | Thomas from Sales     |   5000 |
| 200 | Jason from Technology |   5500 |
| 300 | Sanjay from Technology|   7000 |
| 400 | Nisha from Marketing  |   9500 |
| 500 | Randy from Technology |   6000 |
| 501 | Ritu from Accounting  |   NULL |
+-----+-----------------------+--------+
6 rows in set (0.00 sec)
```

## 7. Count Total Number of Records

Use count(*) in select command to display the total number of records in a table.

```
mysql> SELECT COUNT(*) FROM employee;
+----------+
| count(*) |
+----------+
|        6 |
+----------+
1 row in set (0.00 sec)
```

## 8. Group By in Select Command

Group By commands will group records based on certain conditions. The following example displays the total number of employees in every department.

```
mysql> SELECT DEPT, COUNT(*) FROM employee GROUP BY DEPT;
+------------+----------+
| dept       | count(*) |
+------------+----------+
| Accounting |        1 |
| Marketing  |        1 |
| Sales      |        1 |
| Technology |        3 |
+------------+----------+
4 rows in set (0.00 sec)
```

Please note that when you use GROUP BY, you can use certain functions to get more meaningful output. IN the above example, we've used count(*) group by commands. Similarly you can use sum(), avg(), etc, when you specify GROUP BY.

## 9. Use HAVING along with GROUP BY

When you use GROUP BY, you can also use HAVING to restrict the records further.

In the following example, it displays only the departments where the number of employee is more than 1.

```
mysql> SELECT COUNT(*) AS CNT, DEPT FROM employee GROUP BY DEPT HAVING CNT > 1;
+-----+------------+
| CNT | dept       |
+-----+------------+
|   3 | Technology |
+-----+------------+
1 row in set (0.00 sec)
```

## 10. Define Alias using ‘AS’ Keyword

Instead of display the column name as specified in the table, you can use your own name in the display using AS keyword.

In the following example, even though the real column name is ID, it is displayed as EMPID.

```
mysql> SELECT ID AS EMPID, NAME AS EMPNAME, DEPT AS DEPARTMENT FROM employee;
+-------+---------+------------+
| EMPID | EMPNAME | DEPARTMENT |
+-------+---------+------------+
|   100 | Thomas  | Sales      |
|   200 | Jason   | Technology |
|   300 | Sanjay  | Technology |
|   400 | Nisha   | Marketing  |
|   500 | Randy   | Technology |
|   501 | Ritu    | Accounting |
+-------+---------+------------+
6 rows in set (0.00 sec)
```

Please note that the AS keyword is optional. The following example is exactly the same as the above.

```
mysql> SELECT id empid, name empname, dept department FROM employee;
```

## 11. Left Join in SELECT command

In the following example, the select command combines two tables. i.e employee and department. For combining these, it uses the common column between these two tables dept. The “Location” column shown in the output is from the department table.

```
mysql> SELECT employee.*, department.location FROM employee LEFT JOIN department ON
( employee.dept = department.dept );
+-----+--------+------------+--------+----------+
| id  | name   | dept       | salary | Location |
+-----+--------+------------+--------+----------+
| 100 | Thomas | Sales      |   5000 | USA      |
| 200 | Jason  | Technology |   5500 | USA      |
| 300 | Sanjay | Technology |   7000 | India    |
| 400 | Nisha  | Marketing  |   9500 | India    |
| 500 | Randy  | Technology |   6000 | UK       |
| 501 | Ritu   | Accounting |   NULL | USA      |
+-----+--------+------------+--------+----------+
```

You can also use table alias name in the JOIN command as shown below. In this example, I've used "E" as alias for employee table, and "D" as alias for department table. This makes the select command smaller and easier to read.

```
mysql> SELECT E.*, d.location FROM employee AS E LEFT JOIN department AS D ON
( e.dept = d.dept );
```

Note: Join itself is a huge topic, which we will discuss in detail as a separate tutorial.

## 12. Performance Analysis using EXPLAIN

When your select query is slow, or behaving in a way you don't understand, use the EXPLAIN command, which will display additional details that MySQL is using internally to execute the query. This might give you some insight on the performance of your MySQL select command.

```
mysql> EXPLAIN SELECT E.*, D.LOCATION FROM employee AS E LEFT JOIN DEPARTMENT AS D
ON ( E.DEPT = D.DEPT );
+----+-------------+-------+--------+---------------+---------+---------
+-----------------+------+-------+
| id | select_type | table | type   | possible_keys | key     | key_len | ref
| rows | Extra |
+----+-------------+-------+--------+---------------+---------+---------
+-----------------+------+-------+
|  1 | SIMPLE      | PS    | ALL    | NULL          | NULL    | NULL    | NULL
|    6 |       |
|  1 | SIMPLE      | P     | eq_ref | PRIMARY       | PRIMARY | 3       |
acme.E.dept     |    1 |       |
+----+-------------+-------+--------+---------------+---------+---------
+-----------------+------+-------+
2 rows in set (0.00 sec)
```

## 13. Force Select Query to use an INDEX

While executing a select query, and joining two tables, MySQL will decide how to use any available Indexes on the tables effectively. The following are few ways of dealing with indexes in SELECT command.

- USE INDEX (list_of_indexes) – This will use one of the indexes specified to query the records from the table.
- IGNORE INDEX (list_of_indexes) – This will use the indexes specified to query the records from the table.
- FORCE INDEX (index_name) – This will force MySQL to use the given index even when MySQL thinks a better and faster way of querying the records are available.

Before you decide to use any one of the above, you should really understand the impact of these commands, as if you don't use these properly, it will slow down your select command.

The following examples forces MySQL to use the employee_emp_nm_idx for this query.

```
mysql> SELECT * FROM employee FORCE INDEX (EMPLOYEE_EMP_NM_IDX) WHERE NAME LIKE
'JOHN%';
```

To display all available indexes on a particular table, use the “show index” command. The following example displays all indexes available on employee table.

```
mysql> SHOW INDEX FROM PROFILES;
+----------+------------+------------------------+--------------+------------
+-----------+-------------+----------+--------+------+------------+---------+
| Table    | Non_unique | Key_name               | Seq_in_index | Column_name |
Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment |
+----------+------------+------------------------+--------------+------------
+-----------+-------------+----------+--------+------+------------+---------+
| employee |          0 | PRIMARY                |            1 | id          | A
|         156 |     NULL | NULL   |      | BTREE      |         |
| employee |          0 | employee_emp_nm_idx    |            1 | name        | A
|         156 |     NULL | NULL   |      | BTREE      |         |
+----------+------------+------------------------+--------------+------------
+-----------+-------------+----------+--------+------+------------+---------+
```

## 14. Sort Records using ORDER BY

Note: desc will short by descending. If you don’t give anything, it is ascending.
The following records will order the records in alphabetical order based on dept column.

```
mysql> SELECT * FROM employee ORDER BY DEPT;
+-----+--------+------------+--------+
| id  | name   | dept       | salary |
+-----+--------+------------+--------+
| 501 | Ritu   | Accounting |   NULL |
| 400 | Nisha  | Marketing  |   9500 |
| 100 | Thomas | Sales      |   5000 |
| 200 | Jason  | Technology |   5500 |
| 300 | Sanjay | Technology |   7000 |
| 500 | Randy  | Technology |   6000 |
+-----+--------+------------+--------+
6 rows in set (0.01 sec)
```

Please note that by default it will sort by ascending order. If you want to sort by descending order, specify the keyword “DESC” after “ORDER BY” as shown below.

```
mysql> SELECT * FROM employee ORDER BY DEPT DESC;
+-----+--------+------------+--------+
| id  | name   | dept       | salary |
+-----+--------+------------+--------+
| 200 | Jason  | Technology |   5500 |
| 300 | Sanjay | Technology |   7000 |
| 500 | Randy  | Technology |   6000 |
| 100 | Thomas | Sales      |   5000 |
| 400 | Nisha  | Marketing  |   9500 |
| 501 | Ritu   | Accounting |   NULL |
+-----+--------+------------+--------+
6 rows in set (0.00 sec)
```

You can also order by multiple columns as shown below.

```
mysql> SELECT * FROM employee ORDER BY DEPT, SALARY DESC;
```

## 15. Limit the Number of Records

Instead of displaying all the records you can just limit how many records mysql should display using the LIMIT as shown below.

Limit format:

```
LIMIT start_record, number_of_records
```

The following example will start from record number 0 (which is the 1st record), and display 3 records from there.

```
mysql> SELECT * FROM employee LIMIT 0,3;
+-----+--------+------------+--------+
| id  | name   | dept       | salary |
+-----+--------+------------+--------+
| 100 | Thomas | Sales      |   5000 |
| 200 | Jason  | Technology |   5500 |
| 300 | Sanjay | Technology |   7000 |
+-----+--------+------------+--------+
3 rows in set (0.00 sec)
```

The following will start from record number 1 (which is the 2nd record), and display 3 records from there.

```
mysql> SELECT * FROM employee LIMIT 1,3;
+-----+--------+------------+--------+
| id  | name   | dept       | salary |
+-----+--------+------------+--------+
| 200 | Jason  | Technology |   5500 |
| 300 | Sanjay | Technology |   7000 |
| 400 | Nisha  | Marketing  |   9500 |
+-----+--------+------------+--------+
3 rows in set (0.00 sec)
```

You can also omit the start_record, in which case, it will always start from record number 0 (i.e first record).

In the following example, we've specified only one value. So, this will start from record number 0, and display 3 records from there.

```
mysql> SELECT * FROM employee LIMIT 3;
+-----+--------+------------+--------+
| id  | name   | dept       | salary |
+-----+--------+------------+--------+
| 100 | Thomas | Sales      |   5000 |
| 200 | Jason  | Technology |   5500 |
| 300 | Sanjay | Technology |   7000 |
+-----+--------+------------+--------+
3 rows in set (0.00 sec)
```

## 16. Limit the Number of Records with OFFSET

Limit OFFSET format:

```
LIMIT number_of_records OFFSET start_record
```

You can also use the keyword OFFSET, where you'll specify the start record after the keyword OFFSET.

The following will display total of 3 records. Since the offset is specified as 1, it will start from the 2nd record.

```
mysql> SELECT * FROM employee LIMIT 3 OFFSET 1
+-----+--------+------------+--------+
| id  | name   | dept       | salary |
+-----+--------+------------+--------+
| 200 | Jason  | Technology |   5500 |
| 300 | Sanjay | Technology |   7000 |
| 400 | Nisha  | Marketing  |   9500 |
+-----+--------+------------+--------+
3 rows in set (0.00 sec)
```

## 17. Get Unique Values from a Column

To display all unique values from a column, use DISTINCT.

The following example will display all the unique dept values from the employee table.

```
mysql> SELECT DISTINCT DEPT FROM employee;
+------------+
| dept       |
+------------+
| Sales      |
| Technology |
| Marketing  |
| Accounting |
+------------+
```

## 18. Sum of all Values in a Column

To add all the values from a column, use SUM() function.

The following example will display the sum of salary column for all the employees who belong to Technology department.

```
mysql> SELECT SUM(SALARY) FROM employee WHERE DEPT = 'TECHNOLOGY';
+-------------+
| sum(salary) |
+-------------+
|       18500 |
+-------------+
1 row in set (0.01 sec)
```

## 19. Average of all Values in a Column

To average all the values from a column, use AVG() function.

The following example will display the average salary of each and every department. This combines GROUP BY with AVG() function.

```
mysql> SELECT DEPT,AVG(SALARY) FROM employee GROUP BY DEPT;
+------------+-------------+
| dept       | avg(salary) |
+------------+-------------+
| Accounting |        NULL |
| Marketing  |   9500.0000 |
| Sales      |   5000.0000 |
| Technology |   6166.6667 |
+------------+-------------+
4 rows in set (0.03 sec)
```

## 20. SELECT within SELECT command

The example shown below is very lame. There is no reason to do it this way. But, this shows you how you can use select command. In this example the “AS ACTION” gives an alias name to the select subquery. You need to specify an alias in this example. The “ACTION” is just a name. You can change this to anything you like.

```
mysql> SELECT * FROM (SELECT * FROM employee) AS ACTION WHERE ID
```

## 21. Save the Select Output to a File

Using SELECT INTO, you can save the output of a select command into a file.

Instead of displaying the output on the screen, the following select command example will store the output of the select command into the /tmp/employee.txt file.

```
mysql> SELECT * INTO OUTFILE '/tmp/employee.txt' FROM employee;
Query OK, 6 rows affected (0.00 sec)

# cat /tmp/employee.txt
100     Thomas  Sales   5000
200     Jason   Technology      5500
300     Sanjay  Technology      7000
400     Nisha   Marketing       9500
500     Randy   Technology      6000
501     Ritu    Accounting      \N
```

You can also store the output into a comma delimited file by specifying the “FIELDS TERMINATED BY” as shown in the example below.

```
mysql> SELECT * INTO OUTFILE '/tmp/employee1.txt'   FIELDS TERMINATED BY ',' FROM
employee;
Query OK, 6 rows affected (0.00 sec)

# cat /tmp/employee1.txt
100,Thomas,Sales,5000
200,Jason,Technology,5500
300,Sanjay,Technology,7000
400,Nisha,Marketing,9500
500,Randy,Technology,6000
```

```
501,Ritu,Accounting,\N
```

## 22. Execute a Procedure on the Data Set

You can also call a MySQL procedure that will process the data from the output of the select command.

The following example will execute the procedure salary_report() on the output of the given select command.

```
mysql> SELECT ID, SALARY FROM employee PROCEDURE SALARY_REPORT();
```

## 23. Display a Random Record from a table

Using the rand command you can display a random record from a table. This can be helpful in situations similar to where you are displaying some random tip of the day from a table.

```
mysql> SELECT * FROM employee ORDER BY RAND() LIMIT 1;
+-----+-------+------------+--------+
| id  | name  | dept       | salary |
+-----+-------+------------+--------+
| 200 | Jason | Technology |   5500 |
+-----+-------+------------+--------+
1 row in set (0.00 sec)
```

The same command executed next time, will give a different record as shown below.

```
mysql> SELECT * FROM employee ORDER BY RAND() LIMIT 1;
+-----+--------+-------+--------+
| id  | name   | dept  | salary |
+-----+--------+-------+--------+
| 100 | Thomas | Sales |   5000 |
+-----+--------+-------+--------+
1 row in set (0.00 sec)
```

You can also pass the current date and time as salt using the now() function to rand command as shown below.

```
mysql> SELECT * FROM employee ORDER BY RAND(NOW()) LIMIT 1;
+-----+-------+-----------+--------+
| id  | name  | dept      | salary |
+-----+-------+-----------+--------+
| 400 | Nisha | Marketing |   9500 |
+-----+-------+-----------+--------+
1 row in set (0.00 sec)
```

## 24. High Priority Select Command

When you use high_priority keyword in select statement, it will give that particular select statement higher priority than any update to the table.

Be very careful when you use this command, as you might slow down other updates. Use this only in situations where you need to get a record very quickly. Also make sure the select command you are giving itself is very well optimized before you execute it.

```
mysql> SELECT HIGH_PRIORITY * FROM employee WHERE ID = 100;
```

## 25. Consistent Read in Select Command

If you want a consistent read. i.e When you are selecting rows from a table, if you don’t want any other process to modify the values, you need to enable a share lock mode when you are reading the records.

If you don’t understand the impact of how these works, you might put yourself in a difficult situation if you try to use these on a large table.

The following command will not allow other MySQL sessions to modify the records that are queried by this select statement until it reads all these records.

```
mysql> SELECT * FROM employee WHERE ID = 100 LOCK IN SHARE MODE;
```

Please note that you can also do “FOR UPDATE” as shown below, which will block other sessions from doing “SELECT … LOCK in SHARE MODE” until this transaction is over.

```
mysql> SELECT * FROM employee WHERE ID = 100 FOR UPDATE;
```