MySQL Tutorial

MySQL tutorial provides basic and advanced concepts of MySQL. Our MySQL tutorial is designed for beginners and professionals.

MySQL is a relational database management system. It is open-source and free.

Our MySQL tutorial includes all topics of MySQL database such as insert record, update record, delete record, select record, create table, drop table etc. There are also given MySQL interview questions to help you better understand the MySQL database.

What is MySQL

MySQL is a fast, easy to use relational database. It is currently the most popular open-source database. It is very commonly used in conjunction with PHP scripts to create powerful and dynamic server-side applications.

MySQL is used for many small and big businesses. It is developed, marketed and supported by MySQL AB, a Swedish company. It is written in C and C++.

Reasons of popularity

MySQL is becoming so popular because of these following reasons:

- MySQL is an open-source database so you don't have to pay a single penny to use it.
- MySQL is a very powerful program so it can handle a large set of functionality of the most expensive and powerful database packages.
- MySQL is customizable because it is an open source database and the open-source GPL license facilitates programmers to modify the SQL software according to their own specific environment.
- MySQL is quicker than other databases so it can work well even with the large data set.
- MySQL supports many operating systems with many languages like PHP, PERL, C, C++, JAVA, etc.
- MySQL uses a standard form of the well-known SQL data language.
- MySQL is very friendly with PHP, the most popular language for web development.
- MySQL supports large databases, up to 50 million rows or more in a table. The default file size limit for a table is 4GB, but you can increase this (if your operating system can handle it) to a theoretical limit of 8 million terabytes (TB).

MySQL Features

- **Relational Database Management System (RDBMS):** MySQL is a relational database management system.
- **Easy to use:** MySQL is easy to use. You have to get only the basic knowledge of SQL. You can build and interact with MySQL with only a few simple SQL statements.
- **It is secure:** MySQL consist of a solid data security layer that protects sensitive data from intruders. Passwords are encrypted in MySQL.
- **Client/ Server Architecture:** MySQL follows a client /server architecture. There is a database server (MySQL) and arbitrarily many clients (application programs), which communicate with the server; that is, they query data, save changes, etc.
- **Free to download:** MySQL is free to use and you can download it from MySQL official website.
- **It is scalable:** MySQL can handle almost any amount of data, up to as much as 50 million rows or more. The default file size limit is about 4 GB. However, you can increase this number to a theoretical limit of 8 TB of data.
- **Compatibale on many operating systems:** MySQL is compatible to run on many operating systems, like Novell NetWare, Windows* Linux*, many varieties of UNIX* (such as Sun* Solaris*, AIX, and DEC* UNIX), OS/2, FreeBSD*, and others. MySQL also provides a facility that the clients can run on the same computer as the server or on another computer (communication via a local network or the Internet).
- **Allows roll-back:** MySQL allows transactions to be rolled back, commit and crash recovery.
- **High Performance:** MySQL is faster, more reliable and cheaper because of its unique storage engine architecture.
- **High Flexibility:** MySQL supports a large number of embedded applications which makes MySQL very flexible.
- **High Productivity:** MySQL uses Triggers, Stored procedures and views which allows the developer to give a higher productivity.

Disadvantages / Drawback of MySQL:

Following are the few disadvantages of MySQL:

- MySQL version less than 5.0 doesn't support ROLE, COMMIT and stored procedure.
- MySQL does not support a very large database size as efficiently.
- MySQL doesn't handle transactions very efficiently and it is prone to data corruption.
- MySQL is accused that it doesn't have a good developing and debugging tool compared to paid databases.
- MySQL doesn't support SQL check constraints.

How to install MySQL

Download MySQL

Follow these steps:

- Go to MySQL official website **http://www.mysql.com/downloads/**
- Choose the version number for MySQL community server which you want.

Installing MySQL on Windows

Your downloaded MySQL is neatly packaged with an installer. Download the installer package, unzip it anywhere and run setup.exe.

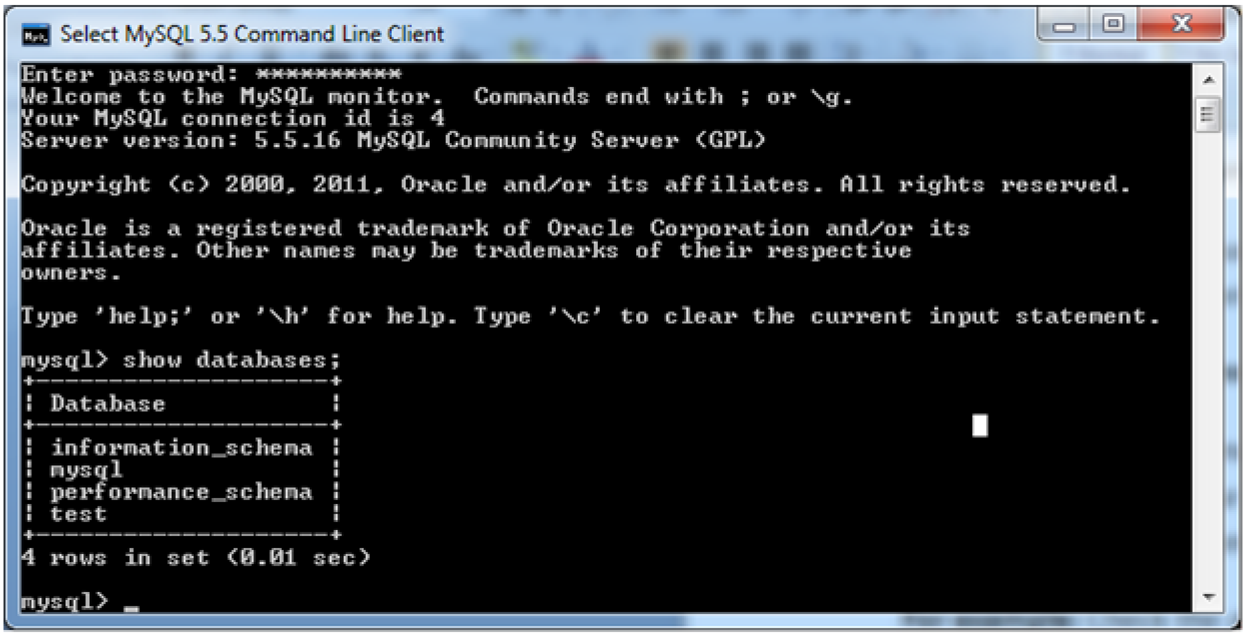
By default, this process will install everything under C:\mysql.

Verify MySQL installation

Once MySQL has been successfully installed, the base tables have been initialized, and the server has been started, you can verify its working via some simple tests.

Open your MySQL Command Line Client, it should be appeared with a mysql> prompt. If you have set any password, write your password here. Now, you are connected to the MySQL server and you can execute all the SQL command at mysql> prompt as follows:

For example: Check the already created databases with show databases command:



```
Enter password: ***********
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 4
Server version: 5.5.16 MySQL Community Server (GPL)

Copyright (c) 2000, 2011, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| test               |
+--------------------+
4 rows in set (0.01 sec)

mysql> _
```

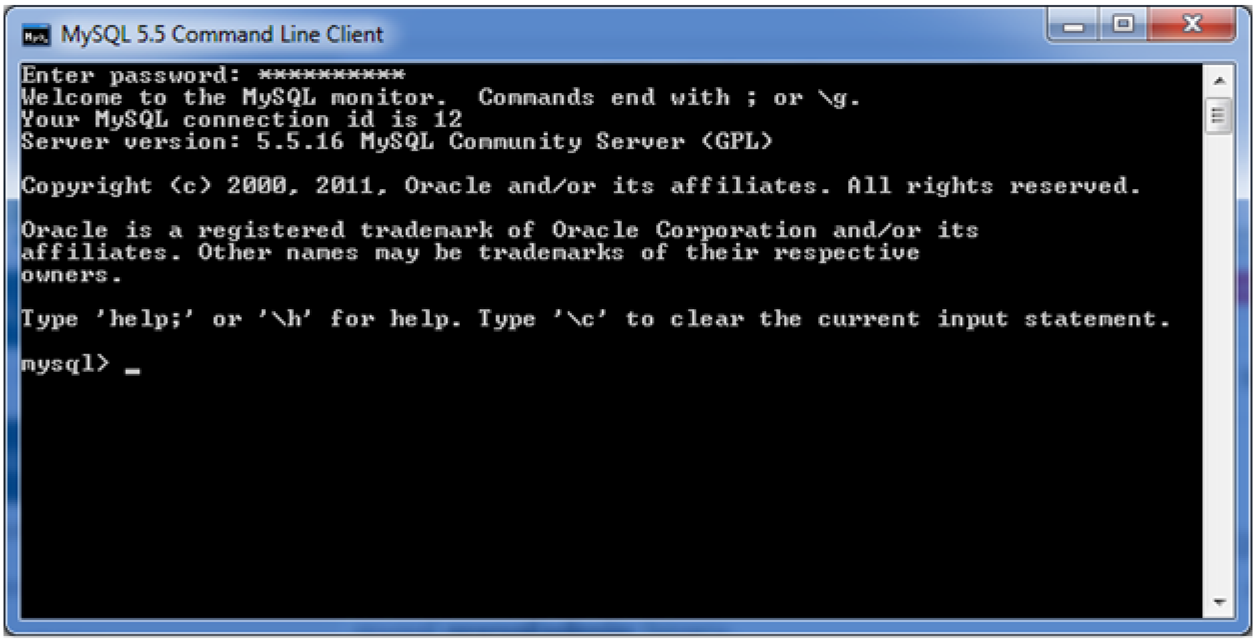
What is Primary key

A primary key is a single field or combination of fields that contains a unique record. It must be filled. None of the field of primary key can contain a null value. A table can have only one primary key.

MySQL Create Database

You can create a MySQL database by using MySQL Command Line Client.

Open the MySQL console and write down password, if you set one while installation. You will get the following:



```
Enter password: ***********
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 12
Server version: 5.5.16 MySQL Community Server (GPL)

Copyright (c) 2000, 2011, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> _
```

Now you are ready to create database.

Syntax:

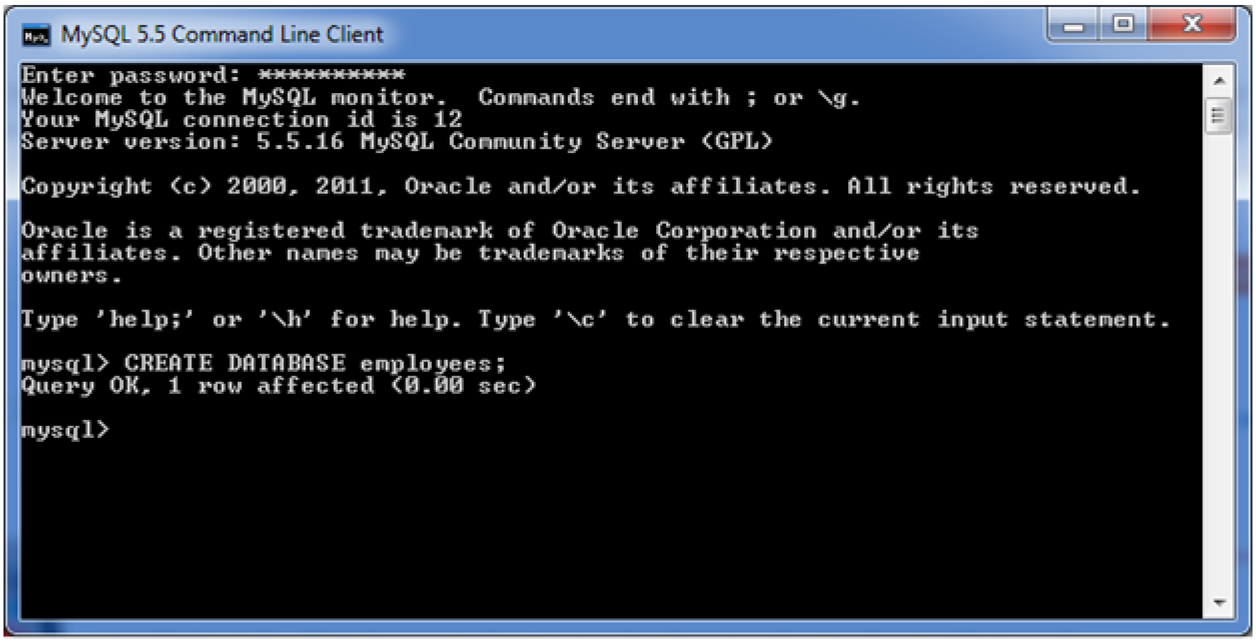
1. **CREATE DATABASE** database_name;

Example:

Let's take an example to create a database name "employees"

1. **CREATE DATABASE** employees;

It will look like this:

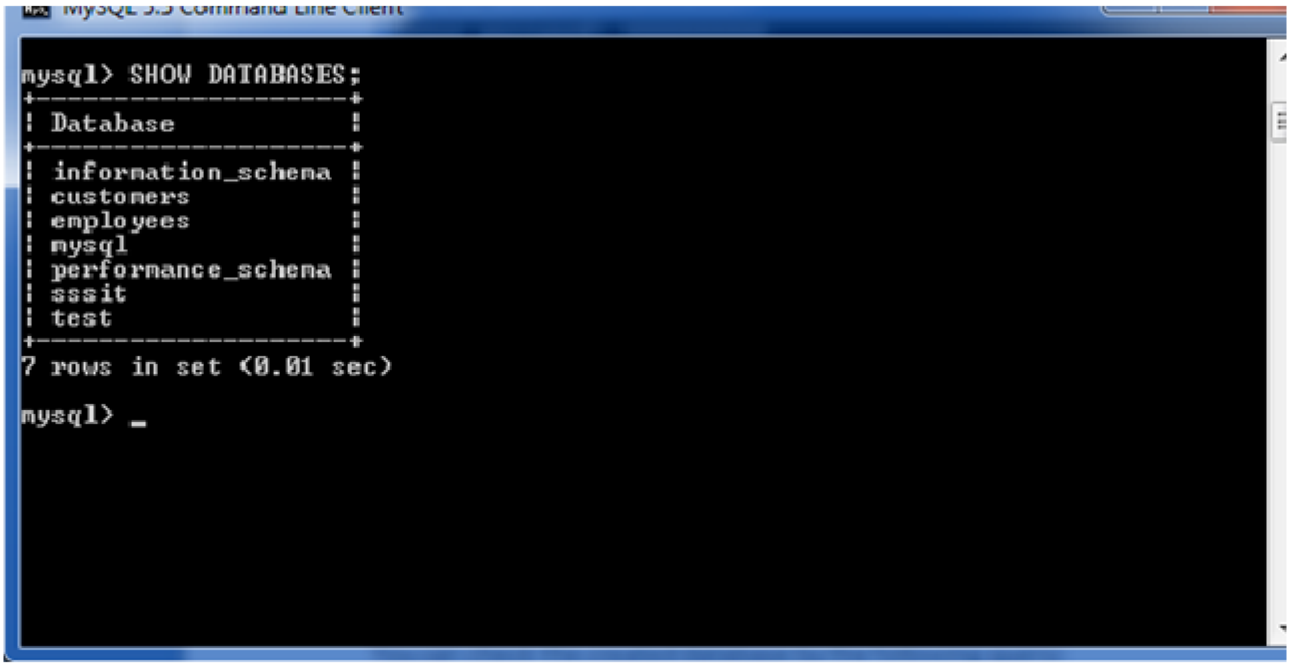


```
Enter password: ***********
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 12
Server version: 5.5.16 MySQL Community Server (GPL)

Copyright (c) 2000, 2011, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> CREATE DATABASE employees;
Query OK, 1 row affected (0.00 sec)

mysql>
```

You can check the created database by the following query:

1. SHOW DATABASES;

Output



```
mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| customers          |
| employees          |
| mysql              |
| performance_schema |
| sssit              |
| test               |
+--------------------+
7 rows in set (0.01 sec)

mysql> _
```

Here, you can see the all created databases.

MySQL SELECT Database

SELECT Database is used in MySQL to select a particular database to work with. This query is used when multiple databases are available with MySQL Server.

You can use SQL command **USE** to select a particular database.

Syntax:

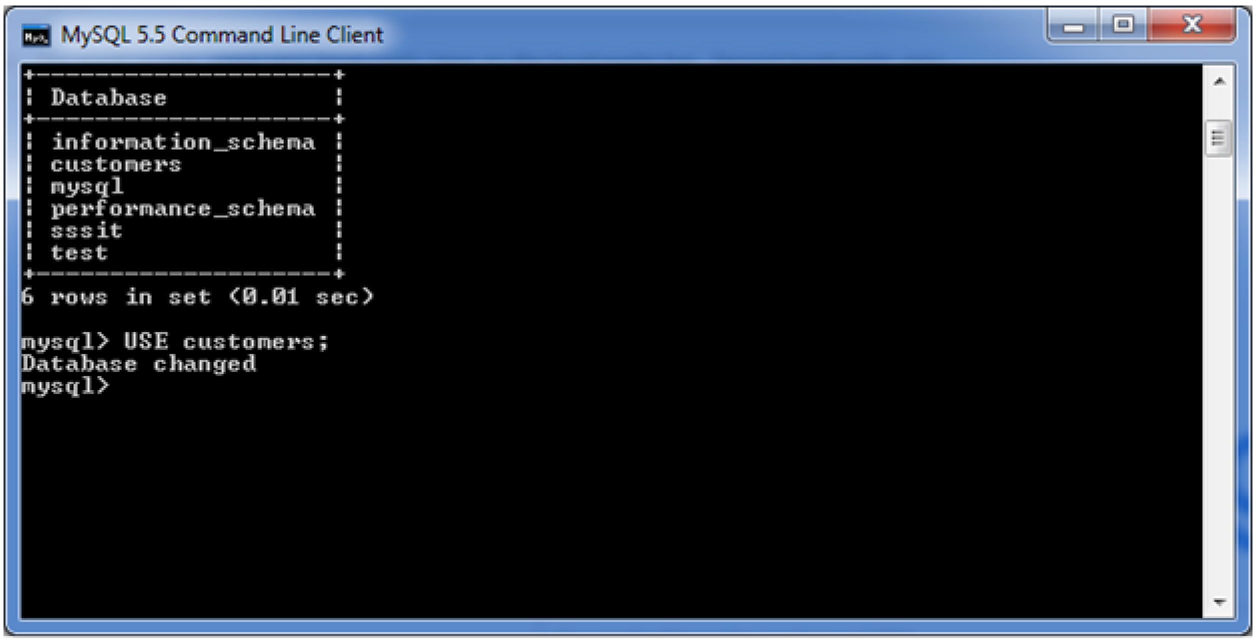
1. USE database_name;

Example:

Let's take an example to use a database name "customers".

1. USE customers;

It will look like this:



```
+--------------------+
| Database           |
+--------------------+
| information_schema |
| customers          |
| mysql              |
| performance_schema |
| sssit              |
| test               |
+--------------------+
6 rows in set (0.01 sec)

mysql> USE customers;
Database changed
mysql>
```

Note: All the database names, table names and table fields name are case sensitive. You must have to use proper names while giving any SQL command.

MySQL Drop Database

You can drop/delete/remove a MySQL database easily with the MySQL command. You should be careful while deleting any database because you will lose your all the data available in your database.

Syntax:

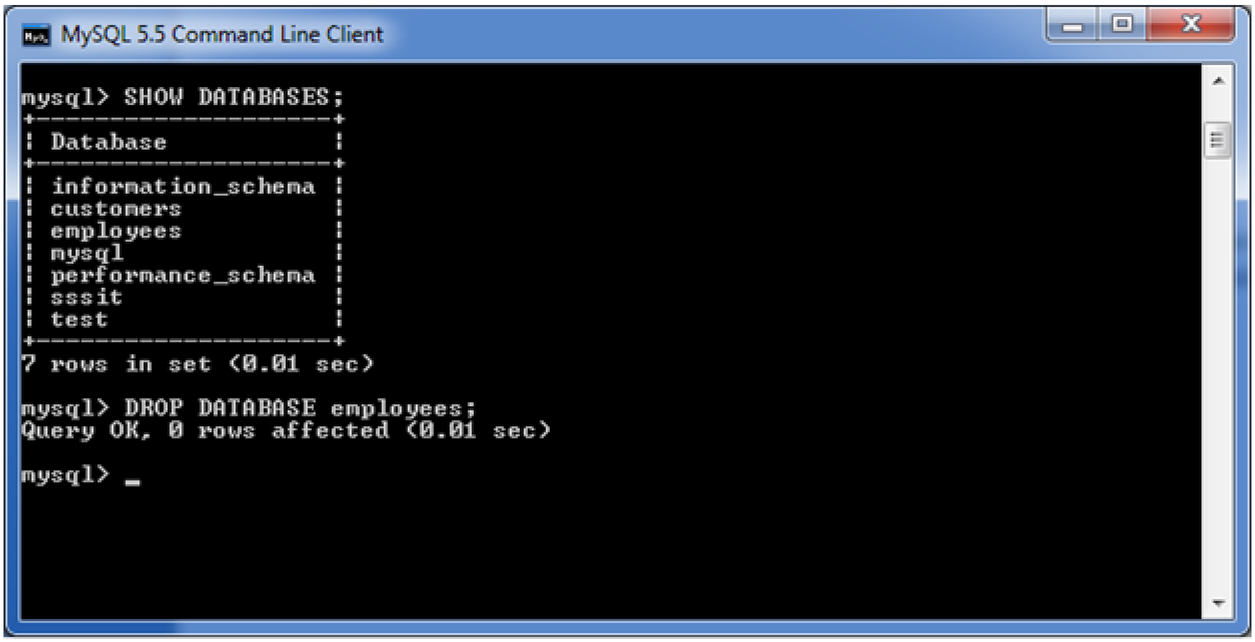
1. **DROP DATABASE** database_name;

Example:

Let's take an example to drop a database name "employees"

1. **DROP DATABASE** employees;

It will look like this:

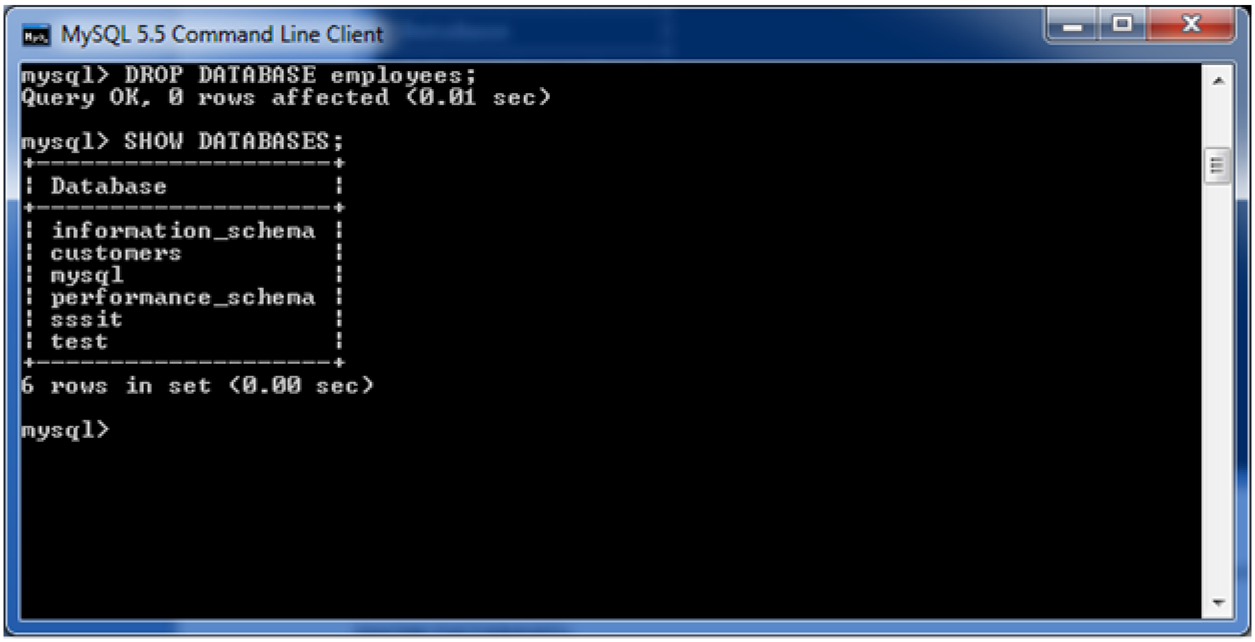


```
mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| customers          |
| employees          |
| mysql              |
| performance_schema |
| sssit              |
| test               |
+--------------------+
7 rows in set (0.01 sec)

mysql> DROP DATABASE employees;
Query OK, 0 rows affected (0.01 sec)

mysql> _
```

Now you can check that either your database is removed by executing the following query:

1. SHOW DATABASES;

Output:



```
mysql> DROP DATABASE employees;
Query OK, 0 rows affected (0.01 sec)

mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| customers          |
| mysql              |
| performance_schema |
| sssit              |
| test               |
+--------------------+
6 rows in set (0.00 sec)

mysql>
```

Here, you can see that the database "employees" is removed.

MySQL CREATE TABLE

The MySQL CREATE TABLE command is used to create a new table into the database. A table creation command requires three things:

- Name of the table
- Names of fields
- Definitions for each field

Syntax:

Following is a generic syntax for creating a MySQL table in the database.

```
CREATE TABLE table_name (column_name column_type...);
```

Example:

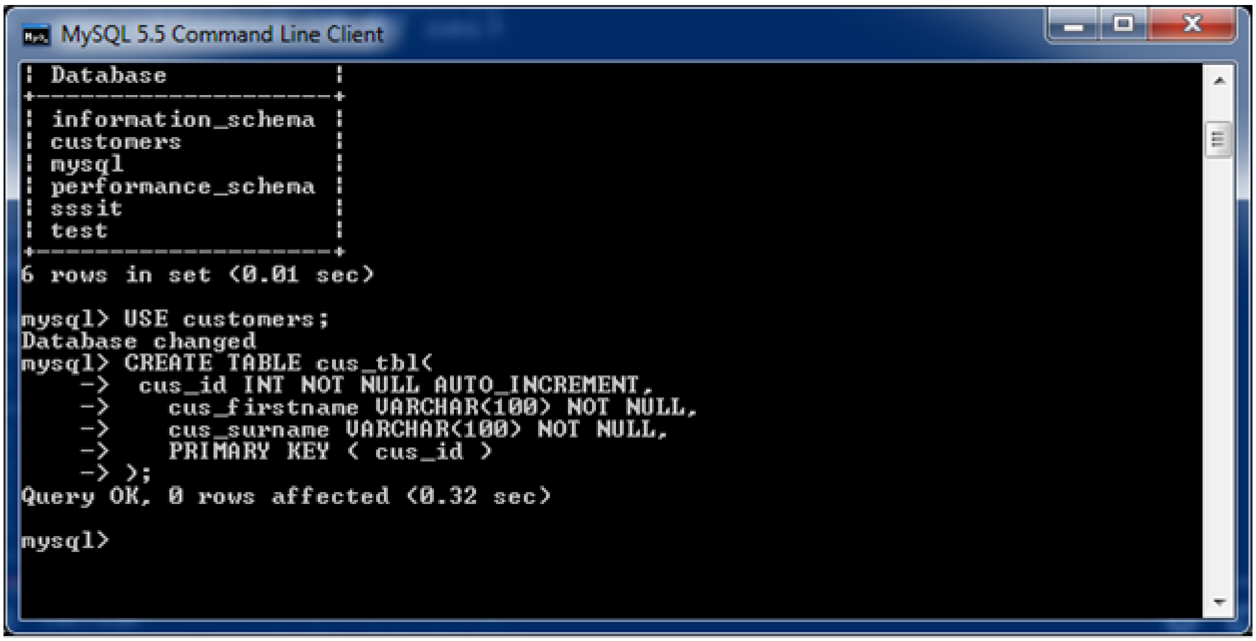
Here, we will create a table named "cus_tbl" in the database "customers".

```
CREATE TABLE cus_tbl(
  cus_id INT NOT NULL AUTO_INCREMENT,
  cus_firstname VARCHAR(100) NOT NULL,
  cus_surname VARCHAR(100) NOT NULL,
  PRIMARY KEY ( cus_id )
);
```

Note:

1. Here, NOT NULL is a field attribute and it is used because we don't want this field to be NULL. If you will try to create a record with NULL value, then MySQL will raise an error.
2. The field attribute AUTO_INCREMENT specifies MySQL to go ahead and add the next available number to the id field.PRIMARY KEY is used to define a column as primary key. You can use multiple columns separated by comma to define a primary key.

Visual representation of creating a MySQL table:

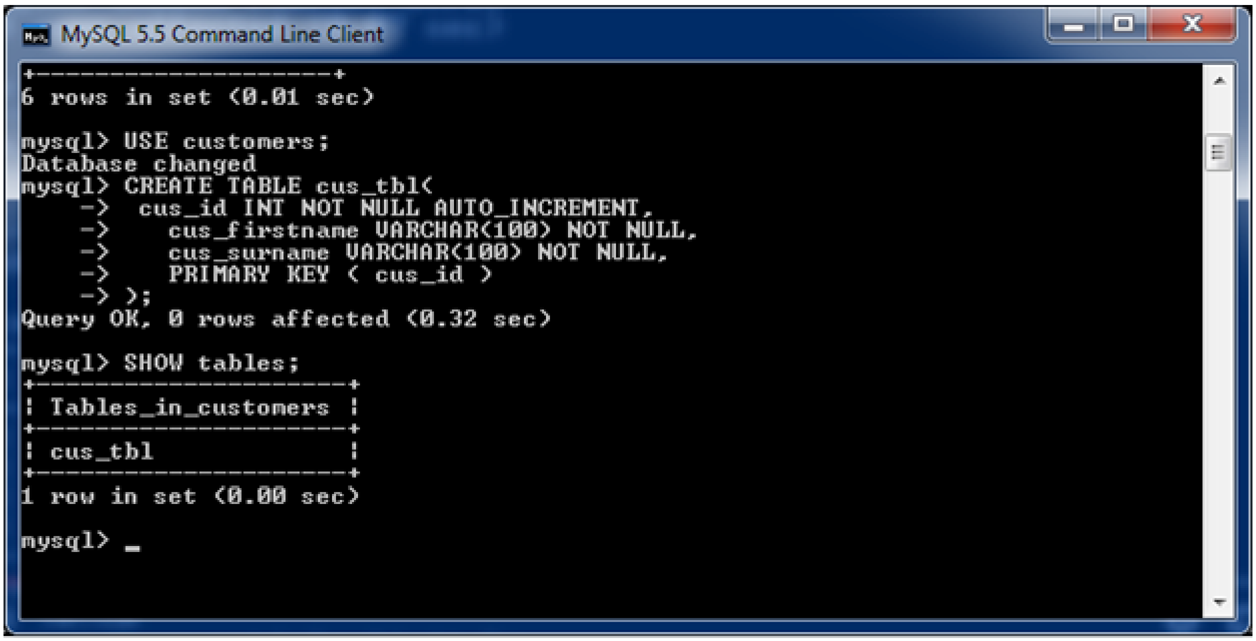


```
| Database           |
+--------------------+
| information_schema |
| customers          |
| mysql              |
| performance_schema |
| sssit              |
| test               |
+--------------------+
6 rows in set (0.01 sec)

mysql> USE customers;
Database changed
mysql> CREATE TABLE cus_tbl(
    ->  cus_id INT NOT NULL AUTO_INCREMENT,
    ->    cus_firstname VARCHAR(100) NOT NULL,
    ->    cus_surname VARCHAR(100) NOT NULL,
    ->    PRIMARY KEY ( cus_id )
    -> );
Query OK, 0 rows affected (0.32 sec)

mysql>
```

See the created table:

Use the following command to see the table already created:

1. SHOW tables;

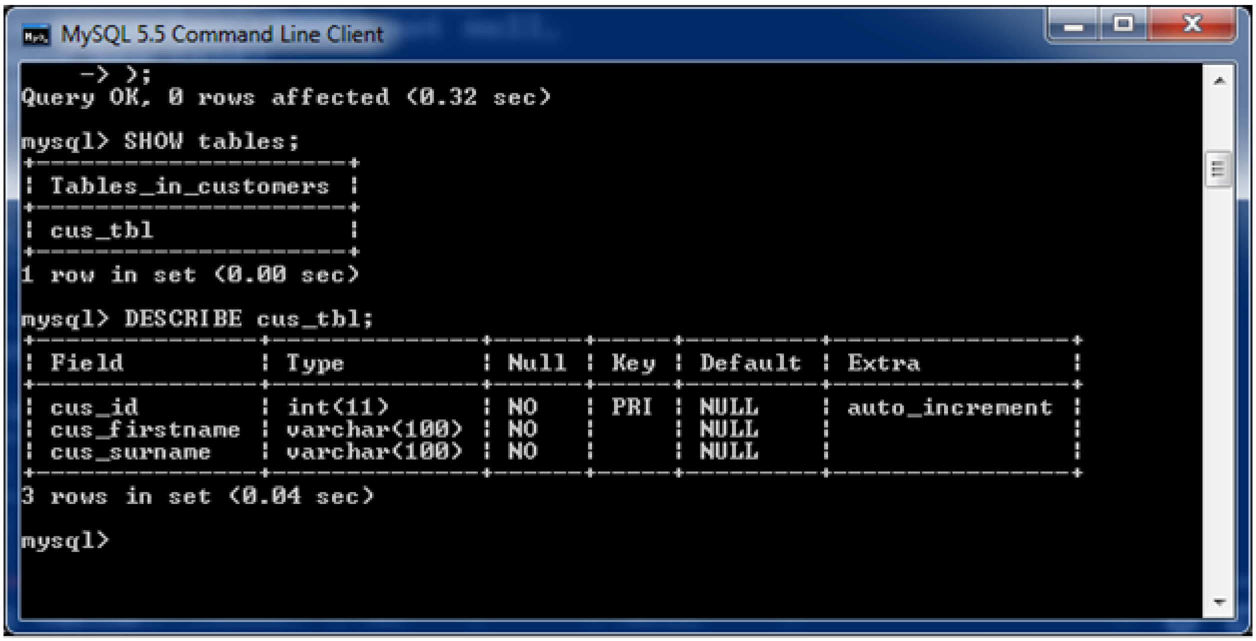


```
+--------------------+
6 rows in set (0.01 sec)

mysql> USE customers;
Database changed
mysql> CREATE TABLE cus_tbl(
    ->  cus_id INT NOT NULL AUTO_INCREMENT,
    ->    cus_firstname VARCHAR(100) NOT NULL,
    ->    cus_surname VARCHAR(100) NOT NULL,
    ->    PRIMARY KEY ( cus_id )
    -> );
Query OK, 0 rows affected (0.32 sec)

mysql> SHOW tables;
+---------------------+
| Tables_in_customers |
+---------------------+
| cus_tbl             |
+---------------------+
1 row in set (0.00 sec)

mysql> _
```

See the table structure:

Use the following command to see the table already created:

1. DESCRIBE cus_tbl;


MySQL 5.5 Command Line Client

```
    -> );
Query OK, 0 rows affected (0.32 sec)

mysql> SHOW tables;
+---------------------+
| Tables_in_customers |
+---------------------+
| cus_tbl             |
+---------------------+
1 row in set (0.00 sec)

mysql> DESCRIBE cus_tbl;
+---------------+--------------+------+-----+---------+----------------+
| Field         | Type         | Null | Key | Default | Extra          |
+---------------+--------------+------+-----+---------+----------------+
| cus_id        | int(11)      | NO   | PRI | NULL    | auto_increment |
| cus_firstname | varchar(100) | NO   |     | NULL    |                |
| cus_surname   | varchar(100) | NO   |     | NULL    |                |
+---------------+--------------+------+-----+---------+----------------+
3 rows in set (0.04 sec)

mysql>
```

MySQL ALTER Table

MySQL ALTER statement is used when you want to change the name of your table or any table field. It is also used to add or delete an existing column in a table.

The ALTER statement is always used with "ADD", "DROP" and "MODIFY" commands according to the situation.

1) ADD a column in the table

Syntax:

```
1.    ALTER TABLE table_name
2.    ADD new_column_name column_definition
3.    [ FIRST | AFTER column_name ];
```

Parameters

table_name: It specifies the name of the table that you want to modify.

new_column_name: It specifies the name of the new column that you want to add to the table.

column_definition: It specifies the data type and definition of the column (NULL or NOT NULL, etc).

FIRST | AFTER column_name: It is optional. It tells MySQL where in the table to create the column. If this parameter is not specified, the new column will be added to the end of the table.

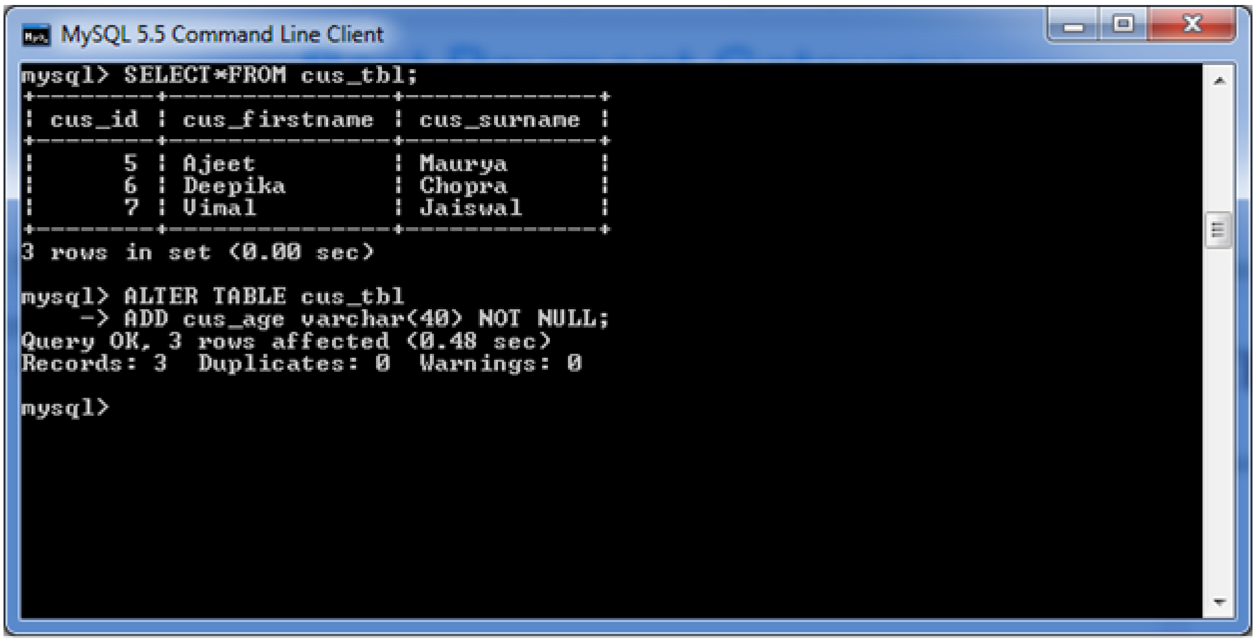
Example:

In this example, we add a new column "cus_age" in the existing table "cus_tbl".

Use the following query to do this:

```
1.    ALTER TABLE cus_tbl
2.    ADD cus_age varchar(40) NOT NULL;
```

Output:

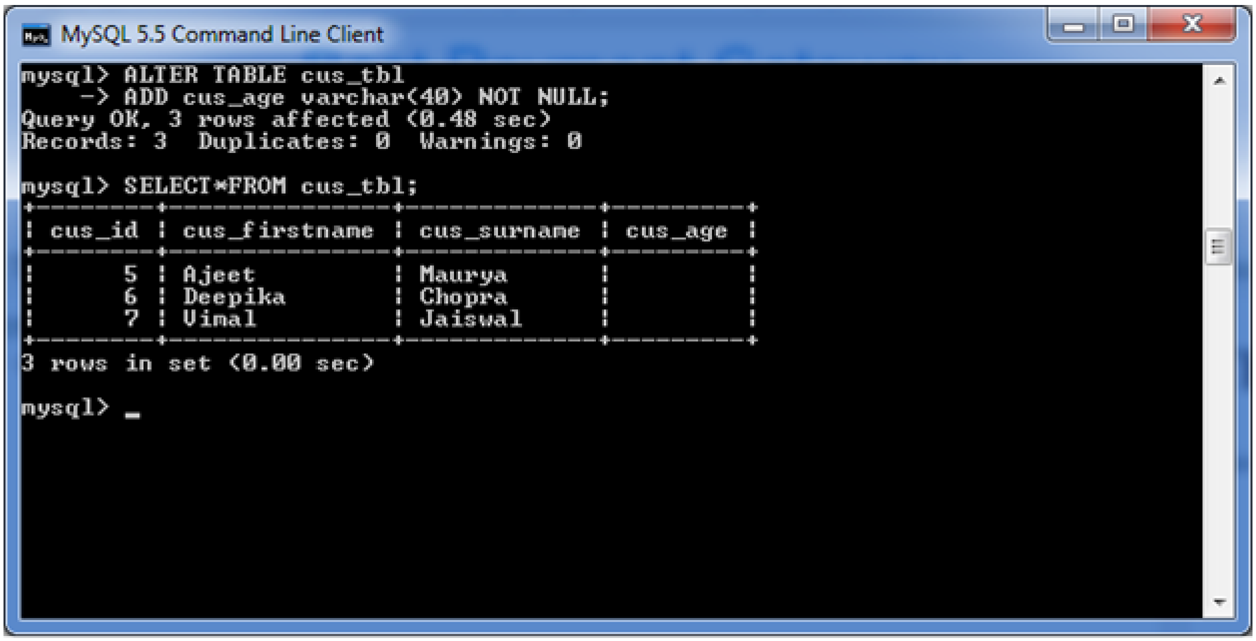


```
mysql> SELECT*FROM cus_tbl;
+--------+---------------+-------------+
| cus_id | cus_firstname | cus_surname |
+--------+---------------+-------------+
|      5 | Ajeet         | Maurya      |
|      6 | Deepika       | Chopra      |
|      7 | Vimal         | Jaiswal     |
+--------+---------------+-------------+
3 rows in set (0.00 sec)

mysql> ALTER TABLE cus_tbl
    -> ADD cus_age varchar(40) NOT NULL;
Query OK, 3 rows affected (0.48 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql>
```

See the recently added column:

```
1.    SELECT* FROM cus_tbl;
```

Output:



```
mysql> ALTER TABLE cus_tbl
    -> ADD cus_age varchar(40) NOT NULL;
Query OK, 3 rows affected (0.48 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> SELECT*FROM cus_tbl;
+--------+---------------+-------------+---------+
| cus_id | cus_firstname | cus_surname | cus_age |
+--------+---------------+-------------+---------+
|      5 | Ajeet         | Maurya      |         |
|      6 | Deepika       | Chopra      |         |
|      7 | Vimal         | Jaiswal     |         |
+--------+---------------+-------------+---------+
3 rows in set (0.00 sec)

mysql> _
```

2) Add multiple columns in the table

Syntax:

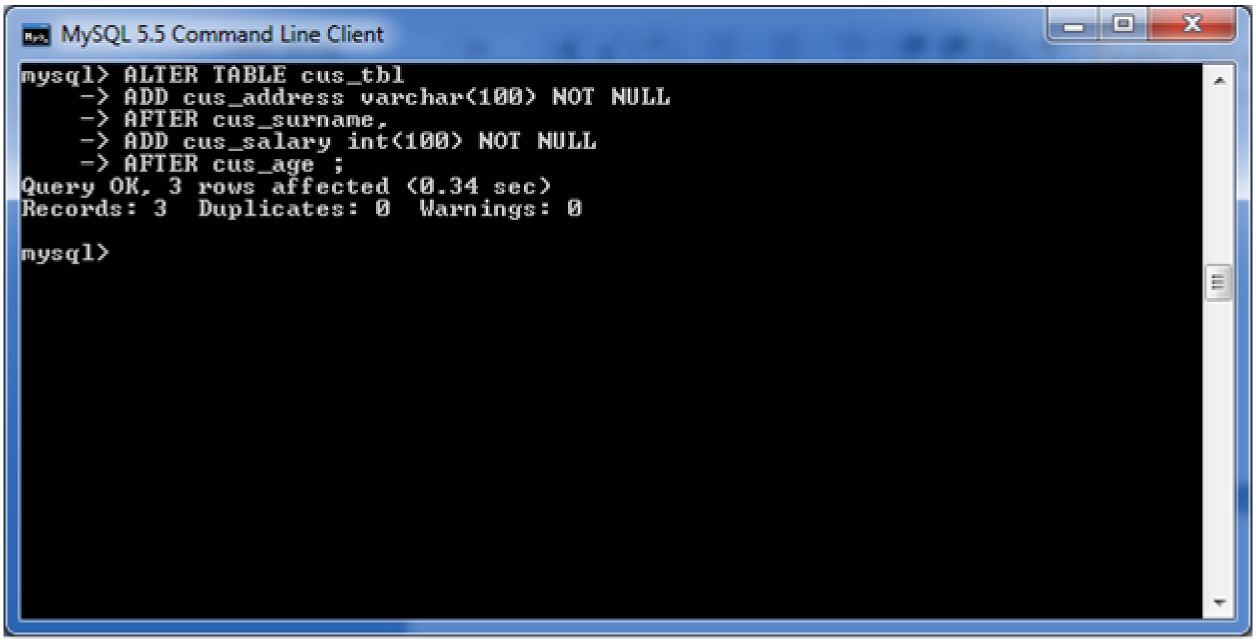
```
ALTER TABLE table_name
ADD new_column_name column_definition
[ FIRST | AFTER column_name ],
ADD new_column_name column_definition
[ FIRST | AFTER column_name ],
 ...
;
```

Example:

In this example, we add two new columns "cus_address", and cus_salary in the existing table "cus_tbl". cus_address is added after cus_surname column and cus_salary is added after cus_age column.

Use the following query to do this:

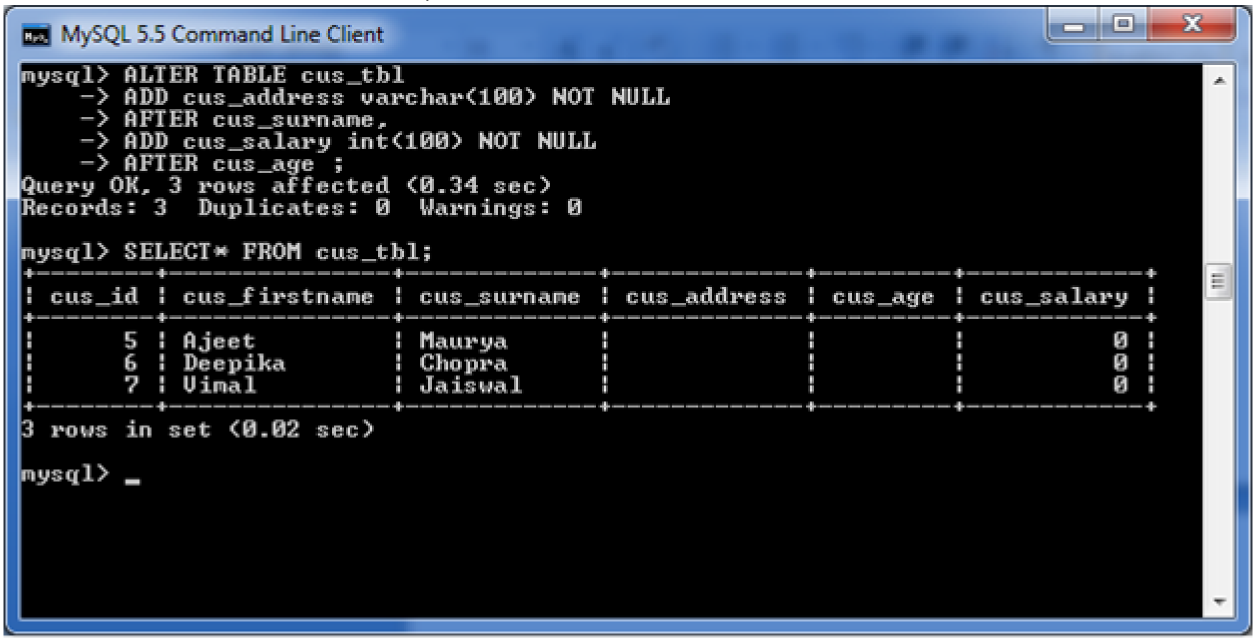
```
ALTER TABLE cus_tbl
ADD cus_address varchar(100) NOT NULL
AFTER cus_surname,
ADD cus_salary int(100) NOT NULL
AFTER cus_age ;
```



```
mysql> ALTER TABLE cus_tbl
    -> ADD cus_address varchar(100) NOT NULL
    -> AFTER cus_surname,
    -> ADD cus_salary int(100) NOT NULL
    -> AFTER cus_age ;
Query OK, 3 rows affected (0.34 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql>
```

See the recently added columns:

1. **SELECT*** **FROM** cus_tbl;



```
mysql> ALTER TABLE cus_tbl
    -> ADD cus_address varchar(100) NOT NULL
    -> AFTER cus_surname,
    -> ADD cus_salary int(100) NOT NULL
    -> AFTER cus_age ;
Query OK, 3 rows affected (0.34 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> SELECT* FROM cus_tbl;
+--------+---------------+-------------+-------------+---------+------------+
| cus_id | cus_firstname | cus_surname | cus_address | cus_age | cus_salary |
+--------+---------------+-------------+-------------+---------+------------+
|      5 | Ajeet         | Maurya      |             |         |          0 |
|      6 | Deepika       | Chopra      |             |         |          0 |
|      7 | Vimal         | Jaiswal     |             |         |          0 |
+--------+---------------+-------------+-------------+---------+------------+
3 rows in set (0.02 sec)

mysql> _
```

3) MODIFY column in the table

The MODIFY command is used to change the column definition of the table.

Syntax:

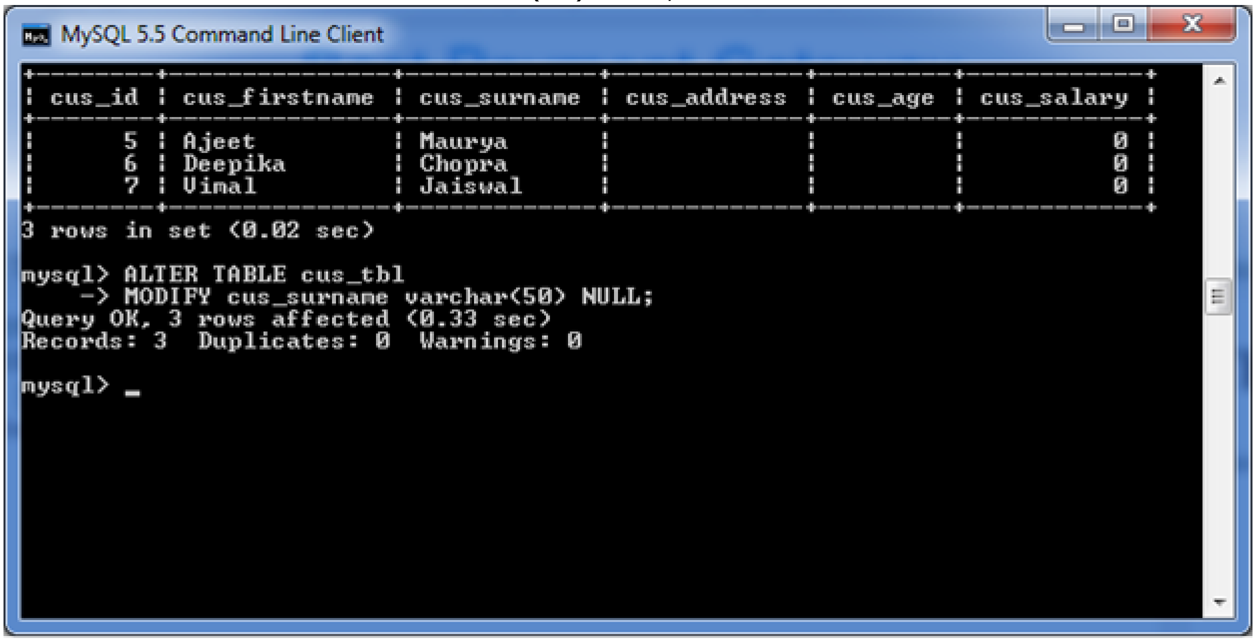
1. **ALTER TABLE** table_name
2. **MODIFY** column_name column_definition
3. [**FIRST** | **AFTER** column_name];

Example:

In this example, we modify the column cus_surname to be a data type of varchar(50) and force the column to allow NULL values.

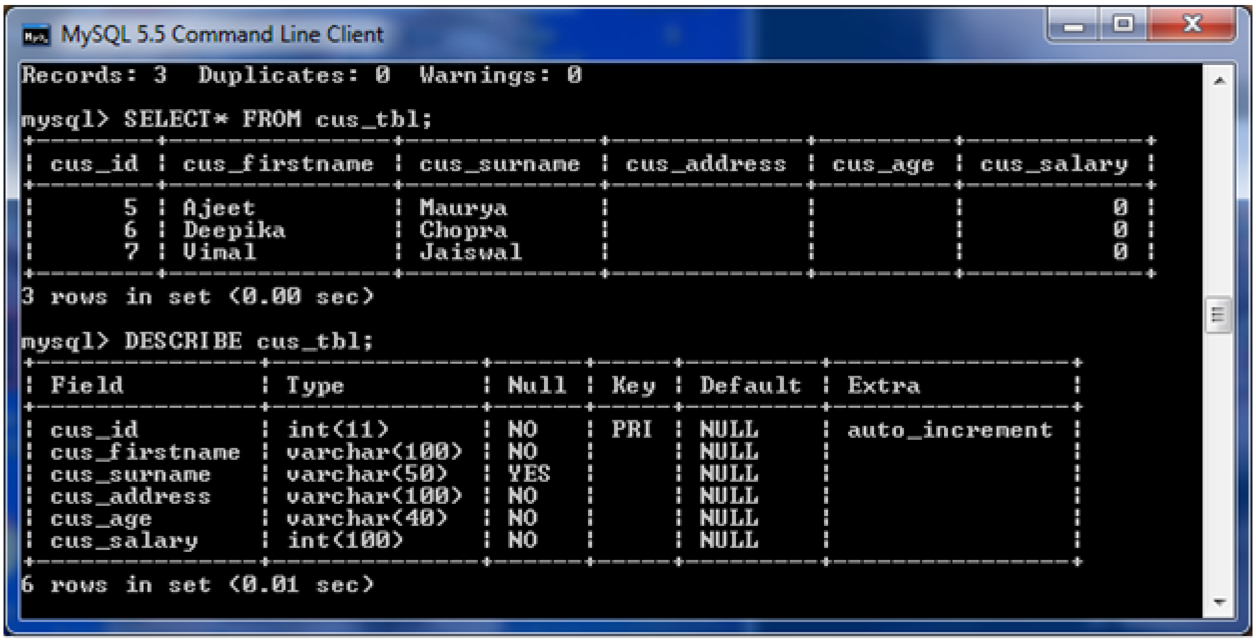
Use the following query to do this:

1. **ALTER TABLE** cus_tbl
2. **MODIFY** cus_surname **varchar**(50) NULL;



```
+--------+---------------+-------------+-------------+---------+------------+
| cus_id | cus_firstname | cus_surname | cus_address | cus_age | cus_salary |
+--------+---------------+-------------+-------------+---------+------------+
|      5 | Ajeet         | Maurya      |             |         |          0 |
|      6 | Deepika       | Chopra      |             |         |          0 |
|      7 | Vimal         | Jaiswal     |             |         |          0 |
+--------+---------------+-------------+-------------+---------+------------+
3 rows in set (0.02 sec)

mysql> ALTER TABLE cus_tbl
    -> MODIFY cus_surname varchar(50) NULL;
Query OK, 3 rows affected (0.33 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> _
```

See the table structure:



```
Records: 3  Duplicates: 0  Warnings: 0

mysql> SELECT* FROM cus_tbl;
+--------+---------------+-------------+-------------+---------+------------+
| cus_id | cus_firstname | cus_surname | cus_address | cus_age | cus_salary |
+--------+---------------+-------------+-------------+---------+------------+
|      5 | Ajeet         | Maurya      |             |         |          0 |
|      6 | Deepika       | Chopra      |             |         |          0 |
|      7 | Vimal         | Jaiswal     |             |         |          0 |
+--------+---------------+-------------+-------------+---------+------------+
3 rows in set (0.00 sec)

mysql> DESCRIBE cus_tbl;
+---------------+--------------+------+-----+---------+----------------+
| Field         | Type         | Null | Key | Default | Extra          |
+---------------+--------------+------+-----+---------+----------------+
| cus_id        | int(11)      | NO   | PRI | NULL    | auto_increment |
| cus_firstname | varchar(100) | NO   |     | NULL    |                |
| cus_surname   | varchar(50)  | YES  |     | NULL    |                |
| cus_address   | varchar(100) | NO   |     | NULL    |                |
| cus_age       | varchar(40)  | NO   |     | NULL    |                |
| cus_salary    | int(100)     | NO   |     | NULL    |                |
+---------------+--------------+------+-----+---------+----------------+
6 rows in set (0.01 sec)
```

4) DROP column in table

Syntax:

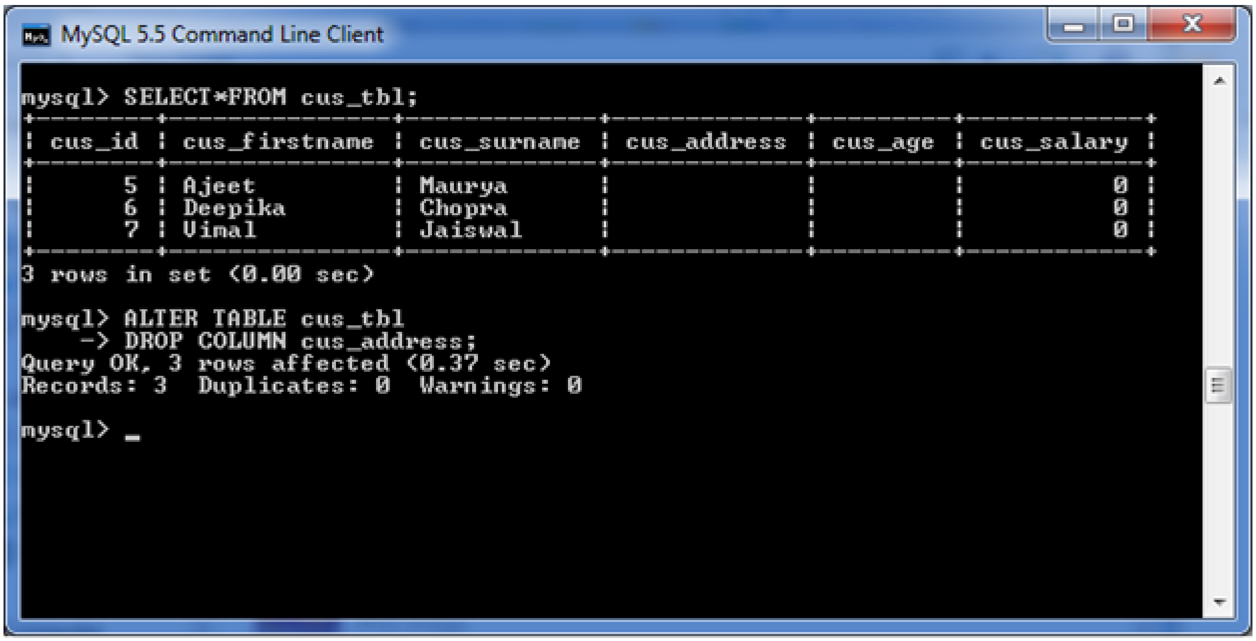
1. **ALTER TABLE** table_name
2. **DROP COLUMN** column_name;

Let's take an example to drop the column name "cus_address" from the table "cus_tbl".

Use the following query to do this:

1. **ALTER TABLE** cus_tbl
2. **DROP COLUMN** cus_address;

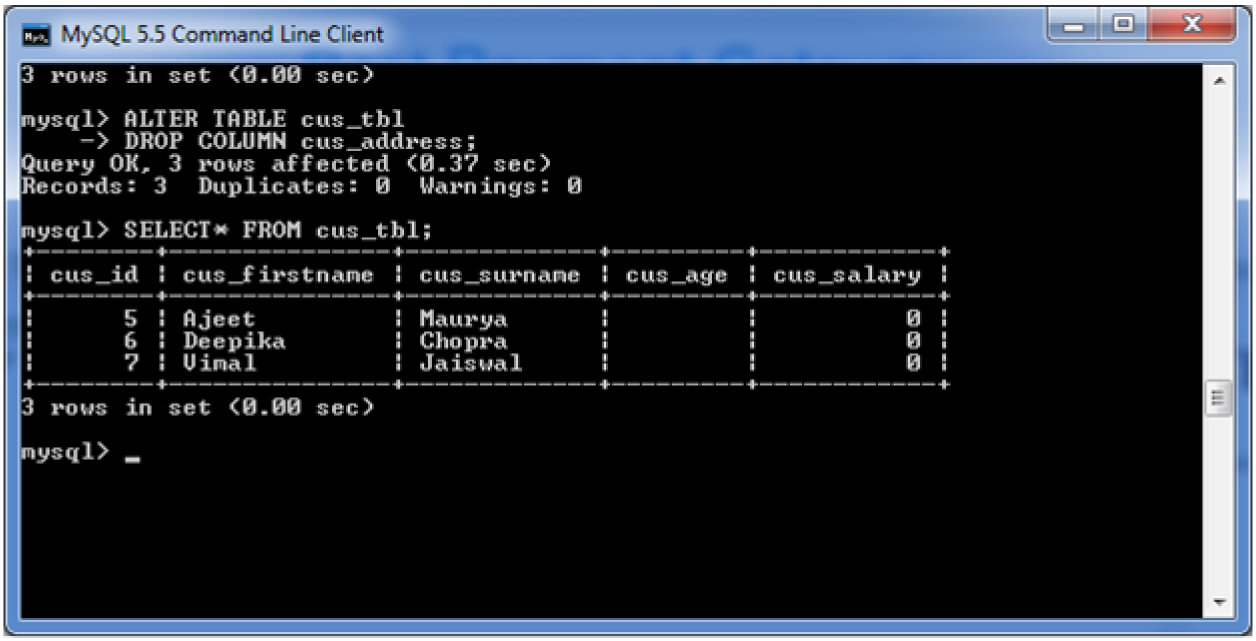
Output:



MySQL 5.5 Command Line Client

```
mysql> SELECT*FROM cus_tbl;
+--------+---------------+-------------+-------------+---------+------------+
| cus_id | cus_firstname | cus_surname | cus_address | cus_age | cus_salary |
+--------+---------------+-------------+-------------+---------+------------+
|      5 | Ajeet         | Maurya      |             |         |          0 |
|      6 | Deepika       | Chopra      |             |         |          0 |
|      7 | Vimal         | Jaiswal     |             |         |          0 |
+--------+---------------+-------------+-------------+---------+------------+
3 rows in set (0.00 sec)

mysql> ALTER TABLE cus_tbl
    -> DROP COLUMN cus_address;
Query OK, 3 rows affected (0.37 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> _
```

See the table structure:



```
3 rows in set (0.00 sec)

mysql> ALTER TABLE cus_tbl
    -> DROP COLUMN cus_address;
Query OK, 3 rows affected (0.37 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> SELECT* FROM cus_tbl;
+--------+---------------+-------------+---------+------------+
| cus_id | cus_firstname | cus_surname | cus_age | cus_salary |
+--------+---------------+-------------+---------+------------+
|      5 | Ajeet         | Maurya      |         |          0 |
|      6 | Deepika       | Chopra      |         |          0 |
|      7 | Vimal         | Jaiswal     |         |          0 |
+--------+---------------+-------------+---------+------------+
3 rows in set (0.00 sec)

mysql> _
```

5) RENAME column in table

Syntax:

```
1. ALTER TABLE table_name
2. CHANGE COLUMN old_name new_name
3. column_definition
4. [ FIRST | AFTER column_name ]
```

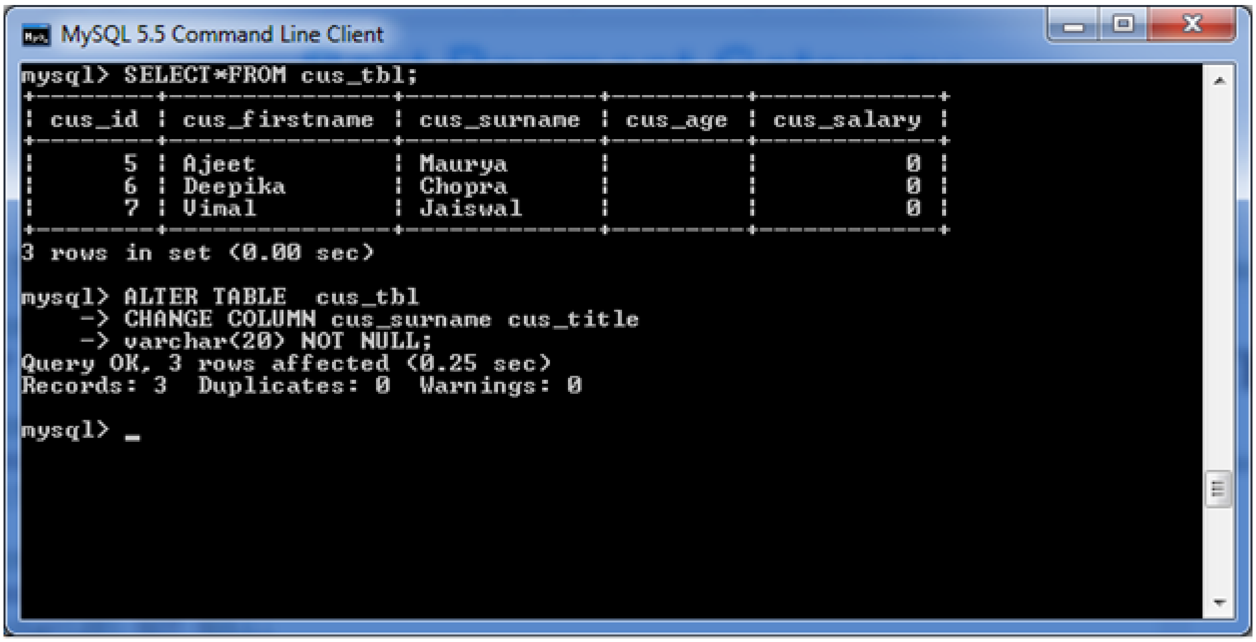
Example:

In this example, we will change the column name "cus_surname" to "cus_title".

Use the following query to do this:

```
1. ALTER TABLE cus_tbl
2. CHANGE COLUMN cus_surname cus_title
3. varchar(20) NOT NULL;
```

Output:



```
mysql> SELECT*FROM cus_tbl;
+--------+---------------+-------------+---------+------------+
| cus_id | cus_firstname | cus_surname | cus_age | cus_salary |
+--------+---------------+-------------+---------+------------+
|      5 | Ajeet         | Maurya      |         |          0 |
|      6 | Deepika       | Chopra      |         |          0 |
|      7 | Vimal         | Jaiswal     |         |          0 |
+--------+---------------+-------------+---------+------------+
3 rows in set (0.00 sec)

mysql> ALTER TABLE  cus_tbl
    -> CHANGE COLUMN cus_surname cus_title
    -> varchar(20) NOT NULL;
Query OK, 3 rows affected (0.25 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> _
```

6) RENAME table

Syntax:

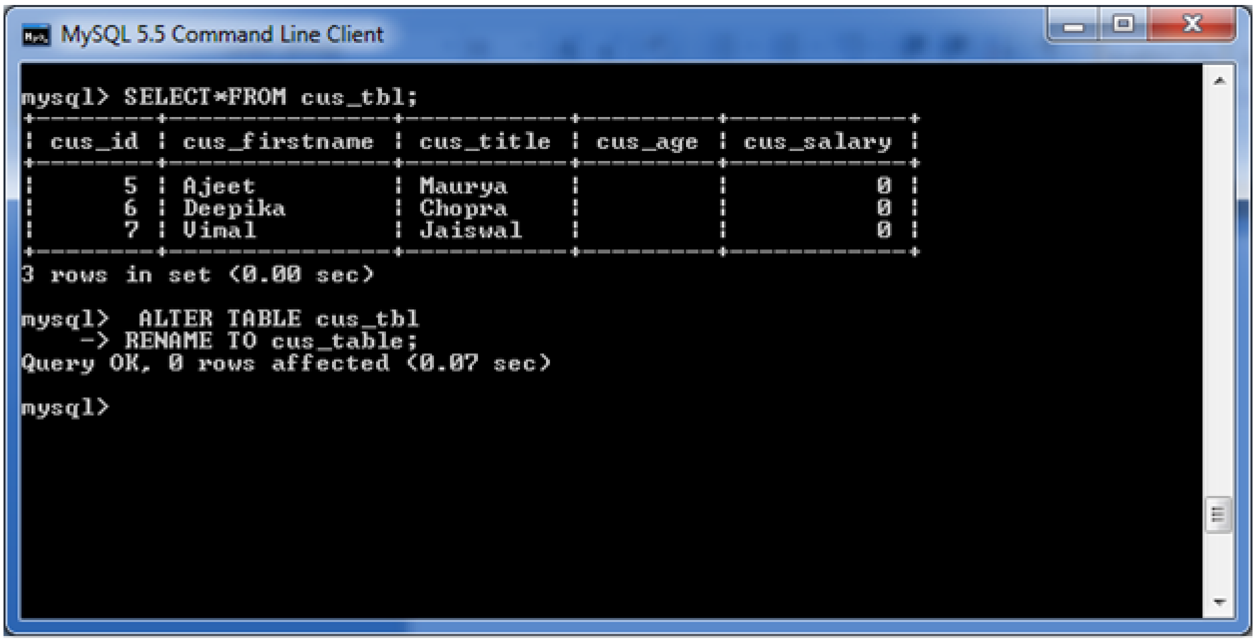
1. **ALTER TABLE** table_name
2. RENAME **TO** new_table_name;

Example:

In this example, the table name cus_tbl is renamed as cus_table.

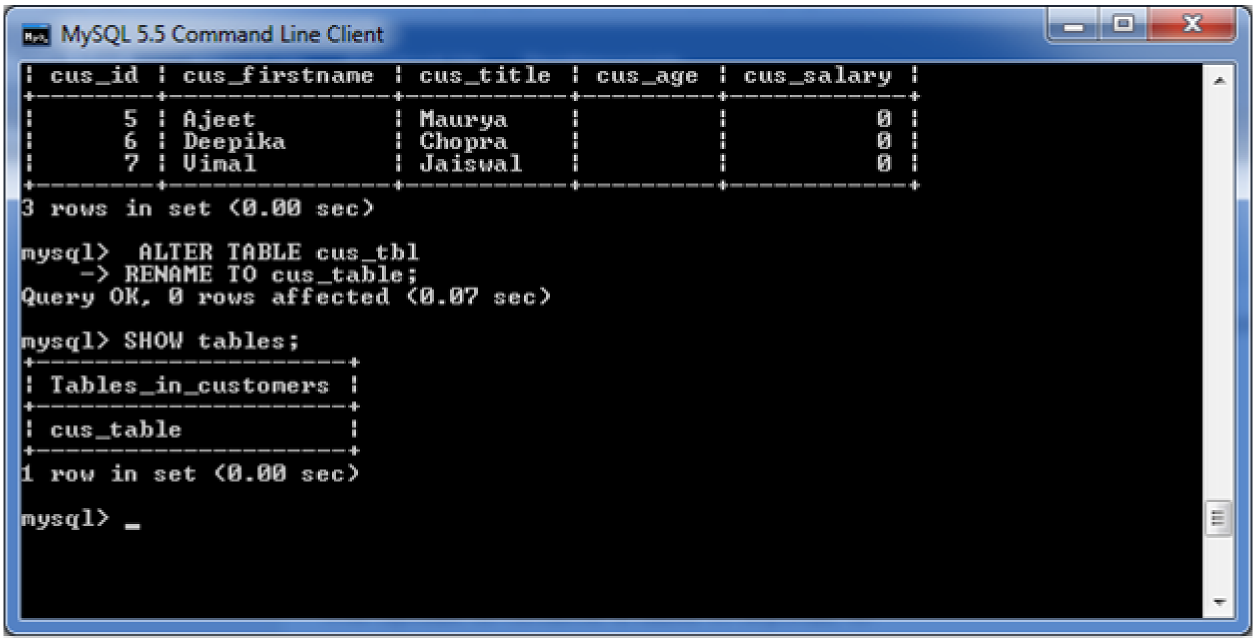
1. **ALTER TABLE** cus_tbl
2. RENAME **TO** cus_table;

Output:



```
mysql> SELECT*FROM cus_tbl;
+--------+---------------+-----------+---------+------------+
| cus_id | cus_firstname | cus_title | cus_age | cus_salary |
+--------+---------------+-----------+---------+------------+
|      5 | Ajeet         | Maurya    |         |          0 |
|      6 | Deepika       | Chopra    |         |          0 |
|      7 | Vimal         | Jaiswal   |         |          0 |
+--------+---------------+-----------+---------+------------+
3 rows in set (0.00 sec)

mysql>  ALTER TABLE cus_tbl
    -> RENAME TO cus_table;
Query OK, 0 rows affected (0.07 sec)

mysql>
```

See the renamed table:



```
| cus_id | cus_firstname | cus_title | cus_age | cus_salary |
+--------+---------------+-----------+---------+------------+
|      5 | Ajeet         | Maurya    |         |          0 |
|      6 | Deepika       | Chopra    |         |          0 |
|      7 | Vimal         | Jaiswal   |         |          0 |
+--------+---------------+-----------+---------+------------+
3 rows in set (0.00 sec)

mysql>  ALTER TABLE cus_tbl
    -> RENAME TO cus_table;
Query OK, 0 rows affected (0.07 sec)

mysql> SHOW tables;
+----------------------+
| Tables_in_customers  |
+----------------------+
| cus_table            |
+----------------------+
1 row in set (0.00 sec)

mysql> _
```

MySQL TRUNCATE Table

MYSQL TRUNCATE statement removes the complete data without removing its structure.

The TRUNCATE TABLE statement is used when you want to delete the complete data from a table without removing the table structure.

Syntax:

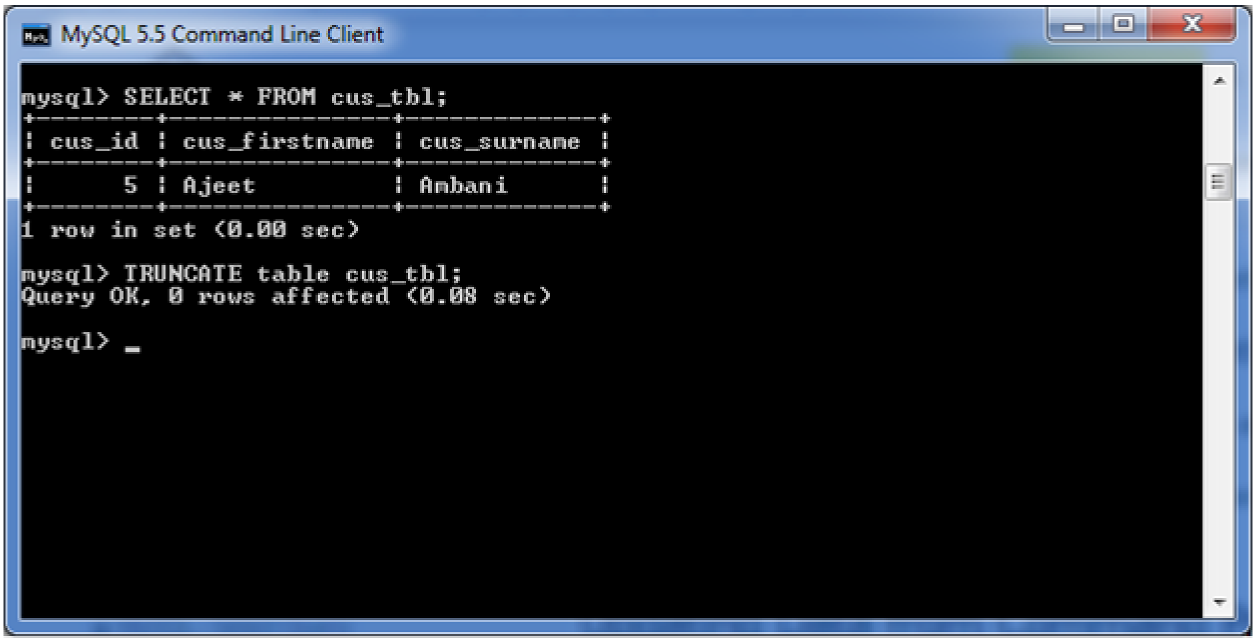
```
1.        TRUNCATE TABLE  table_name;
```

Example:

This example specifies how to truncate a table. In this example, we truncate the table "cus_tbl".

1. **TRUNCATE TABLE** cus_tbl;

Output:

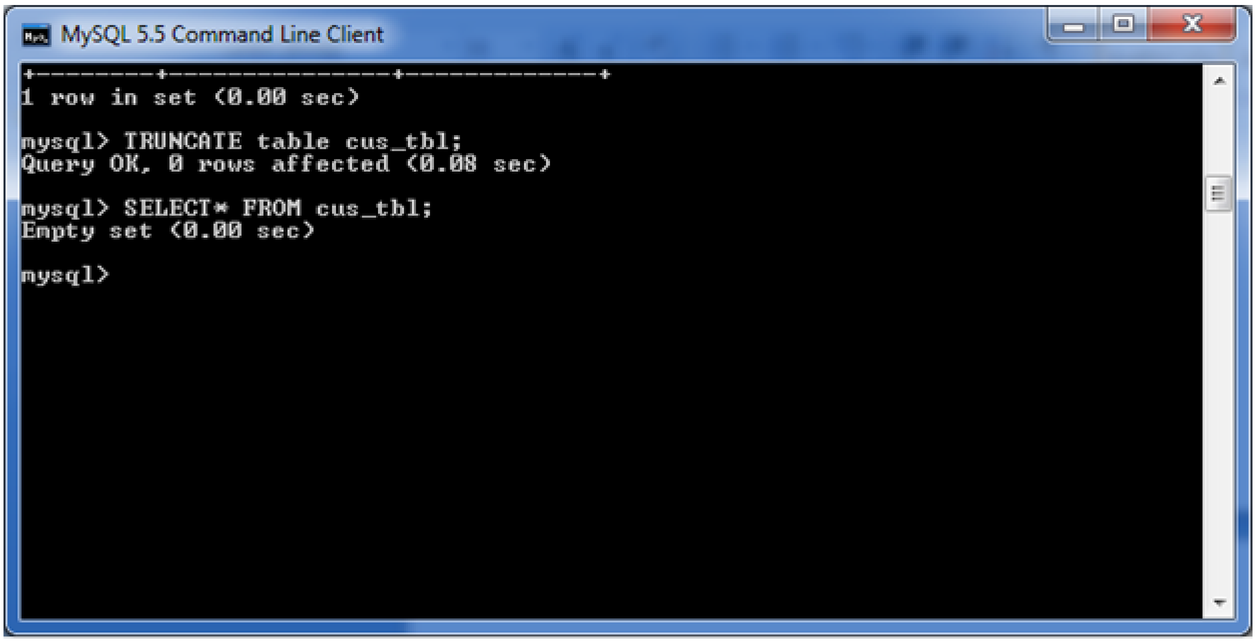


```
mysql> SELECT * FROM cus_tbl;
+--------+---------------+-------------+
| cus_id | cus_firstname | cus_surname |
+--------+---------------+-------------+
|      5 | Ajeet         | Ambani      |
+--------+---------------+-------------+
1 row in set (0.00 sec)

mysql> TRUNCATE table cus_tbl;
Query OK, 0 rows affected (0.08 sec)

mysql> _
```

See the table:

1. **SELECT*** **FROM** cus_tbl;

Output:



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

mysql> TRUNCATE table cus_tbl;
Query OK, 0 rows affected (0.08 sec)

mysql> SELECT* FROM cus_tbl;
Empty set (0.00 sec)

mysql>
```

MySQL DROP Table

MYSQL DROP table statement removes the complete data with structure.

Syntax:

```
1.        DROP TABLE  table_name;
```

Example:

This example specifies how to drop a table. In this example, we are dropping the table "cus_tbl".

```
1.        DROP TABLE  cus_tbl;
```

MySQL View

In MySQL, View is a virtual table created by a query by joining one or more tables.

MySQL Create VIEW

A VIEW is created by SELECT statements. SELECT statements are used to take data from the source table to make a VIEW.

Syntax:

```
1.        CREATE [OR REPLACE] VIEW view_name AS
2.        SELECT columns
3.        FROM tables
4.        [WHERE conditions];
```

Parameters:

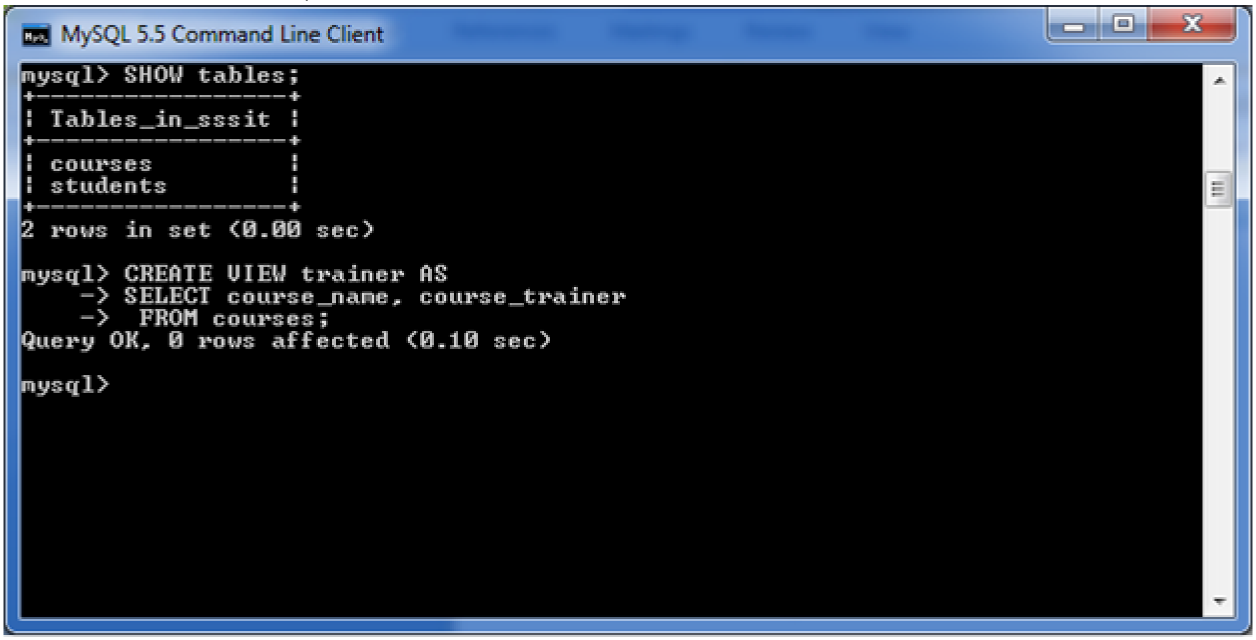
OR REPLACE: It is optional. It is used when a VIEW already exist. If you do not specify this clause and the VIEW already exists, the CREATE VIEW statement will return an error.

view_name: It specifies the name of the VIEW that you want to create in MySQL.

WHERE conditions: It is also optional. It specifies the conditions that must be met for the records to be included in the VIEW.

The following example will create a VIEW name "trainer". This is a virtual table made by taking data from the table "courses".

```
CREATE VIEW trainer AS
SELECT course_name, course_trainer
 FROM courses;
```



```
mysql> SHOW tables;
+-----------------+
| Tables_in_sssit |
+-----------------+
| courses         |
| students        |
+-----------------+
2 rows in set (0.00 sec)

mysql> CREATE VIEW trainer AS
    -> SELECT course_name, course_trainer
    ->  FROM courses;
Query OK, 0 rows affected (0.10 sec)

mysql>
```

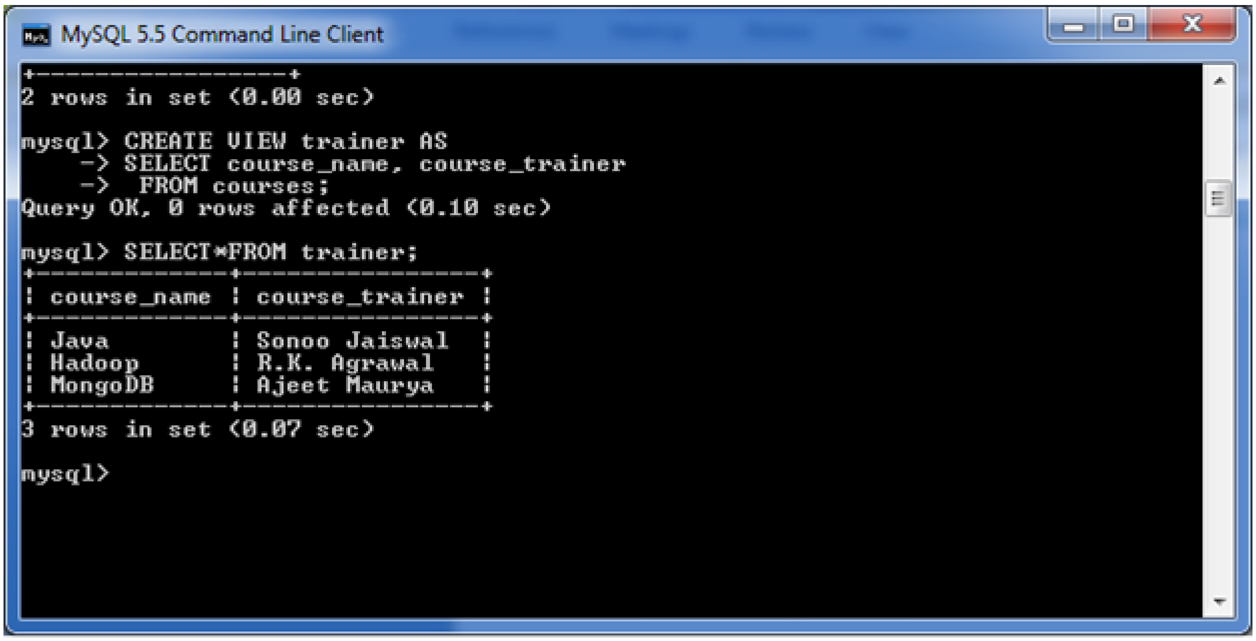
To see the created VIEW:

Syntax:

```
SELECT * FROM view_name;
```

Let's see how it looks the created VIEW:

```
SELECT * FROM trainer;
```



```
+------------------+
2 rows in set (0.00 sec)

mysql> CREATE VIEW trainer AS
    -> SELECT course_name, course_trainer
    ->  FROM courses;
Query OK, 0 rows affected (0.10 sec)

mysql> SELECT*FROM trainer;
+-------------+----------------+
| course_name | course_trainer |
+-------------+----------------+
| Java        | Sonoo Jaiswal  |
| Hadoop      | R.K. Agrawal   |
| MongoDB     | Ajeet Maurya   |
+-------------+----------------+
3 rows in set (0.07 sec)

mysql>
```

MySQL Update VIEW

In MYSQL, the ALTER VIEW statement is used to modify or update the already created VIEW without dropping it.

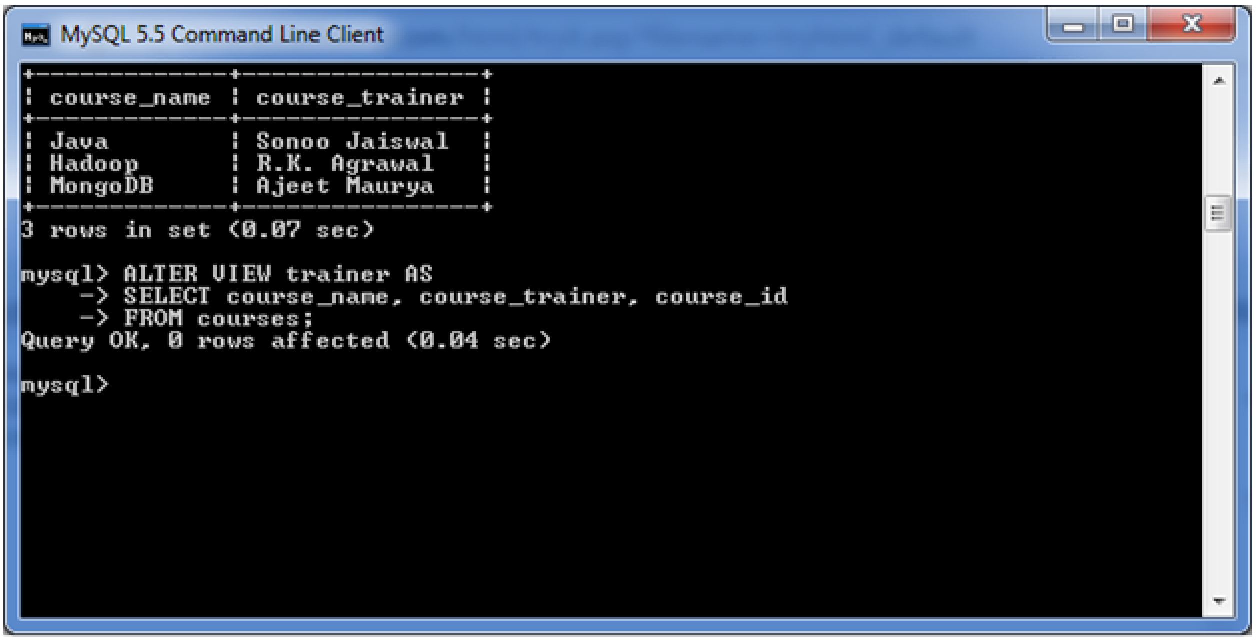
Syntax:

```
ALTER VIEW view_name AS
SELECT columns
FROM table
WHERE conditions;
```

Example:

The following example will alter the already created VIEW name "trainer" by adding a new column.

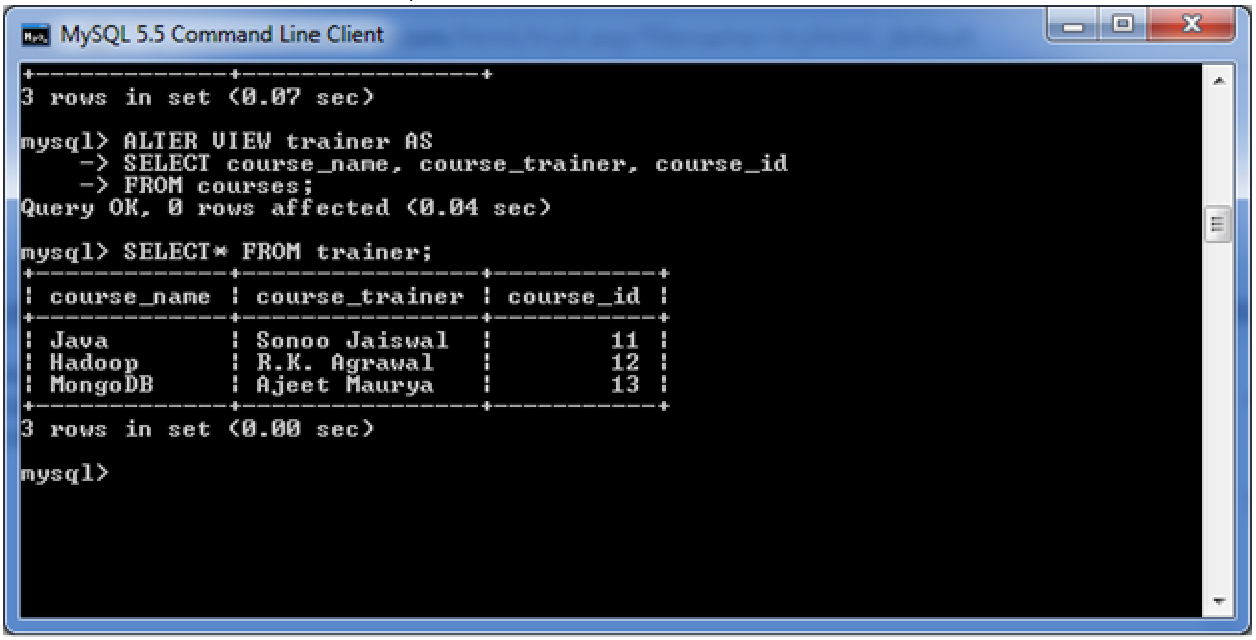
```
ALTER VIEW trainer AS
SELECT course_name, course_trainer, course_id
FROM courses;
```



```
+-------------+-----------------+
| course_name | course_trainer |
+-------------+-----------------+
| Java        | Sonoo Jaiswal   |
| Hadoop      | R.K. Agrawal    |
| MongoDB     | Ajeet Maurya    |
+-------------+-----------------+
3 rows in set (0.07 sec)

mysql> ALTER VIEW trainer AS
    -> SELECT course_name, course_trainer, course_id
    -> FROM courses;
Query OK, 0 rows affected (0.04 sec)

mysql>
```

To see the altered VIEW:

1. **SELECT*****FROM** trainer;



```
+-------------+-----------------+
3 rows in set (0.07 sec)

mysql> ALTER VIEW trainer AS
    -> SELECT course_name, course_trainer, course_id
    -> FROM courses;
Query OK, 0 rows affected (0.04 sec)

mysql> SELECT* FROM trainer;
+-------------+-----------------+-----------+
| course_name | course_trainer  | course_id |
+-------------+-----------------+-----------+
| Java        | Sonoo Jaiswal   |        11 |
| Hadoop      | R.K. Agrawal    |        12 |
| MongoDB     | Ajeet Maurya    |        13 |
+-------------+-----------------+-----------+
3 rows in set (0.00 sec)

mysql>
```

MySQL Drop VIEW

You can drop the VIEW by using the DROP VIEW statement.

Syntax:

1. **DROP VIEW** [IF EXISTS] view_name;

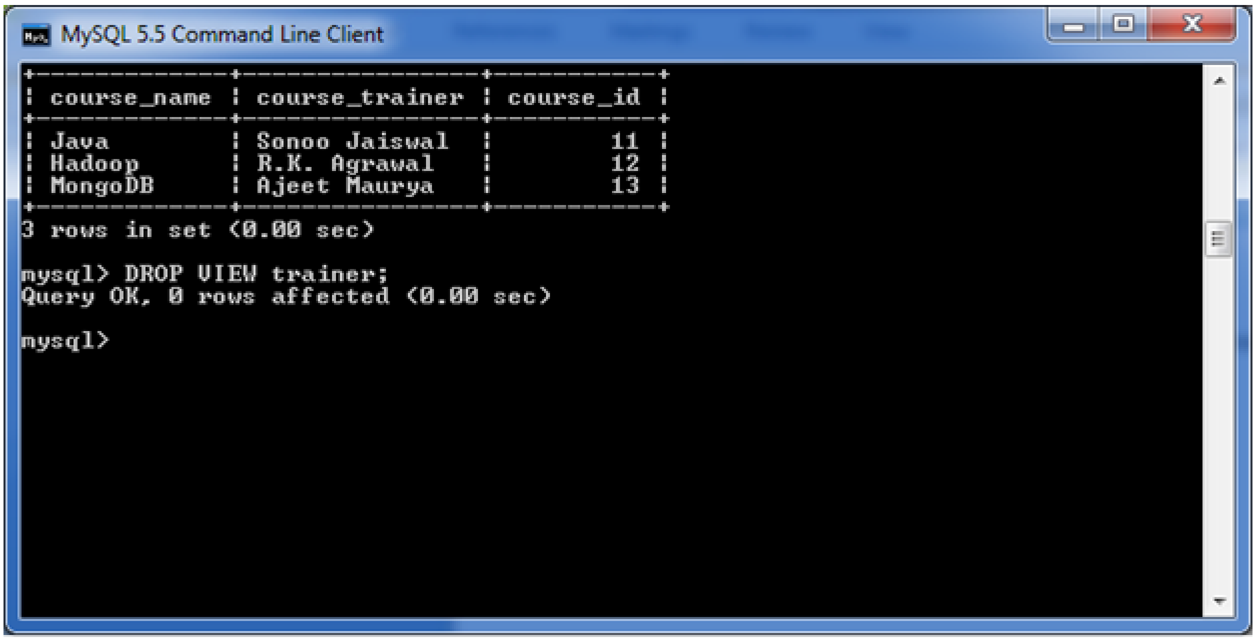
Parameters:

view_name: It specifies the name of the VIEW that you want to drop.

IF EXISTS: It is optional. If you do not specify this clause and the VIEW doesn't exist, the DROP VIEW statement will return an error.

Example:

1. **DROP VIEW** trainer;



```
+-------------+----------------+-----------+
| course_name | course_trainer | course_id |
+-------------+----------------+-----------+
| Java        | Sonoo Jaiswal  |        11 |
| Hadoop      | R.K. Agrawal   |        12 |
| MongoDB     | Ajeet Maurya   |        13 |
+-------------+----------------+-----------+
3 rows in set (0.00 sec)

mysql> DROP VIEW trainer;
Query OK, 0 rows affected (0.00 sec)

mysql>
```

MySQL Queries

A list of commonly used MySQL queries to create database, use database, create table, insert record, update record, delete record, select record, truncate table and drop table are given below.

1) MySQL Create Database

MySQL create database is used to create database. For example

1. **create database** db1;

More Details...

2) MySQL Select/Use Database

MySQL use database is used to select database. For example

1. use db1;

More Details...

3) MySQL Create Query

MySQL create query is used to create a table, view, procedure and function. For example:

```
CREATE TABLE customers
(id int(10),
 name varchar(50),
 city varchar(50),
 PRIMARY KEY (id )
);
```

More Details...

4) MySQL Alter Query

MySQL alter query is used to add, modify, delete or drop colums of a table. Let's see a query to add column in customers table:

```
ALTER TABLE customers
ADD age varchar(50);
```

More Details...

5) MySQL Insert Query

MySQL insert query is used to insert records into table. For example:

```
insert into customers values(101,'rahul','delhi');
```

More Details...

6) MySQL Update Query

MySQL update query is used to update records of a table. For example:

```
update customers set name='bob', city='london' where id=101;
```

More Details...

7) MySQL Delete Query

MySQL update query is used to delete records of a table from database. For example:

```
delete from customers where id=101;
```

More Details...

8) MySQL Select Query

Oracle select query is used to fetch records from database. For example:

```
SELECT * from customers;
```

More Details...

9) MySQL Truncate Table Query

MySQL update query is used to truncate or remove records of a table. It doesn't remove structure. For example:

```
truncate table customers;
```

More Details...

10) MySQL Drop Query

MySQL drop query is used to drop a table, view or database. It removes structure and data of a table if you drop table. For example:

```
drop table customers;
```

MySQL INSERT Statement

MySQL INSERT statement is used to insert data in MySQL table within the database. We can insert single or multiple records using a single query in MySQL.

Syntax:

The SQL INSERT INTO command is used to insert data in MySQL table. Following is a generic syntax:

```
INSERT INTO table_name ( field1, field2,...fieldN )
VALUES
( value1, value2,...valueN );
```

Field name is optional. If you want to specify partial values, field name is mandatory.

Syntax for all fields:

```
INSERT INTO table_name VALUES ( value1, value2,...valueN );
```

MySQL INSERT Example 1: for all fields

If you have to store all the field values, either specify all field name or don't specify any field.

Example:

```
INSERT INTO emp VALUES (7, 'Sonoo', 40000);
```

Or,

```
INSERT INTO emp(id,name,salary) VALUES (7, 'Sonoo', 40000);
```

MySQL INSERT Example 2: for partial fields

In such case, it is mandatory to specify field names.

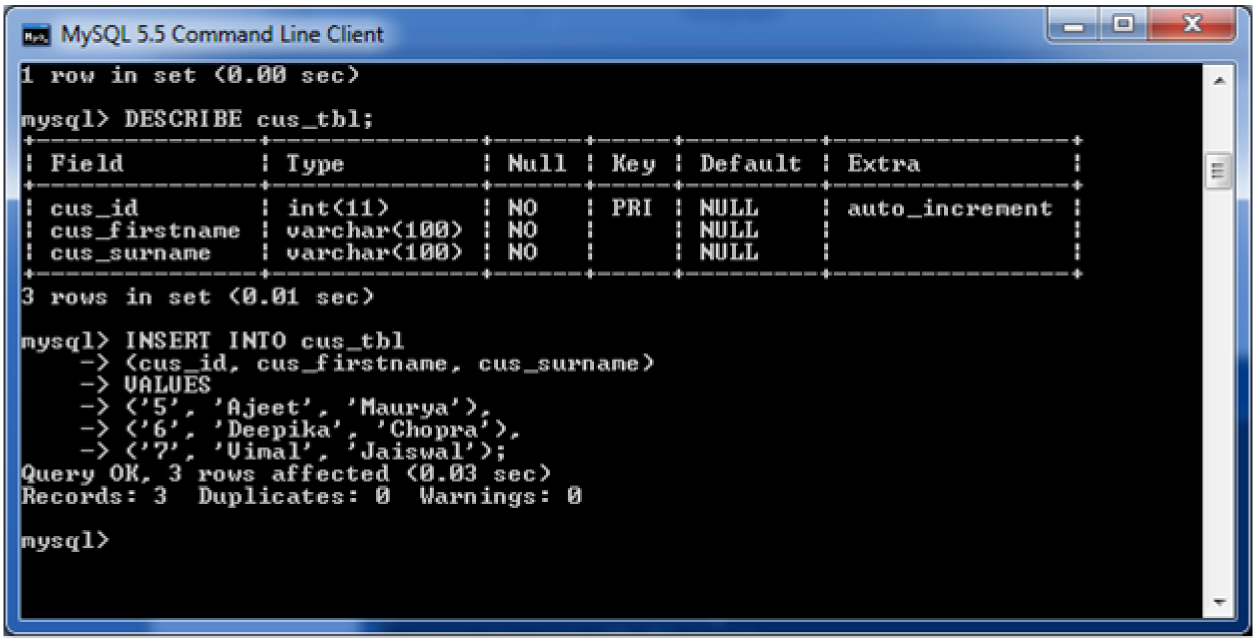
```
INSERT INTO emp(id,name) VALUES (7, 'Sonoo');
```

MySQL INSERT Example 3: inserting multiple records

Here, we are going to insert record in the "cus_tbl" table of "customers" database.

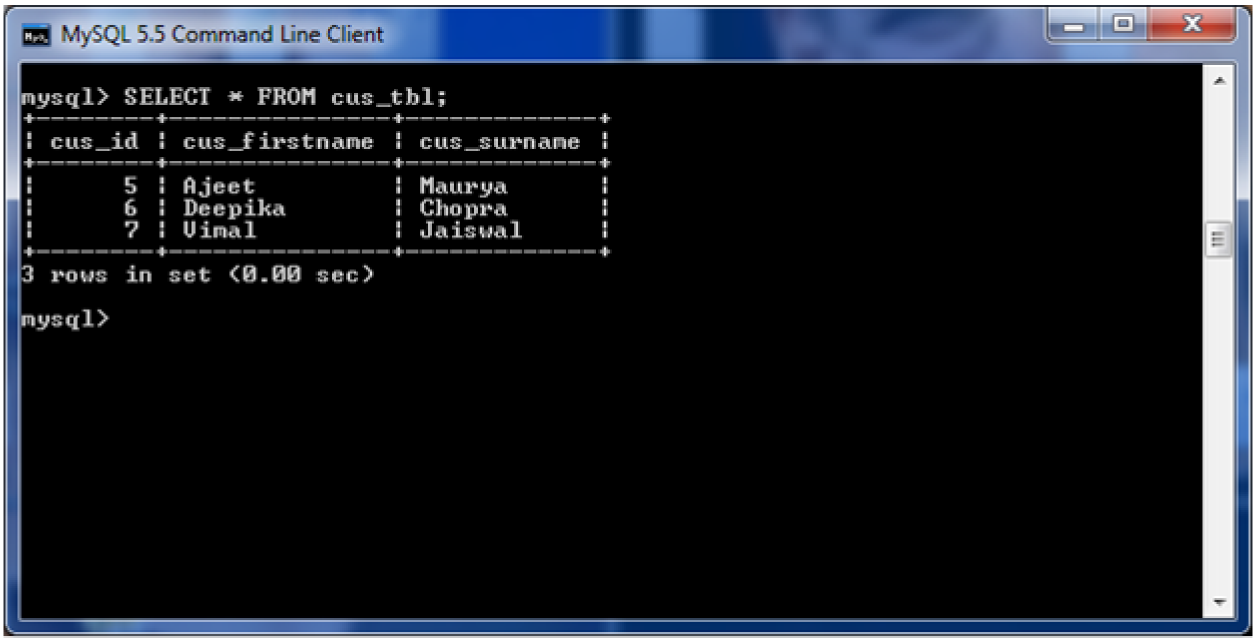
```
 INSERT INTO cus_tbl
(cus_id, cus_firstname, cus_surname)
VALUES
(5, 'Ajeet', 'Maurya'),
(6, 'Deepika', 'Chopra'),
(7, 'Vimal', 'Jaiswal');
```

Visual Representation:



```
1 row in set (0.00 sec)

mysql> DESCRIBE cus_tbl;
+----------------+--------------+------+-----+---------+----------------+
| Field          | Type         | Null | Key | Default | Extra          |
+----------------+--------------+------+-----+---------+----------------+
| cus_id         | int(11)      | NO   | PRI | NULL    | auto_increment |
| cus_firstname  | varchar(100) | NO   |     | NULL    |                |
| cus_surname    | varchar(100) | NO   |     | NULL    |                |
+----------------+--------------+------+-----+---------+----------------+
3 rows in set (0.01 sec)

mysql> INSERT INTO cus_tbl
    -> (cus_id, cus_firstname, cus_surname)
    -> VALUES
    -> ('5', 'Ajeet', 'Maurya'),
    -> ('6', 'Deepika', 'Chopra'),
    -> ('7', 'Vimal', 'Jaiswal');
Query OK, 3 rows affected (0.03 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql>
```

See the data within the table by using the SELECT command:



```
mysql> SELECT * FROM cus_tbl;
+--------+---------------+-------------+
| cus_id | cus_firstname | cus_surname |
+--------+---------------+-------------+
|      5 | Ajeet         | Maurya      |
|      6 | Deepika       | Chopra      |
|      7 | Vimal         | Jaiswal     |
+--------+---------------+-------------+
3 rows in set (0.00 sec)

mysql>
```

MySQL UPDATE Query

MySQL UPDATE statement is used to update data of the MySQL table within the database. It is used when you need to modify the table.

Syntax:

Following is a generic syntax of UPDATE command to modify data into the MySQL table:

```
1.    UPDATE table_name SET field1=new-value1, field2=new-value2
2.    [WHERE Clause]
```

Note:

- One or more field can be updated altogether.
- Any condition can be specified by using WHERE clause.
- You can update values in a single table at a time.
- WHERE clause is used to update selected rows in a table.

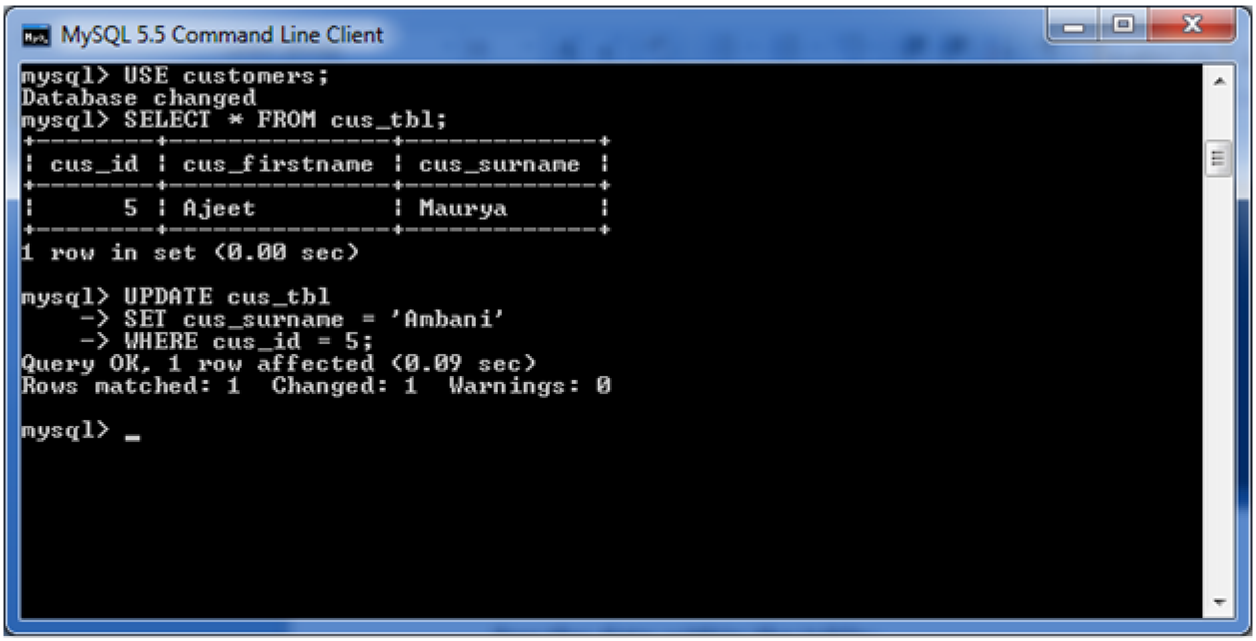
Example:

Here, we have a table "cus_tbl" within the database "customers". We are going to update the data within the table "cus_tbl".

This query will update cus_surname field for a record having cus_id as 5.

```
UPDATE cus_tbl
SET cus_surname = 'Ambani'
WHERE cus_id = 5;
```

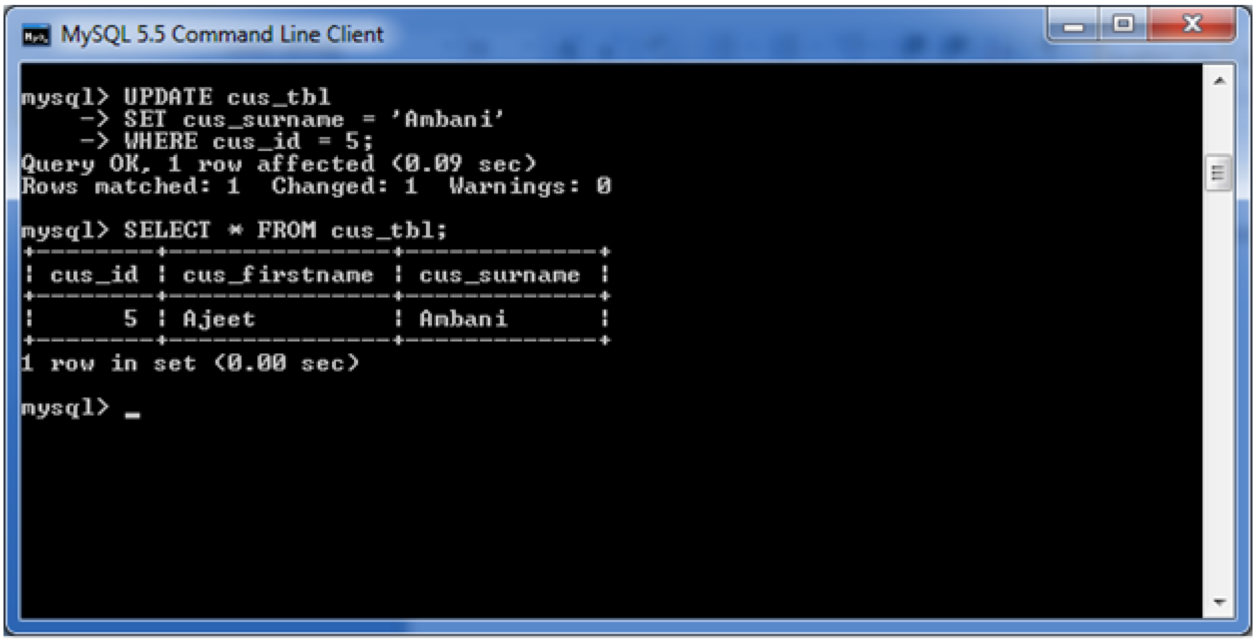
Visual Representation:



```
mysql> USE customers;
Database changed
mysql> SELECT * FROM cus_tbl;
+--------+---------------+-------------+
| cus_id | cus_firstname | cus_surname |
+--------+---------------+-------------+
|      5 | Ajeet         | Maurya      |
+--------+---------------+-------------+
1 row in set (0.00 sec)

mysql> UPDATE cus_tbl
    -> SET cus_surname = 'Ambani'
    -> WHERE cus_id = 5;
Query OK, 1 row affected (0.09 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> _
```

Output by SELECT query:

```
SELECT * FROM cus_tbl;
```



```
mysql> UPDATE cus_tbl
    -> SET cus_surname = 'Ambani'
    -> WHERE cus_id = 5;
Query OK, 1 row affected (0.09 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> SELECT * FROM cus_tbl;
+--------+---------------+-------------+
| cus_id | cus_firstname | cus_surname |
+--------+---------------+-------------+
|      5 | Ajeet         | Ambani      |
+--------+---------------+-------------+
1 row in set (0.00 sec)

mysql> _
```

Here, you can see that the table is updated as per your conditions.

MySQL DELETE Statement

MySQL DELETE statement is used to delete data from the MySQL table within the database. By using delete statement, we can delete records on the basis of conditions.

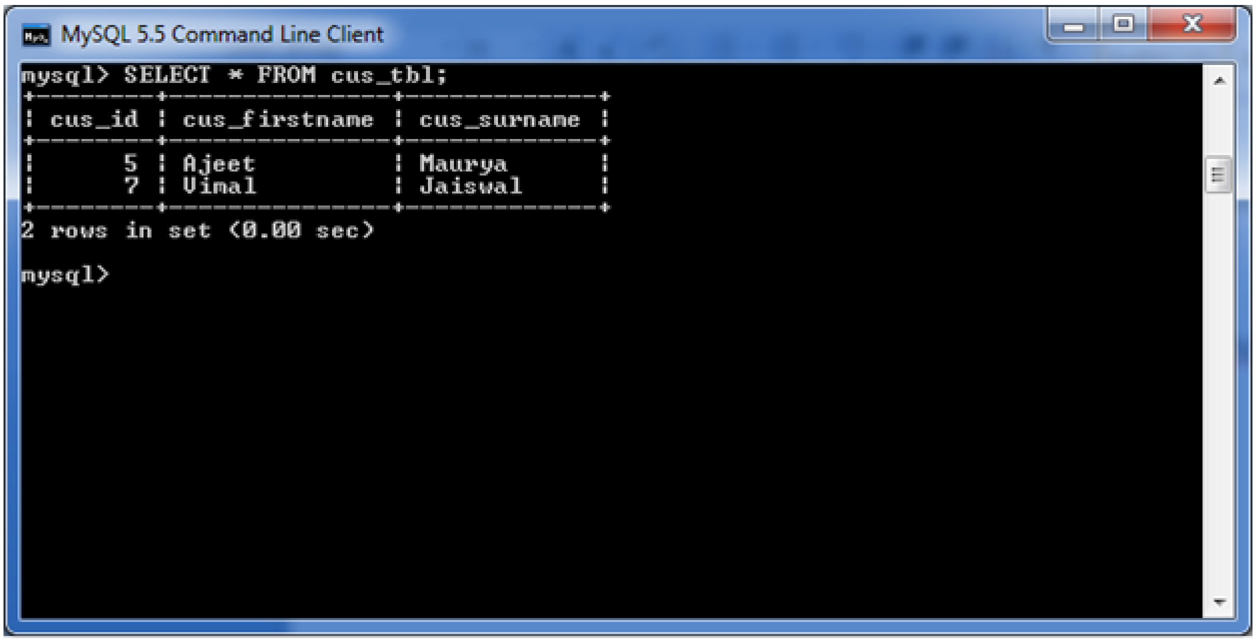
Syntax:

1. **DELETE FROM** table_name
2. **WHERE**
3. (Condition specified);

Example:

1. **DELETE FROM** cus_tbl
2. **WHERE** cus_id = 6;

Output:



```
mysql> SELECT * FROM cus_tbl;
+--------+----------------+--------------+
| cus_id | cus_firstname  | cus_surname  |
+--------+----------------+--------------+
|      5 | Ajeet          | Maurya       |
|      7 | Vimal          | Jaiswal      |
+--------+----------------+--------------+
2 rows in set (0.00 sec)

mysql>
```

MySQL SELECT Statement

The MySQL SELECT statement is used to fetch data from the one or more tables in MySQL. We can retrieve records of all fields or specified fields.

Syntax for specified fields:

1. **SELECT** expressions
2. **FROM** tables
3. [**WHERE** conditions];

Syntax for all fields:

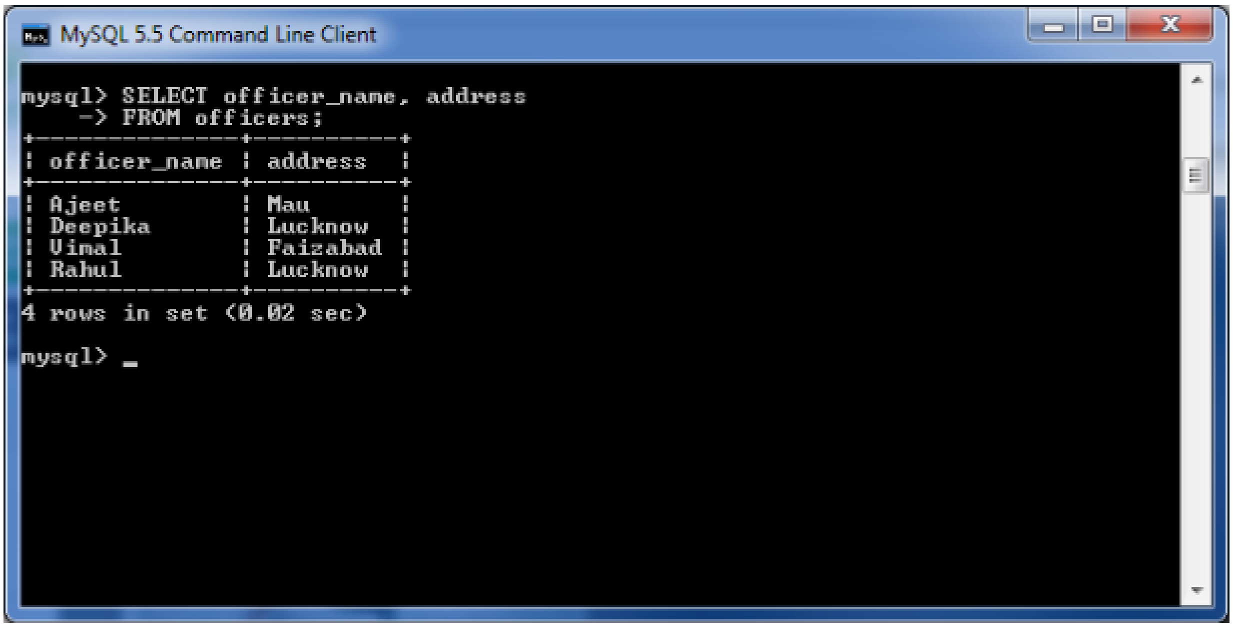
1. **SELECT** * **FROM** tables [**WHERE** conditions];

MySQL SELECT Example 1: for specified fields

In such case, it is mandatory to specify field names.

Example:

1. **SELECT** officer_name, address
2. **FROM** officers

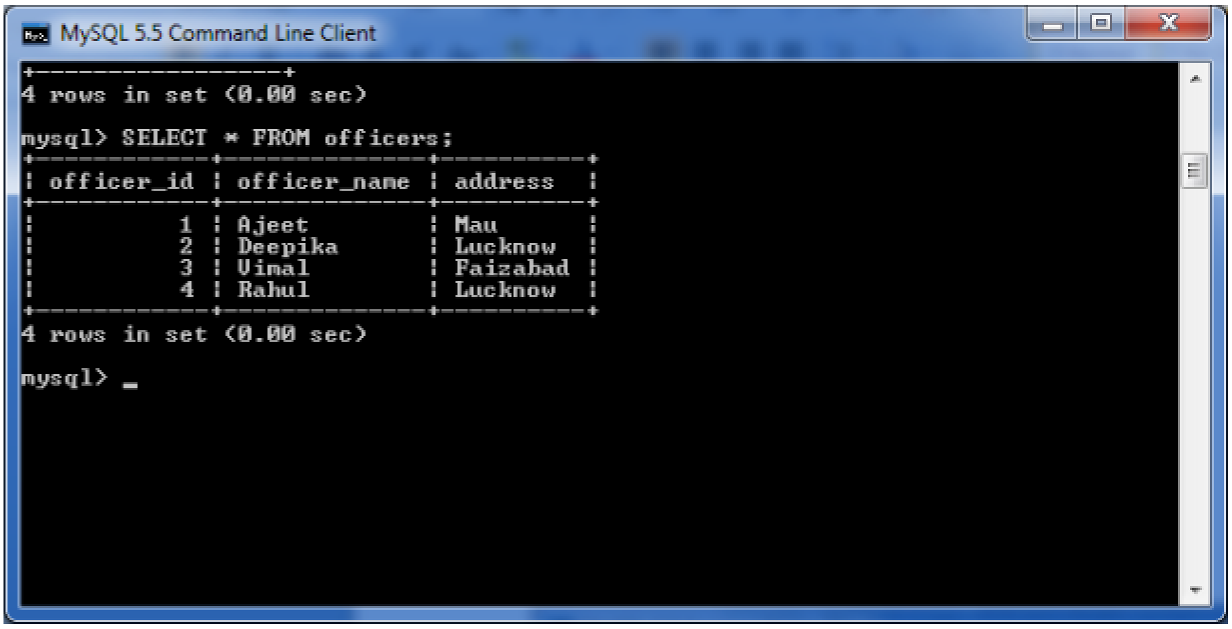


```
mysql> SELECT officer_name, address
    -> FROM officers;
+--------------+----------+
| officer_name | address  |
+--------------+----------+
| Ajeet        | Mau      |
| Deepika      | Lucknow  |
| Vimal        | Faizabad |
| Rahul        | Lucknow  |
+--------------+----------+
4 rows in set (0.02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

mysql> _
```

MySQL SELECT Example 2: for all fields

In such case, we can specify either all fields or * (asterisk) symbol.

1. **SELECT** * **FROM** officers

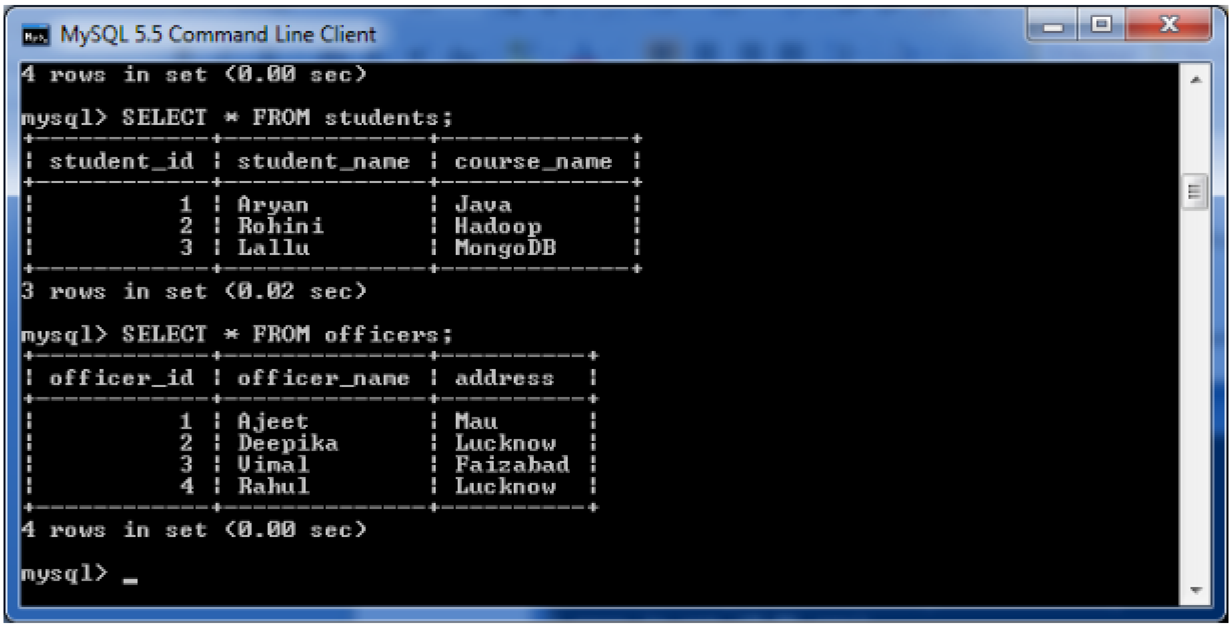


```
+------------------+
4 rows in set (0.00 sec)

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 (0.00 sec)

mysql> _
```

MySQL SELECT Example 3: from multiple tables

MySQL SELECT statement can also be used to retrieve records from multiple tables by using JOIN statement.

Let's take two tables "students" and "officers", having the following data.

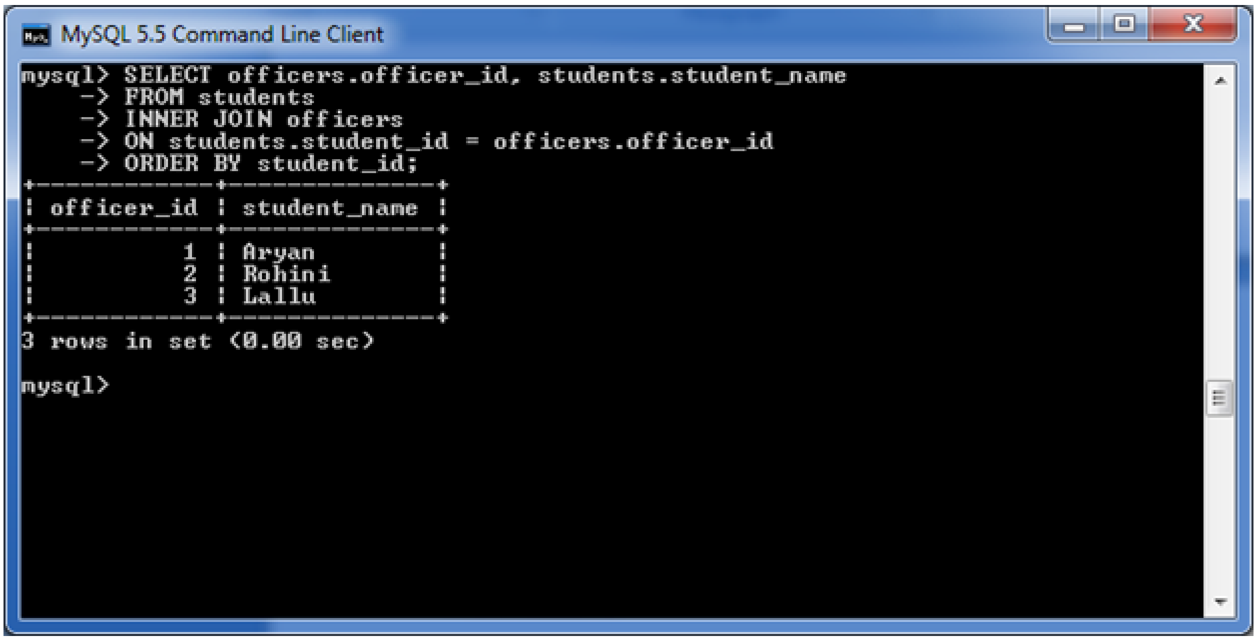


```
4 rows in set (0.00 sec)

mysql> SELECT * FROM students;
+------------+--------------+-------------+
| student_id | student_name | course_name |
+------------+--------------+-------------+
|          1 | Aryan        | Java        |
|          2 | Rohini       | Hadoop      |
|          3 | Lallu        | MongoDB     |
+------------+--------------+-------------+
3 rows in set (0.02 sec)

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 (0.00 sec)

mysql> _
```

Execute the following query:

```
SELECT officers.officer_id, students.student_name
FROM students
INNER JOIN officers
ON students.student_id = officers.officer_id
ORDER BY student_id;
```

Output:



```
mysql> SELECT officers.officer_id, students.student_name
    -> FROM students
    -> INNER JOIN officers
    -> ON students.student_id = officers.officer_id
    -> ORDER BY student_id;
+------------+--------------+
| officer_id | student_name |
+------------+--------------+
|          1 | Aryan        |
|          2 | Rohini       |
|          3 | Lallu        |
+------------+--------------+
3 rows in set (0.00 sec)

mysql>
```

MySQL WHERE Clause

MySQL WHERE Clause is used with SELECT, INSERT, UPDATE and DELETE clause to filter the results. It specifies a specific position where you have to do the operation.

Syntax:

```
WHERE conditions;
```

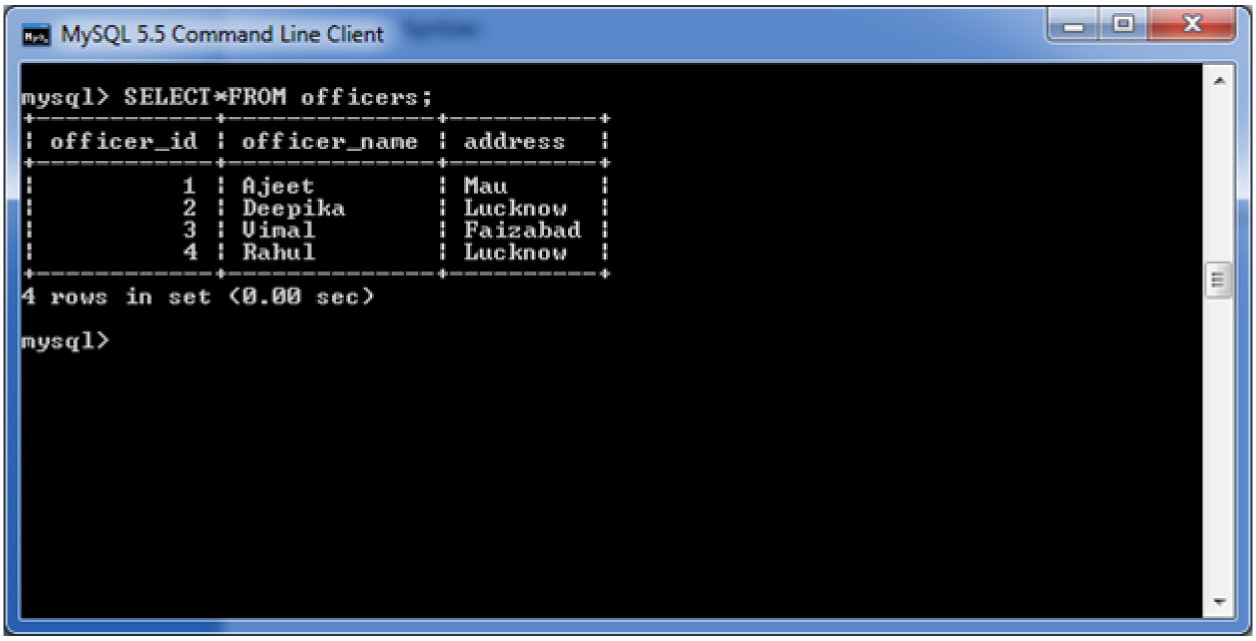
Parameter:

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

MySQL WHERE Clause with single condition

Let's take an example to retrieve data from a table "officers".

Table structure:

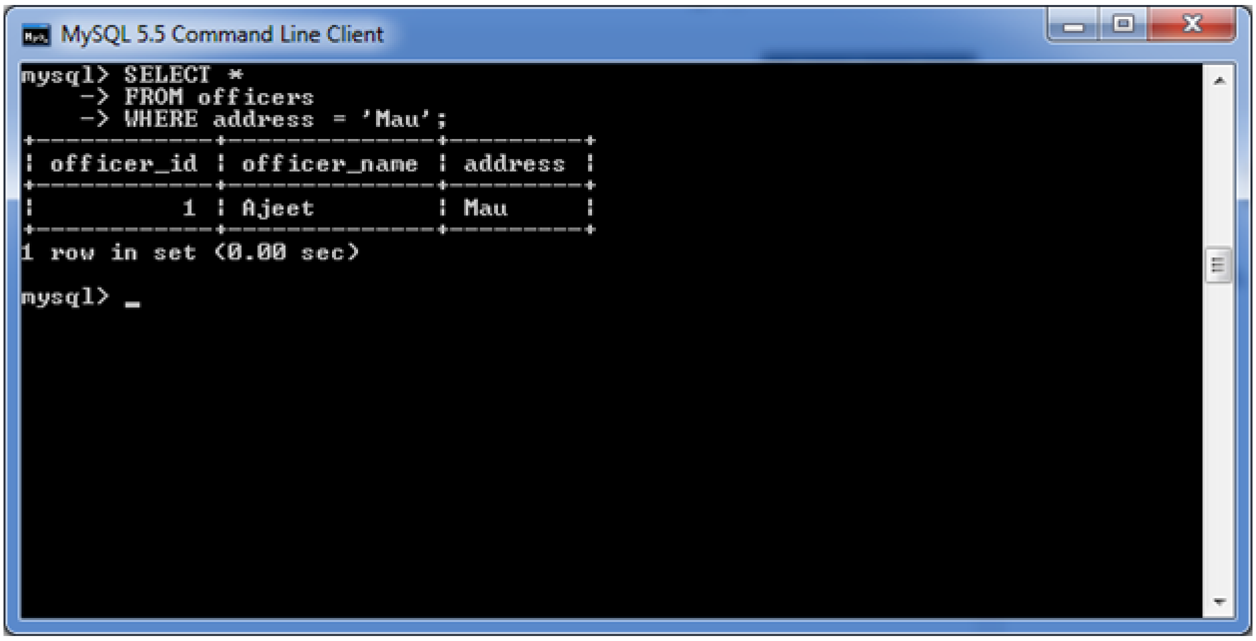


```
mysql> SELECT*FROM officers;
+------------+--------------+----------+
| officer_id | officer_name | address  |
+------------+--------------+----------+
|          1 | Ajeet        | Mau      |
|          2 | Deepika      | Lucknow  |
|          3 | Vimal        | Faizabad |
|          4 | Rahul        | Lucknow  |
+------------+--------------+----------+
4 rows in set (0.00 sec)

mysql>
```

Execute this query:

```
SELECT *
FROM officers
WHERE address = 'Mau';
```

Output:



```
mysql> SELECT *
    -> FROM officers
    -> WHERE address = 'Mau';
+------------+--------------+---------+
| officer_id | officer_name | address |
+------------+--------------+---------+
|          1 | Ajeet        | Mau     |
+------------+--------------+---------+
1 row in set (0.00 sec)

mysql> _
```

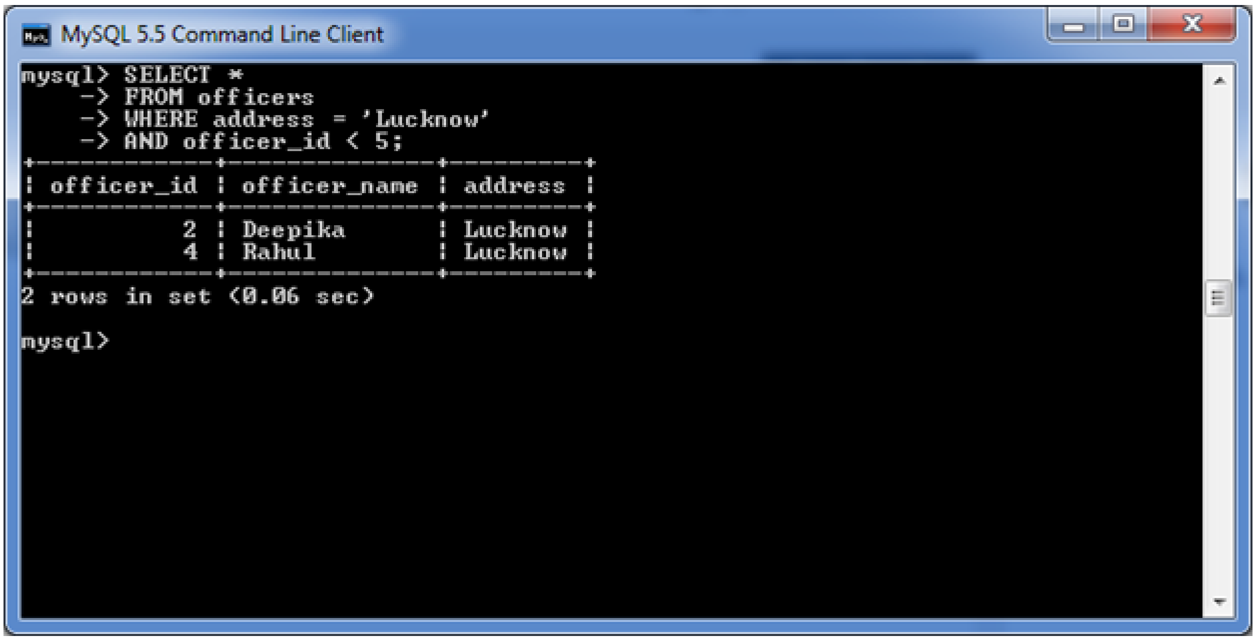
MySQL WHERE Clause with AND condition

In this example, we are retrieving data from the table "officers" with AND condition.

Execute the following query:

```
SELECT *
FROM officers
WHERE address = 'Lucknow'
AND officer_id < 5;
```

Output:



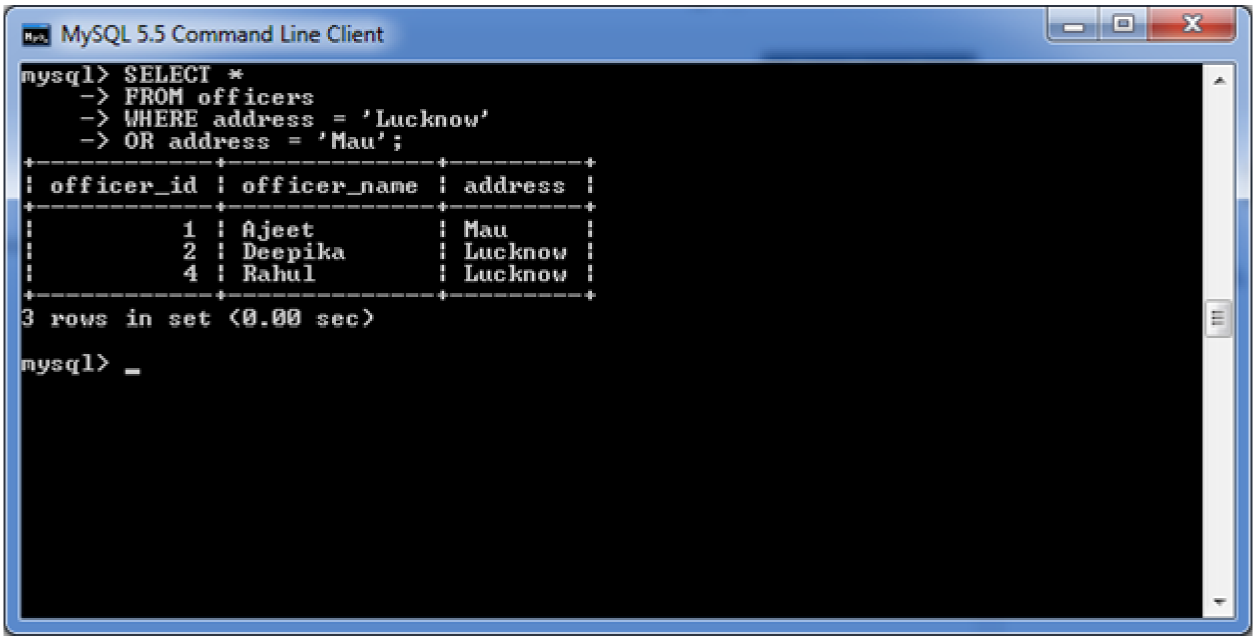
```
mysql> SELECT *
    -> FROM officers
    -> WHERE address = 'Lucknow'
    -> AND officer_id < 5;
+------------+--------------+---------+
| officer_id | officer_name | address |
+------------+--------------+---------+
|          2 | Deepika      | Lucknow |
|          4 | Rahul        | Lucknow |
+------------+--------------+---------+
2 rows in set (0.06 sec)

mysql>
```

WHERE Clause with OR condition

Execute the following query:

```
SELECT *
FROM officers
WHERE address = 'Lucknow'
OR address = 'Mau';
```

Output:



```
mysql> SELECT *
    -> FROM officers
    -> WHERE address = 'Lucknow'
    -> OR address = 'Mau';
+------------+--------------+---------+
| officer_id | officer_name | address |
+------------+--------------+---------+
|          1 | Ajeet        | Mau     |
|          2 | Deepika      | Lucknow |
|          4 | Rahul        | Lucknow |
+------------+--------------+---------+
3 rows in set (0.00 sec)

mysql> _
```

MySQL WHERE Clause with combination of AND & OR conditions

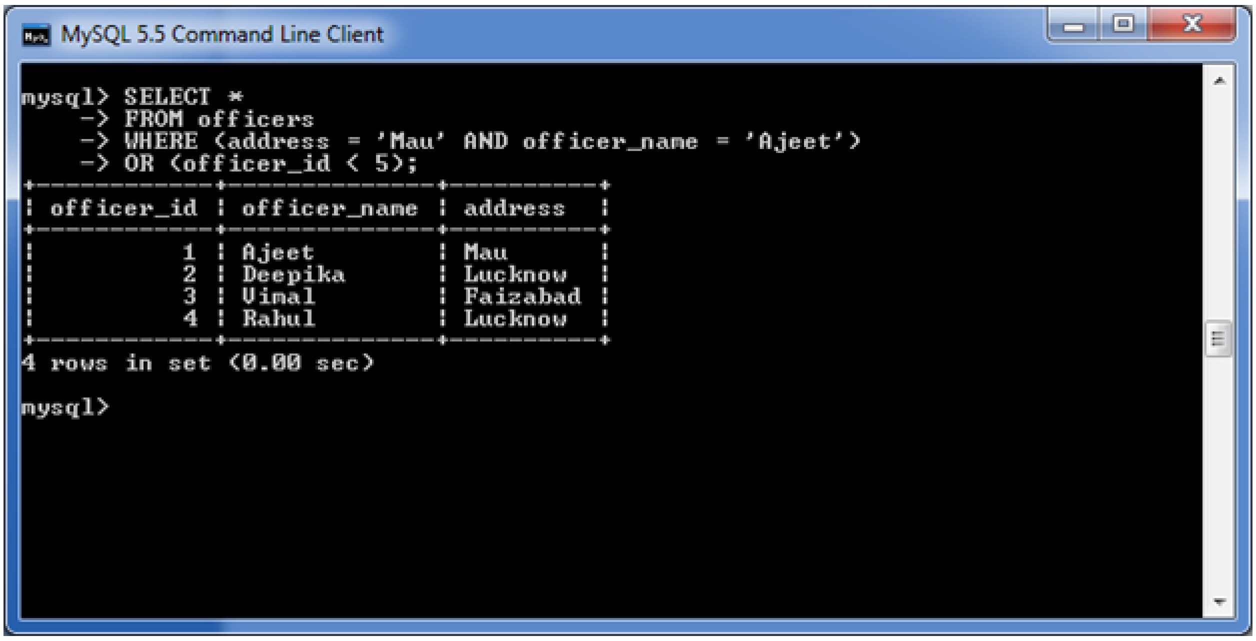
You can also use the AND & OR conditions altogether with the WHERE clause.

See this example:

Execute the following query:

```
SELECT *
FROM officers
WHERE (address = 'Mau' AND officer_name = 'Ajeet')
OR (officer_id < 5);
```

Output:



```
mysql> SELECT *
    -> FROM officers
    -> WHERE (address = 'Mau' AND officer_name = 'Ajeet')
    -> OR (officer_id < 5);
+------------+--------------+----------+
| officer_id | officer_name | address  |
+------------+--------------+----------+
|          1 | Ajeet        | Mau      |
|          2 | Deepika      | Lucknow  |
|          3 | Vimal        | Faizabad |
|          4 | Rahul        | Lucknow  |
+------------+--------------+----------+
4 rows in set (0.00 sec)

mysql>
```

MySQL Distinct Clause

MySQL DISTINCT clause is used to remove duplicate records from the table and fetch only the unique records. The DISTINCT clause is only used with the SELECT statement.

Syntax:

```
1.     SELECT DISTINCT expressions
2.     FROM tables
3.     [WHERE conditions];
```

Parameters

expressions: specify the columns or calculations that you want to retrieve.

tables: specify the name of the tables from where you 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 met for the records to be selected.

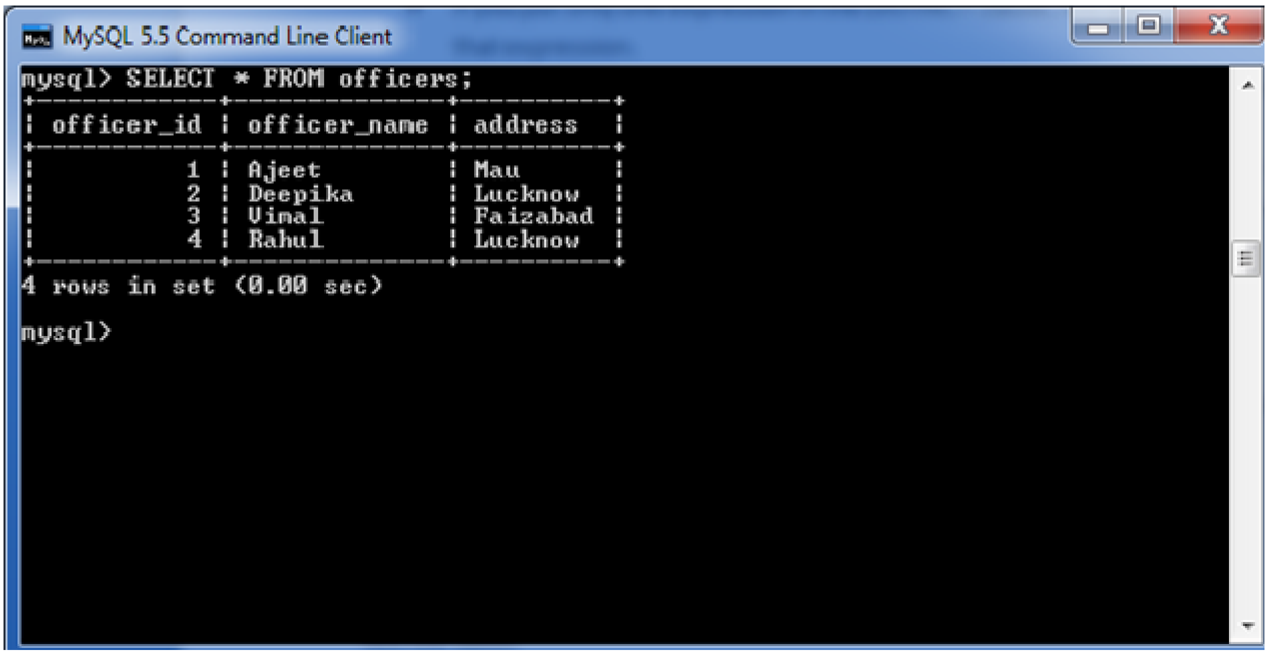
Note:

- If you put only one expression in the DISTINCT clause, the query will return the unique values for that expression.
- If you put more than one expression in the DISTINCT clause, the query will retrieve unique combinations for the expressions listed.
- In MySQL, the DISTINCT clause doesn't ignore NULL values. So if you are using the DISTINCT clause in your SQL statement, your result set will include NULL as a distinct value.

MySQL DISTINCT Clause with single expression

If you use a single expression then the MySQL DISTINCT clause will return a single field with unique records (no duplicate record).

See the table:

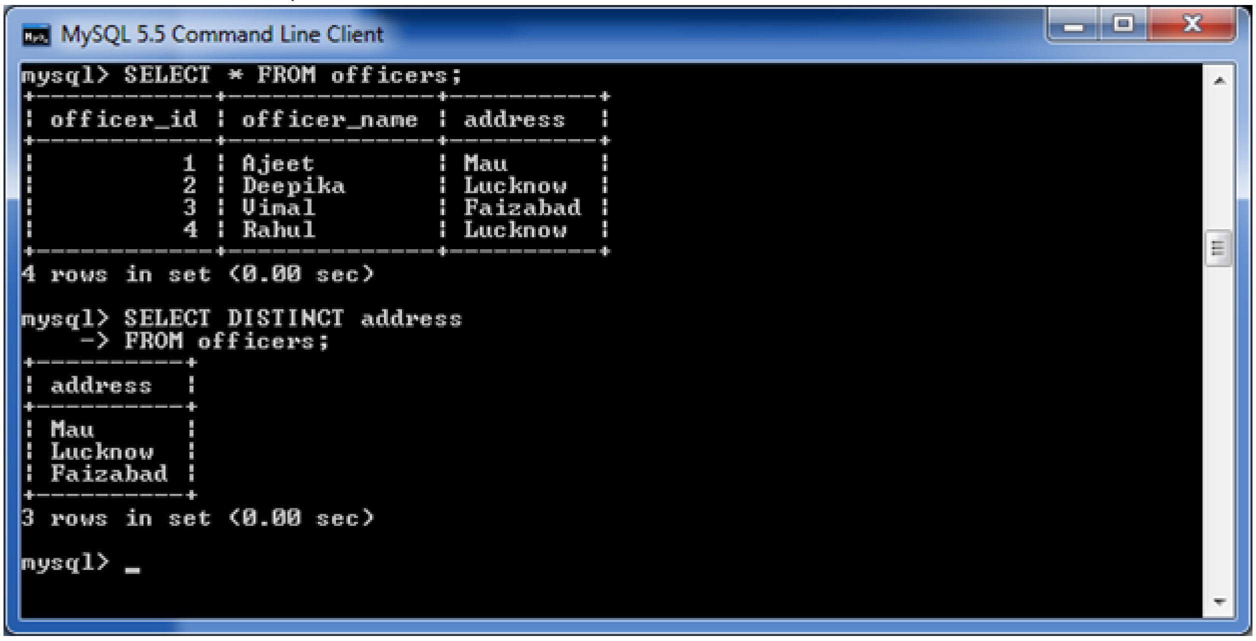


MySQL 5.5 Command Line Client

```
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 (0.00 sec)

mysql>
```

Use the following query:

1. **SELECT DISTINCT** address
2. **FROM** officers;



MySQL 5.5 Command Line Client

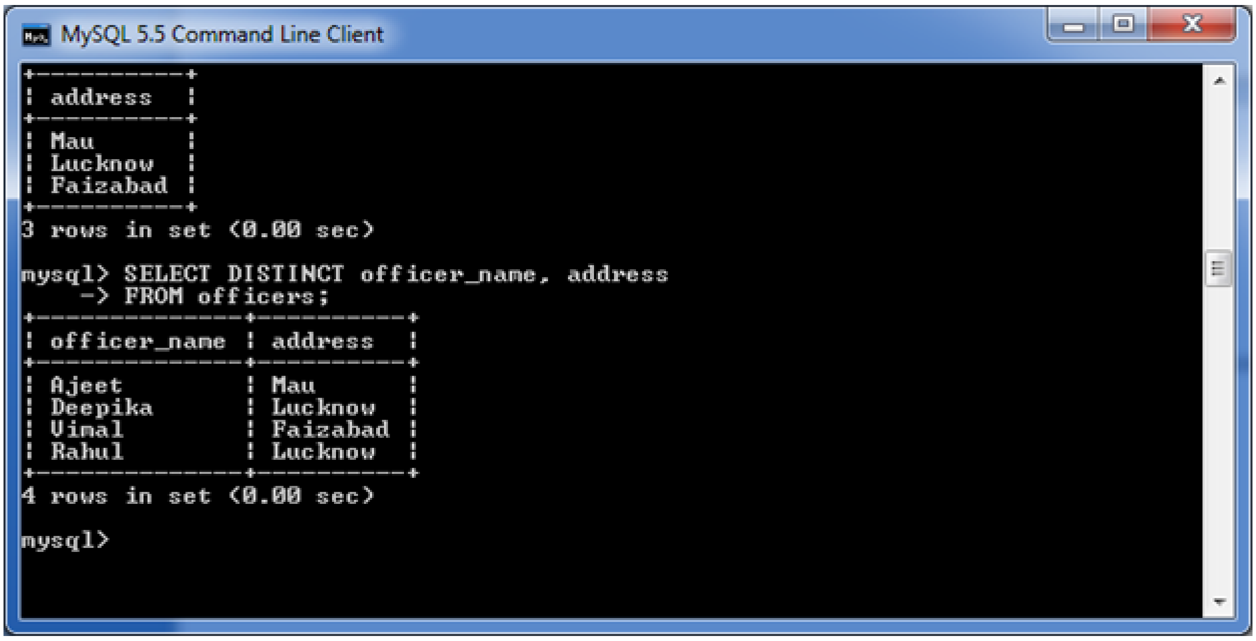
```
mysql> SELECT * FROM officers;
+------------+--------------+----------+
| officer_id | officer_name | address  |
+------------+--------------+----------+
|          1 | Ajeet        | Mau      |
|          2 | Deepika      | Lucknow  |
|          3 | Vimal        | Faizabad |
|          4 | Rahul        | Lucknow  |
+------------+--------------+----------+
4 rows in set (0.00 sec)

mysql> SELECT DISTINCT address
    -> FROM officers;
+----------+
| address  |
+----------+
| Mau      |
| Lucknow  |
| Faizabad |
+----------+
3 rows in set (0.00 sec)

mysql> _
```

MySQL DISTINCT Clause with multiple expressions

If you use multiple expressions with DISTINCT Clause then MySQL DISTINCT clause will remove duplicates from more than one field in your SELECT statement.

Use the following query:

```
SELECT DISTINCT officer_name, address
FROM officers;
```



MySQL 5.5 Command Line Client

```
+----------+
| address  |
+----------+
| Mau      |
| Lucknow  |
| Faizabad |
+----------+
3 rows in set (0.00 sec)

mysql> SELECT DISTINCT officer_name, address
    -> FROM officers;
+--------------+----------+
| officer_name | address  |
+--------------+----------+
| Ajeet        | Mau      |
| Deepika      | Lucknow  |
| Vimal        | Faizabad |
| Rahul        | Lucknow  |
+--------------+----------+
4 rows in set (0.00 sec)

mysql>
```

MySQL FROM Clause

The MySQL FROM Clause is used to select some records from a table. It can also be used to retrieve records from multiple tables using JOIN condition.

Syntax:

```
FROM table1
[ { INNER JOIN | LEFT [OUTER] JOIN| RIGHT [OUTER] JOIN } table2
ON table1.column1 = table2.column1 ]
```

Parameters

table1 and table2: specify tables used in the MySQL statement. The two tables are joined based on table1.column1 = table2.column1.

Note:

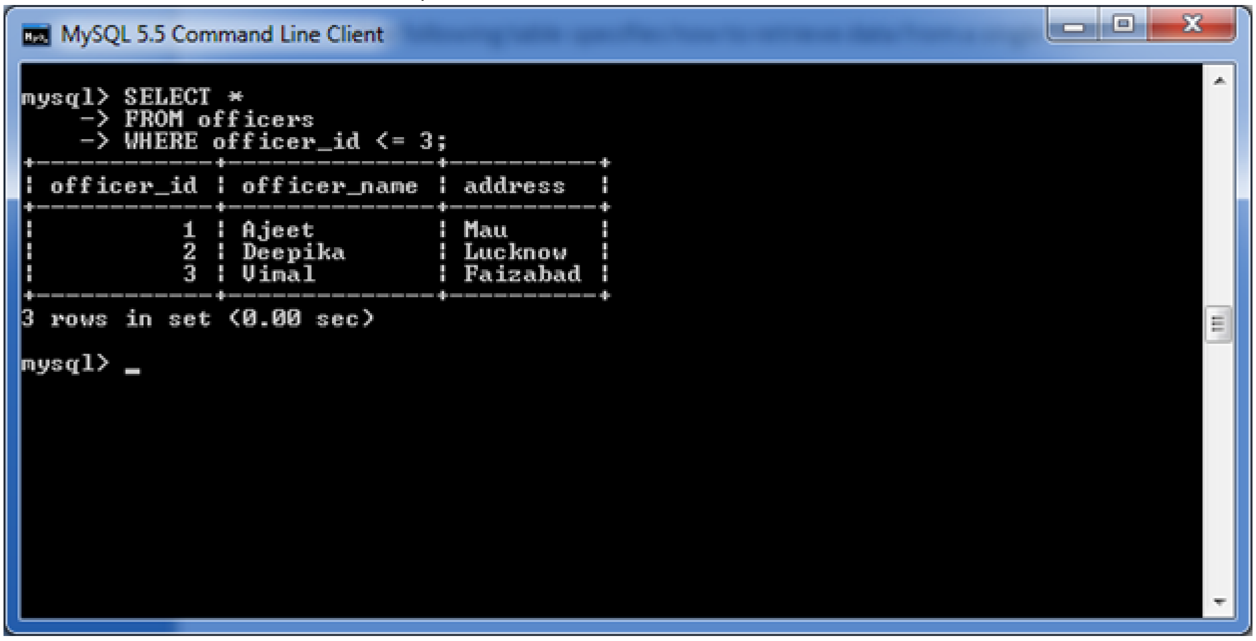
- If you are using the FROM clause in a MySQL statement then at least one table must have been selected.
- If you are using two or more tables in the MySQL FROM clause, these tables are generally joined using INNER or OUTER joins.

MySQL FROM Clause: Retrieve data from one table

The following query specifies how to retrieve data from a single table.

Use the following Query:

```
SELECT *
FROM officers
WHERE officer_id <= 3;
```

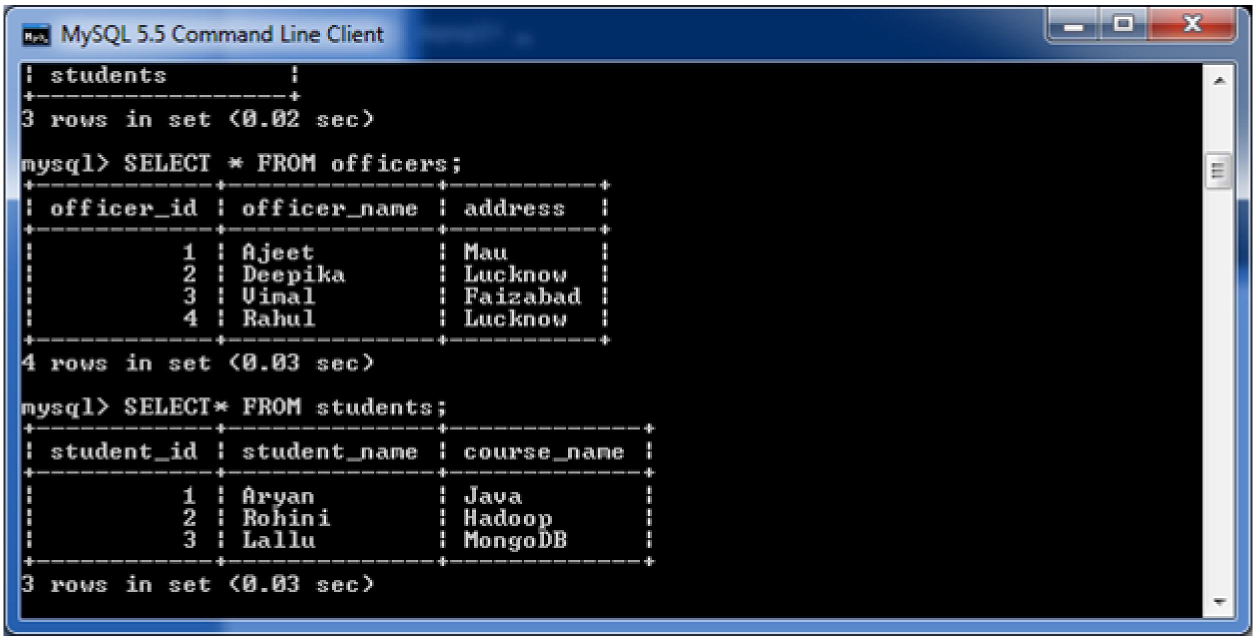


```
mysql> SELECT *
    -> FROM officers
    -> WHERE officer_id <= 3;
+------------+--------------+----------+
| officer_id | officer_name | address  |
+------------+--------------+----------+
|          1 | Ajeet        | Mau      |
|          2 | Deepika      | Lucknow  |
|          3 | Vimal        | Faizabad |
+------------+--------------+----------+
3 rows in set (0.00 sec)

mysql> _
```

MySQL FROM Clause: Retrieve data from two tables with inner join

Let's take an example to retrieve data from two tables using INNER JOIN.

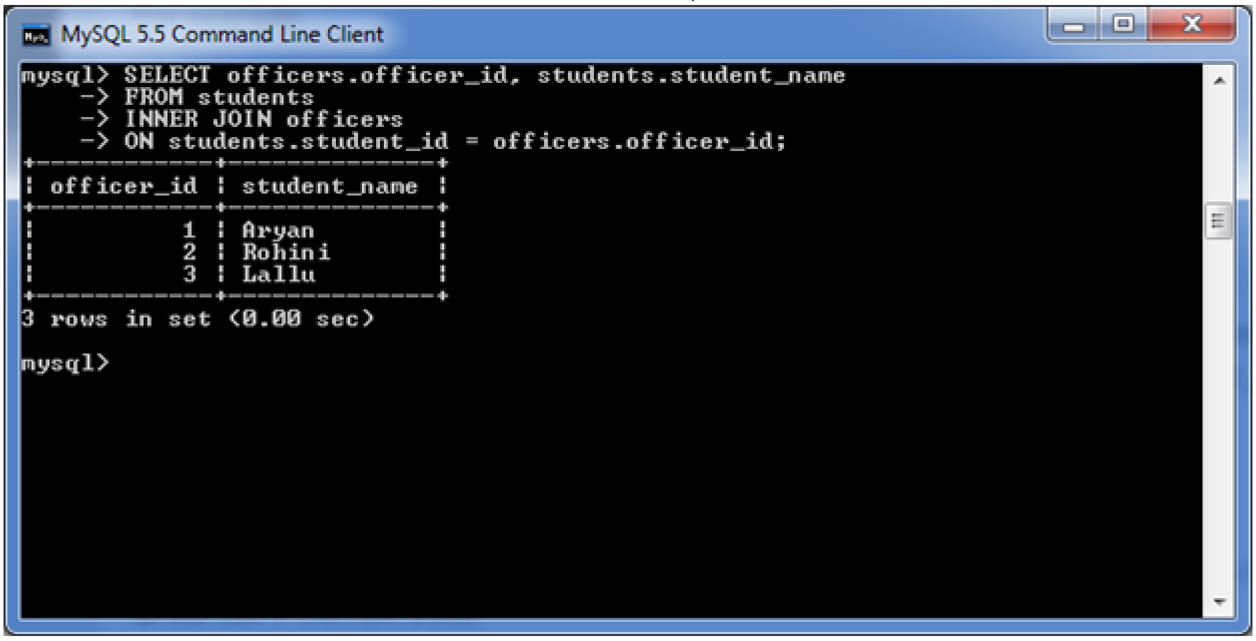
Here, we have two tables "officers" and "students".



```
| students         |
+------------------+
3 rows in set (0.02 sec)

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 (0.03 sec)

mysql> SELECT* FROM students;
+------------+--------------+-------------+
| student_id | student_name | course_name |
+------------+--------------+-------------+
|          1 | Aryan        | Java        |
|          2 | Rohini       | Hadoop      |
|          3 | Lallu        | MongoDB     |
+------------+--------------+-------------+
3 rows in set (0.03 sec)
```

Execute the following query:

```
1.  SELECT officers.officer_id, students.student_name
2.  FROM students
3.  INNER JOIN officers
4.  ON students.student_id = officers.officer_id;
```

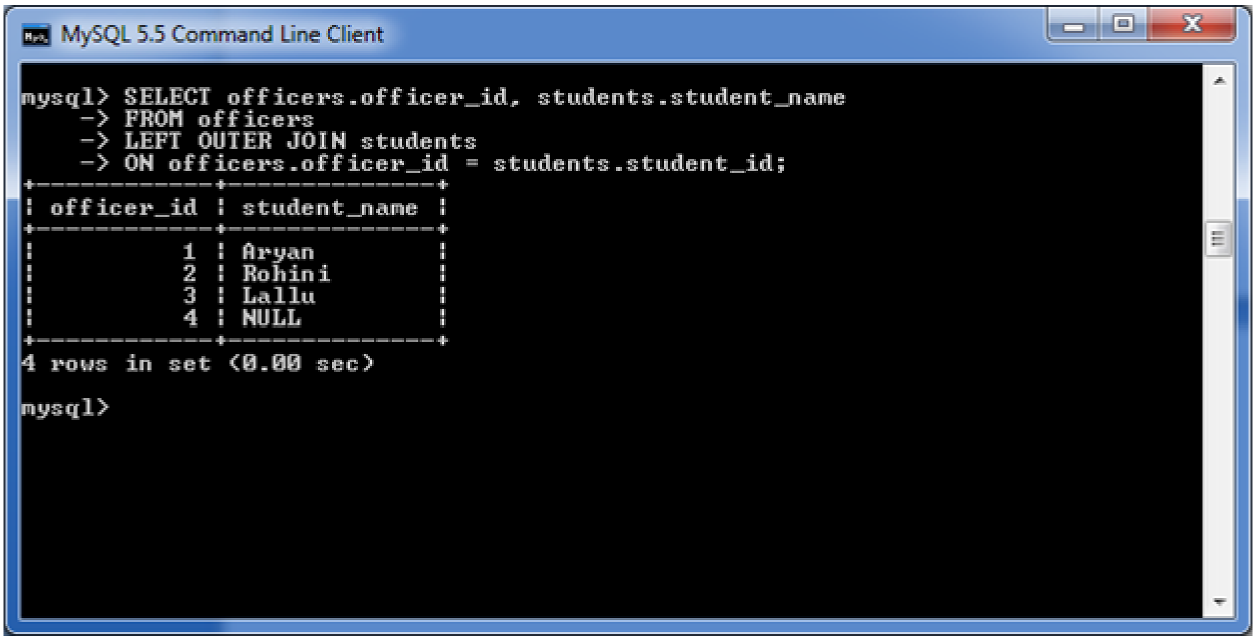


```
mysql> SELECT officers.officer_id, students.student_name
    -> FROM students
    -> INNER JOIN officers
    -> ON students.student_id = officers.officer_id;
+------------+--------------+
| officer_id | student_name |
+------------+--------------+
|          1 | Aryan        |
|          2 | Rohini       |
|          3 | Lallu        |
+------------+--------------+
3 rows in set (0.00 sec)

mysql>
```

MySQL FROM Clause: Retrieve data from two tables using outer join

Execute the following query:

```sql
SELECT officers.officer_id, students.student_name
FROM officers
LEFT OUTER JOIN students
ON officers.officer_id = students.student_id;
```



```
mysql> SELECT officers.officer_id, students.student_name
    -> FROM officers
    -> LEFT OUTER JOIN students
    -> ON officers.officer_id = students.student_id;
+------------+--------------+
| officer_id | student_name |
+------------+--------------+
|          1 | Aryan        |
|          2 | Rohini       |
|          3 | Lallu        |
|          4 | NULL         |
+------------+--------------+
4 rows in set (0.00 sec)

mysql>
```

MySQL ORDER BY Clause

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

Syntax:

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

Parameters

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 provider).

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

Note: You can use MySQL ORDER BY clause in a SELECT statement, SELECT LIMIT statement, and DELETE LIMIT statement.

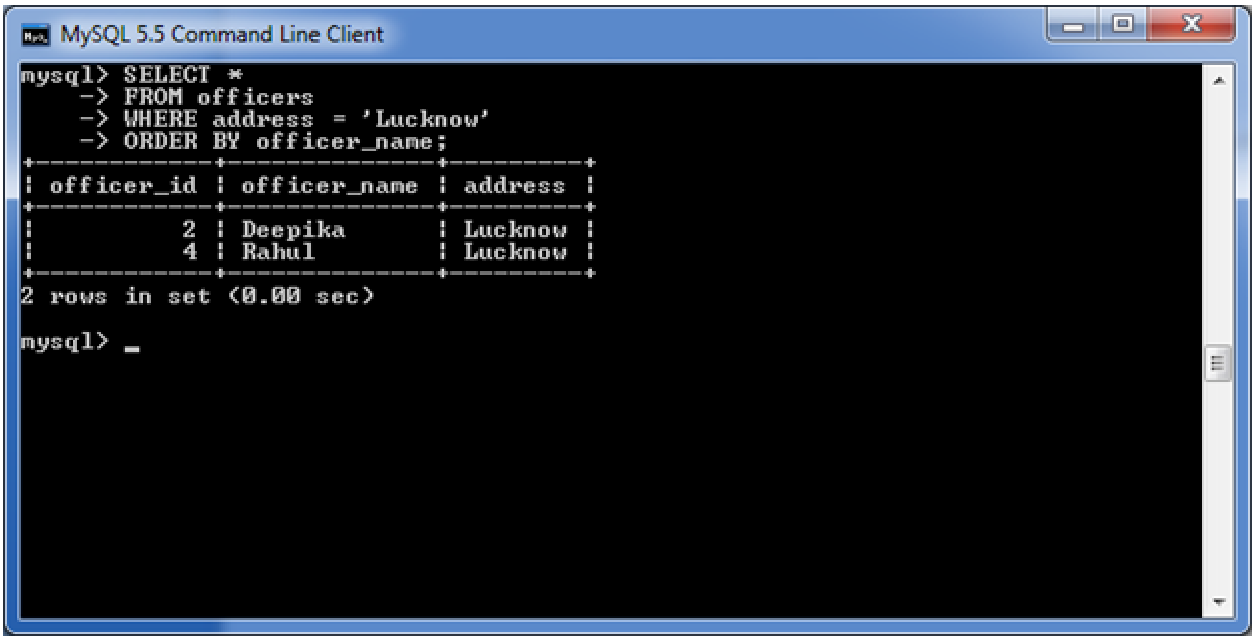
MySQL ORDER BY: without using ASC/DESC attribute

If you use MySQL 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> _
```

MySQL ORDER BY: with ASC attribute

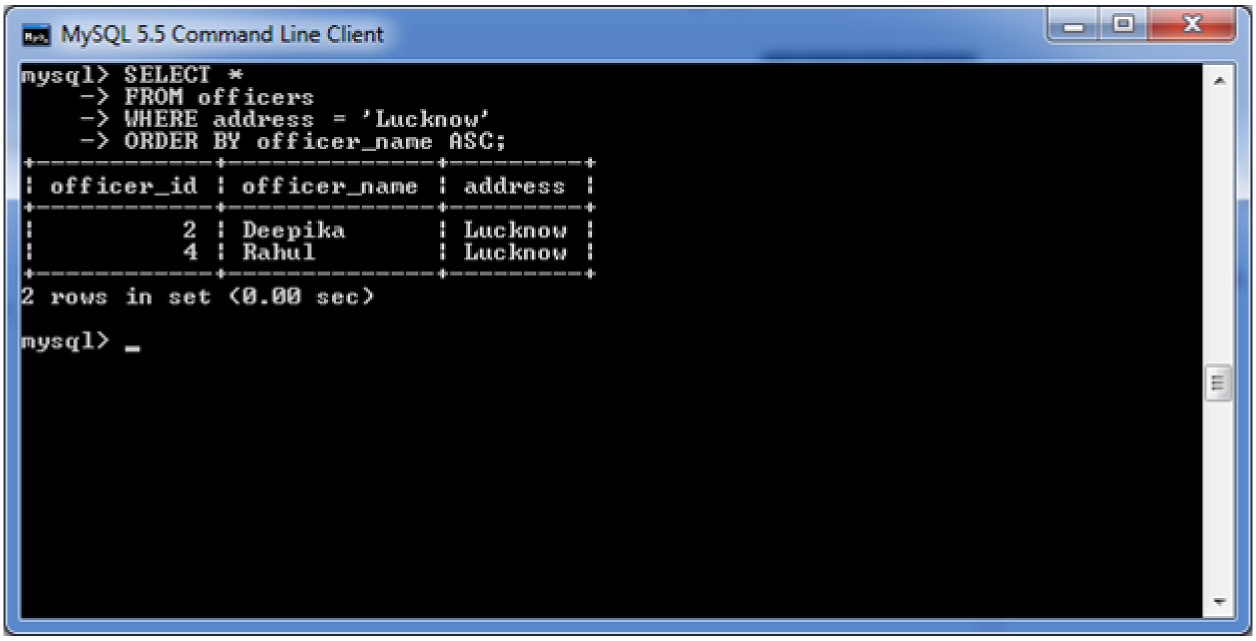
Let's take an example to retrieve the data in ascending order.

Execute the following query:

```
SELECT *
FROM officers
WHERE address = 'Lucknow'
```

4. **ORDER BY** officer_name **ASC**;

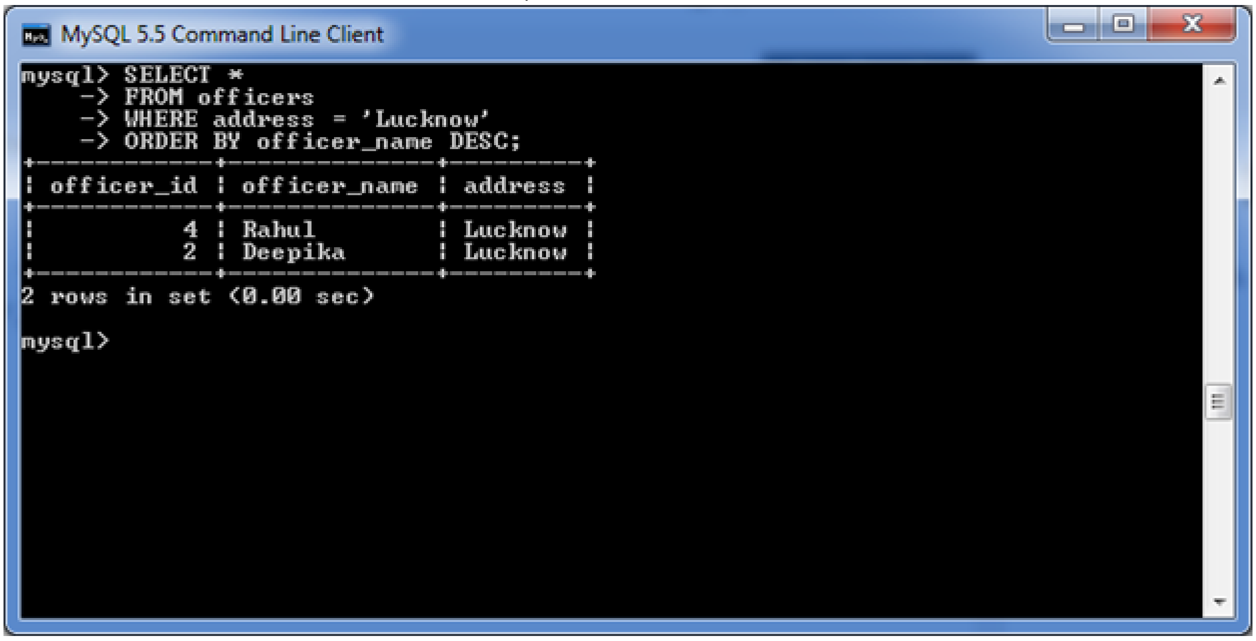
Output:



```
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> _
```

MySQL ORDER BY: with DESC attribute

1. **SELECT** *
2. **FROM** officers
3. **WHERE** address = 'Lucknow'
4. **ORDER BY** officer_name **DESC**;



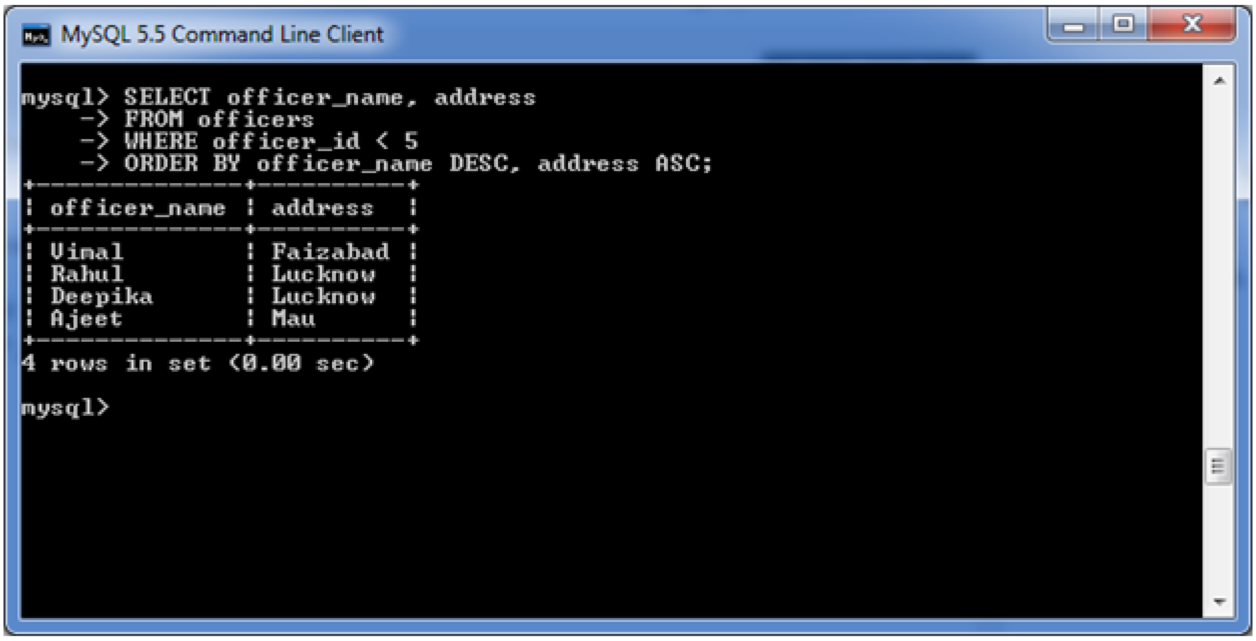
```
mysql> SELECT *
    -> FROM officers
    -> WHERE address = 'Lucknow'
    -> ORDER BY officer_name DESC;
+------------+--------------+---------+
| officer_id | officer_name | address |
+------------+--------------+---------+
|          4 | Rahul        | Lucknow |
|          2 | Deepika      | Lucknow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

mysql>
```

MySQL 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  |
+--------------+----------+
| Vimal        | Faizabad |
| Rahul        | Lucknow  |
| Deepika      | Lucknow  |
| Ajeet        | Mau      |
+--------------+----------+
4 rows in set (0.00 sec)

mysql>
```

MySQL GROUP BY Clause

The MYSQL 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;
```

Parameters

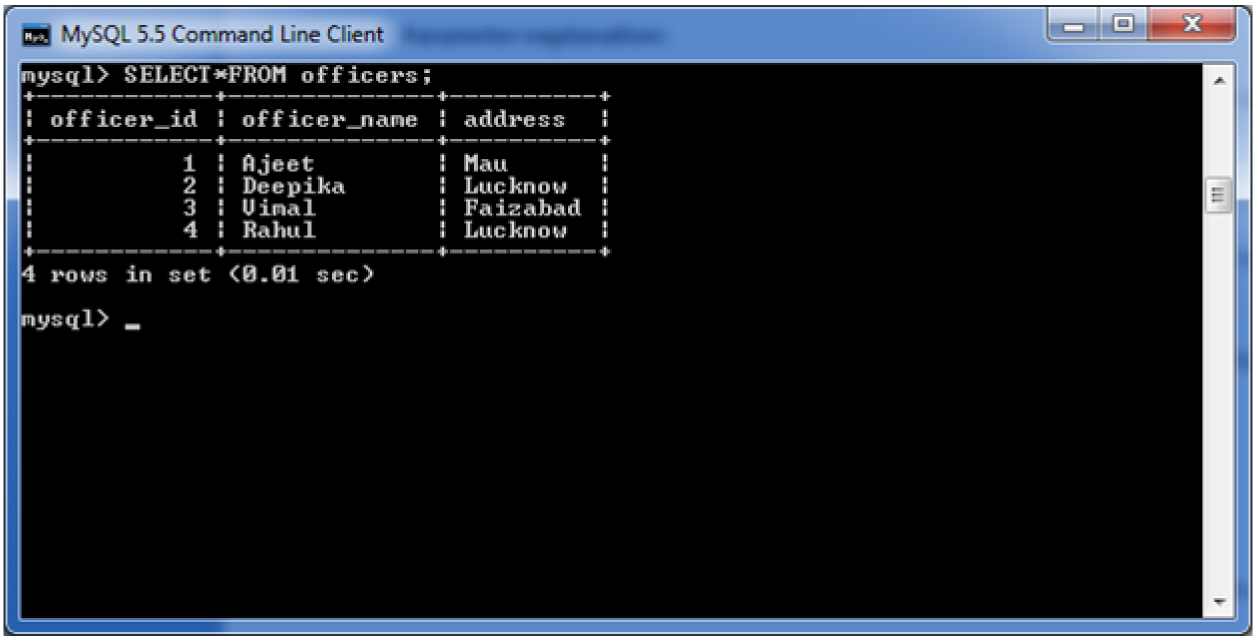
expression1, expression2, ... expression_n: It specifies the expressions that are not encapsulated within an aggregate function and must be included in the GROUP BY clause.

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) MySQL GROUP BY Clause with COUNT function

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



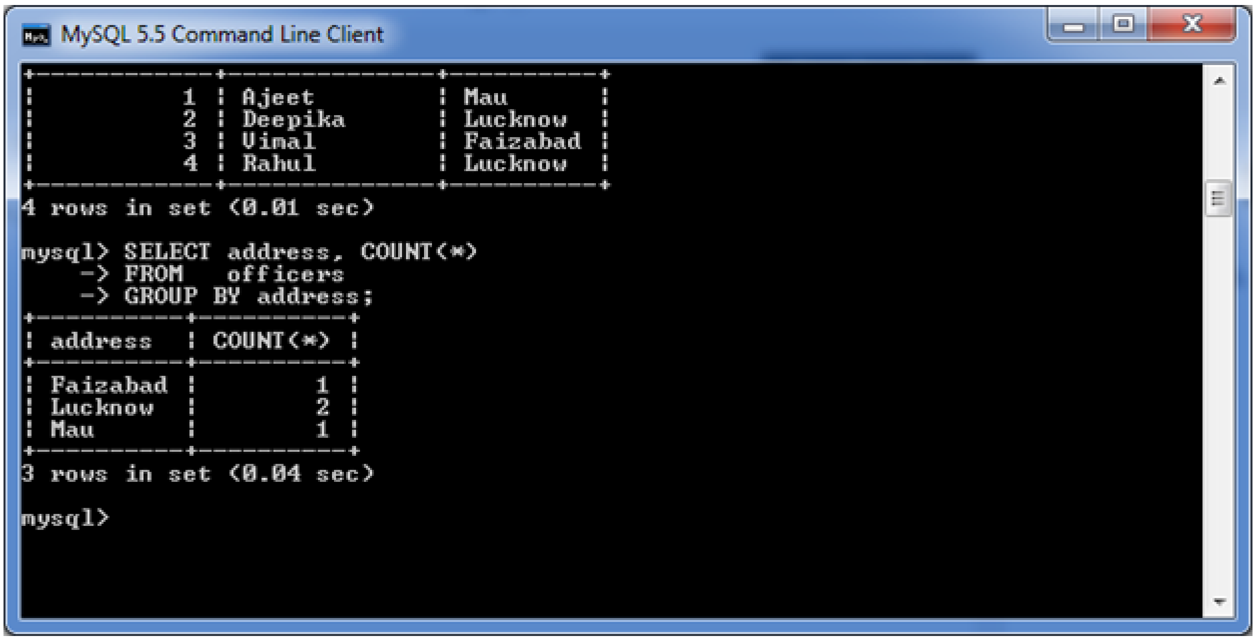
```
mysql> SELECT*FROM officers;
+------------+--------------+----------+
| officer_id | officer_name | address  |
+------------+--------------+----------+
|          1 | Ajeet        | Mau      |
|          2 | Deepika      | Lucknow  |
|          3 | Vimal        | Faizabad |
|          4 | Rahul 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

mysql> _
```

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:

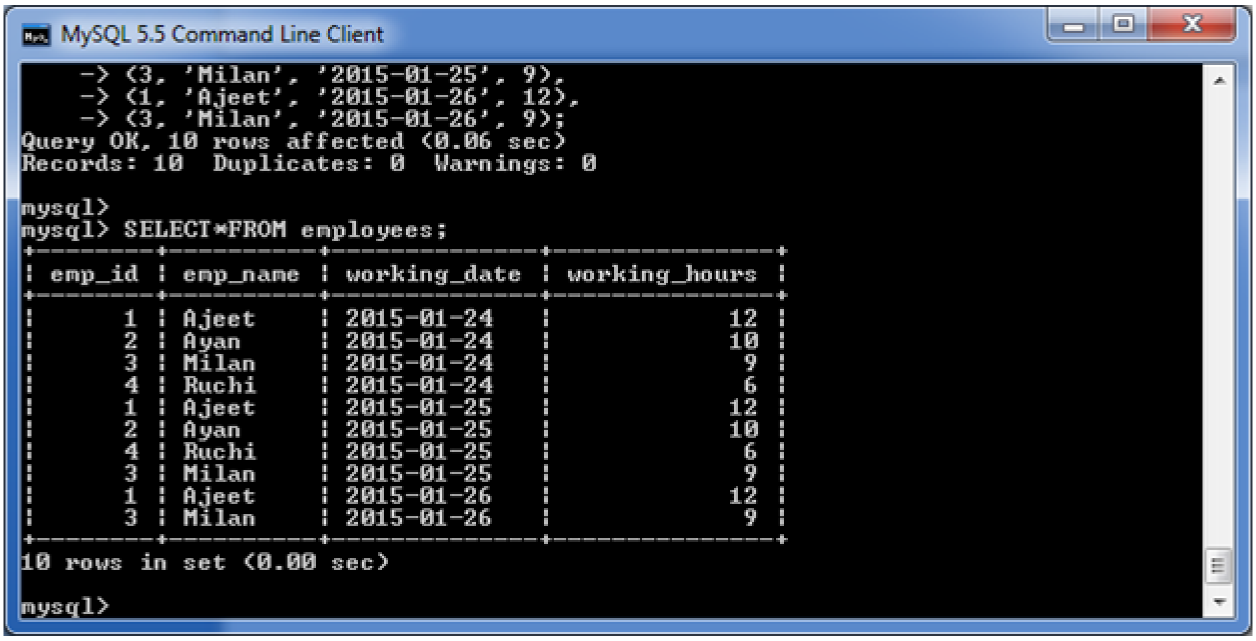


MySQL 5.5 Command Line Client

```
+-------------+--------------+-----------+
|           1 | Ajeet        | Mau       |
|           2 | Deepika      | Lucknow   |
|           3 | Vimal        | Faizabad  |
|           4 | Rahul        | Lucknow   |
+-------------+--------------+-----------+
4 rows in set (0.01 sec)

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) MySQL GROUP BY Clause with SUM function

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



MySQL 5.5 Command Line Client

```
    -> (3, 'Milan', '2015-01-25', 9),
    -> (1, 'Ajeet', '2015-01-26', 12),
    -> (3, 'Milan', '2015-01-26', 9);
Query OK, 10 rows affected (0.06 sec)
Records: 10  Duplicates: 0  Warnings: 0

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)

mysql>
```

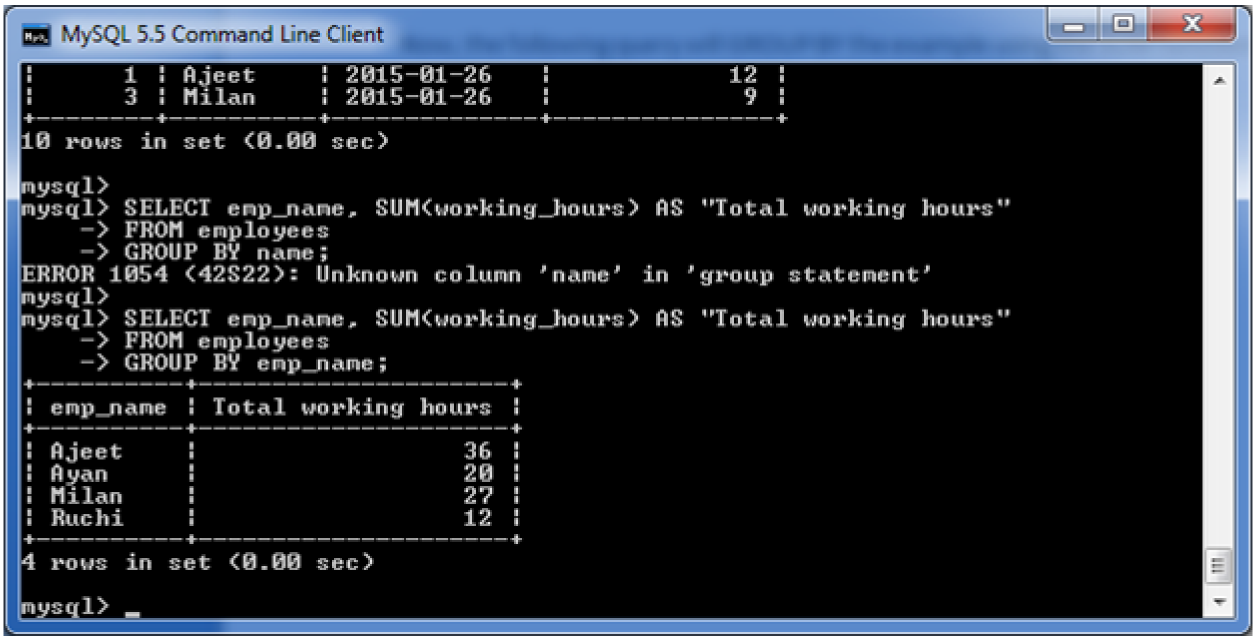
Now, the following query will GROUP BY the example using the SUM function and return the emp_name and total working hours of each employee.

Execute the following query:

```
1.  SELECT emp_name, SUM(working_hours) AS "Total working hours"
2.  FROM employees
```

3. **GROUP BY** emp_name;

Output:



```
|       1 | Ajeet     | 2015-01-26    |               12 |
|       3 | Milan     | 2015-01-26    |                9 |
+---------+-----------+---------------+------------------+
10 rows in set (0.00 sec)

mysql>
mysql> SELECT emp_name, SUM(working_hours) AS "Total working hours"
    -> FROM employees
    -> GROUP BY name;
ERROR 1054 (42S22): Unknown column 'name' in 'group statement'
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)

mysql> _
```

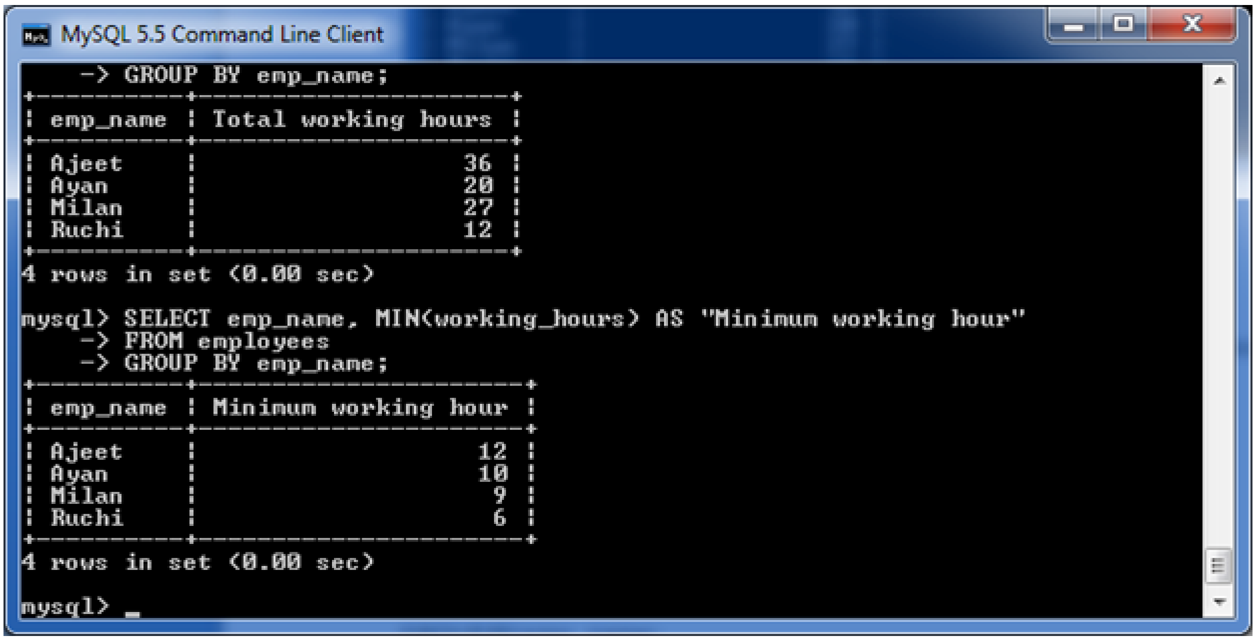
(iii) MySQL GROUP BY Clause with MIN function

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

Execute the following query:

1. **SELECT** emp_name, **MIN**(working_hours) **AS** "Minimum working hour"
2. **FROM** employees
3. **GROUP BY** emp_name;

Output:



```
    -> GROUP BY emp_name;
+----------+---------------------+
| emp_name | Total working hours |
+----------+---------------------+
| Ajeet    |                  36 |
| Ayan     |                  20 |
| Milan    |                  27 |
| Ruchi    |                  12 |
+----------+---------------------+
4 rows in set (0.00 sec)

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)

mysql> _
```

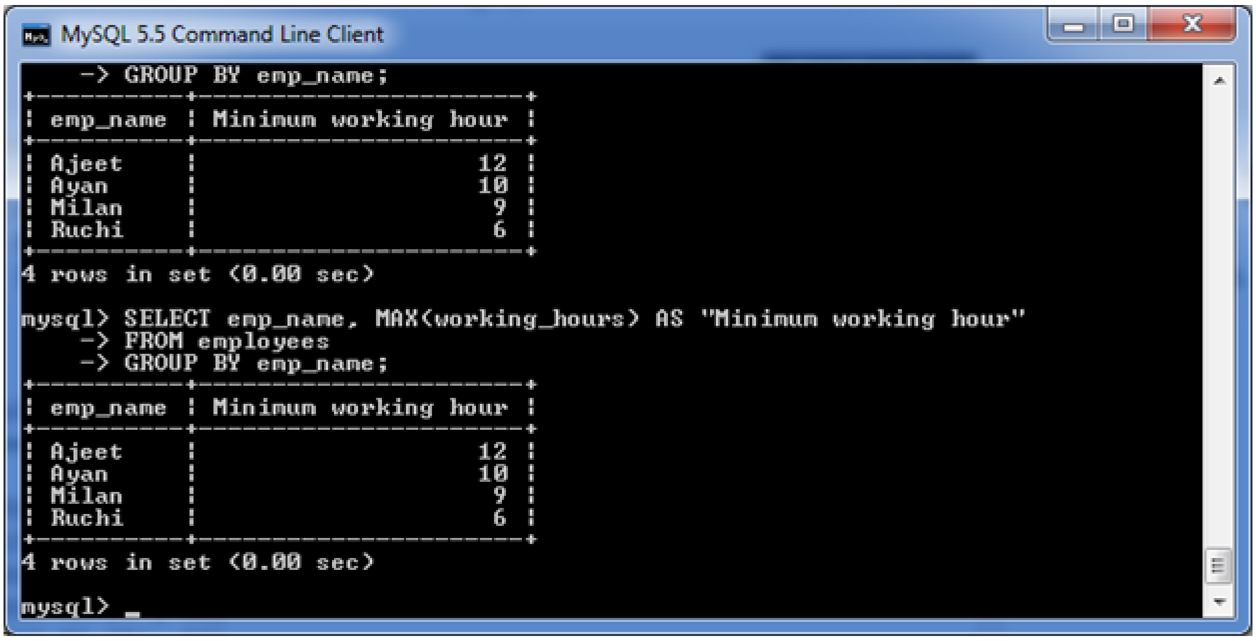
(iv) MySQL GROUP BY Clause with MAX function

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

Execute the following query:

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

Output:



```
    -> GROUP BY emp_name;
+----------+-----------------------+
| emp_name | Minimum working hour |
+----------+-----------------------+
| Ajeet    |                    12 |
| Ayan     |                    10 |
| Milan    |                     9 |
| Ruchi    |                     6 |
+----------+-----------------------+
4 rows in set (0.00 sec)

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)

mysql> _
```

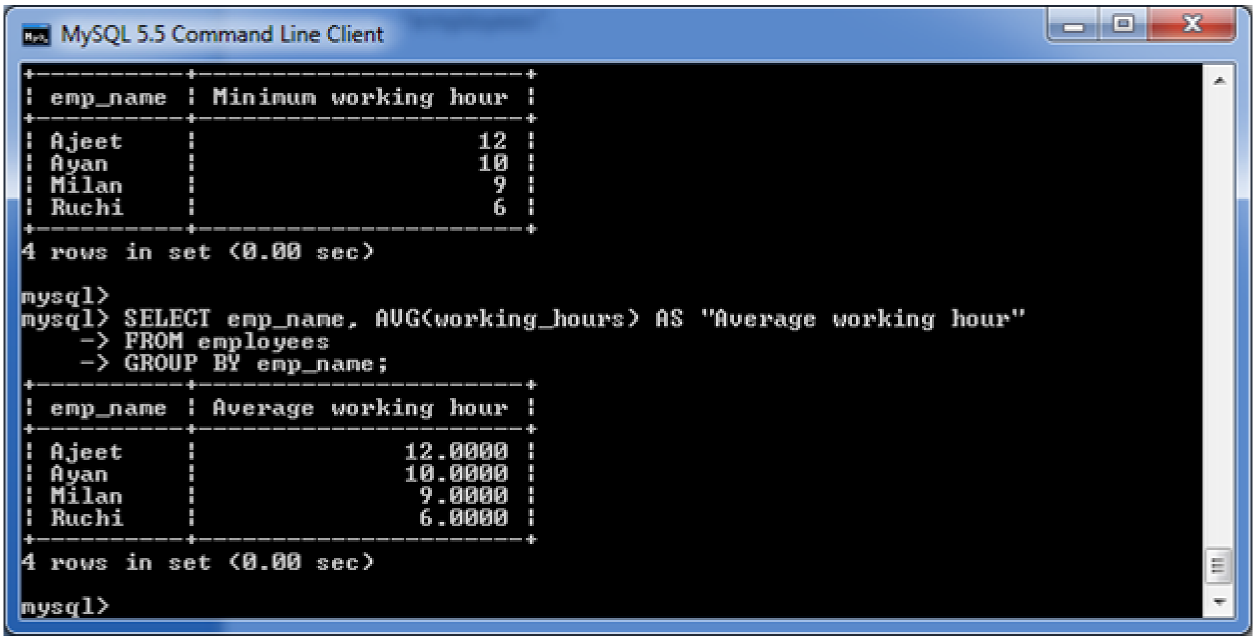
(v) MySQL GROUP BY Clause with AVG function

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

Execute the following query:

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

Output:



```
+----------+----------------------+
| emp_name | Minimum working hour |
+----------+----------------------+
| Ajeet    |                   12 |
| Ayan     |                   10 |
| Milan    |                    9 |
| Ruchi    |                    6 |
+----------+----------------------+
4 rows in set (0.00 sec)

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)

mysql>
```

MySQL HAVING Clause

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

Syntax:

```
1.   SELECT expression1, expression2, ... expression_n,
2.   aggregate_function (expression)
3.   FROM tables
4.   [WHERE conditions]
5.   GROUP BY expression1, expression2, ... expression_n
6.   HAVING condition;
```

Parameters

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

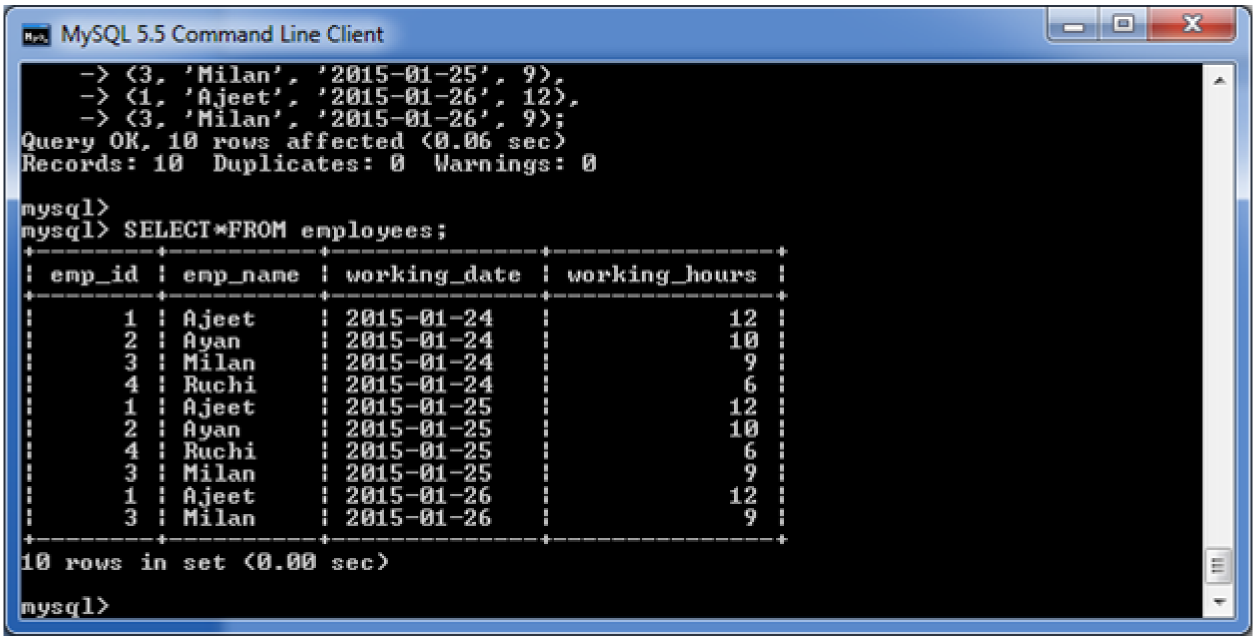
expression1, expression2, ... expression_n: It specifies the expressions that are not encapsulated within an aggregate function and must be included in the GROUP BY clause.

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.

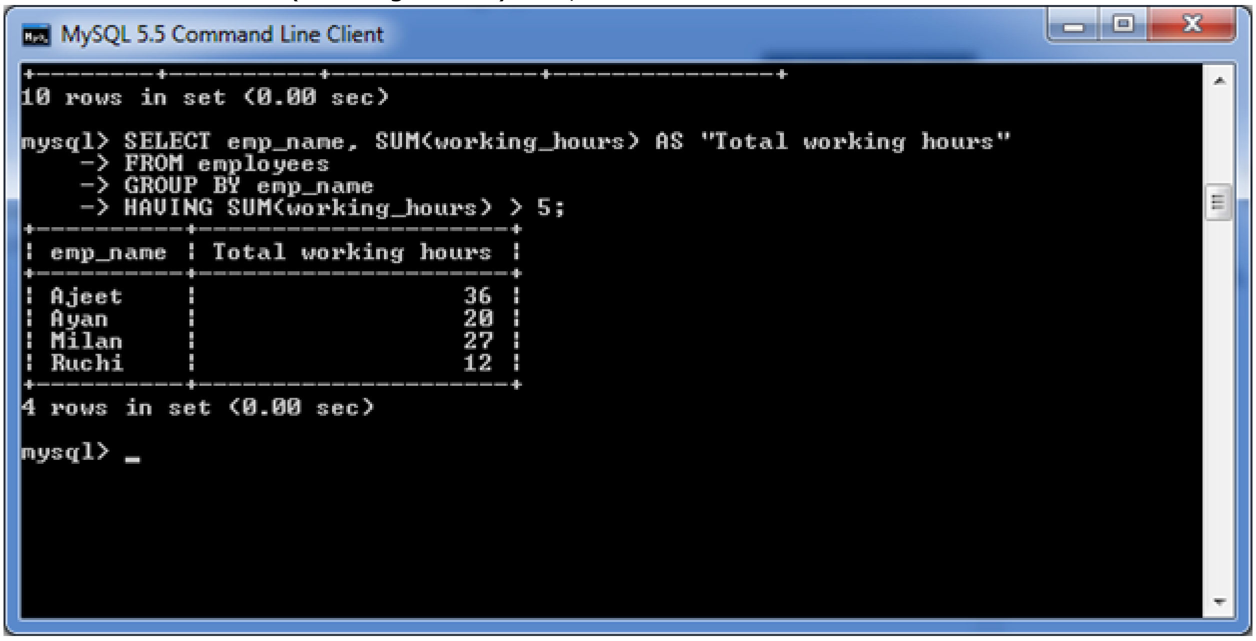


```
MySQL 5.5 Command Line Client
    -> (3, 'Milan', '2015-01-25', 9),
    -> (1, 'Ajeet', '2015-01-26', 12),
    -> (3, 'Milan', '2015-01-26', 9);
Query OK, 10 rows affected (0.06 sec)
Records: 10  Duplicates: 0  Warnings: 0

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)

mysql>
```

Here, we use the SUM function with the HAVING Clause to return the emp_name and sum of their working hours.

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 5.5 Command Line Client
+--------+----------+--------------+---------------+
10 rows in set (0.00 sec)

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 |
+----------+---------------------+
4 rows in set (0.00 sec)

mysql> _
```

Simply, it can also be used with COUNT, MIN, MAX and AVG functions.