D B M S

MySQL (Part 1)

- **Create Database**
- **Show Database**
- **Drop Database**
- **Create Table**
- **Show Table**
- **Drop Table**
- **Alter Table**
- **Insert (Add Data to a Table)**
- **Functions (LAST_INSERT_ID(); NOW();)**
- **Select (Read Data from a Table: WHERE; AND; OR; IN; LIKE; ORDER BY; LIMIT…OFFSET…)**
- **Update (Update Data in a Table)**
- **Delete (Delete Data from a Table)**
- **Truncate Table**

MySQL (Part 1)

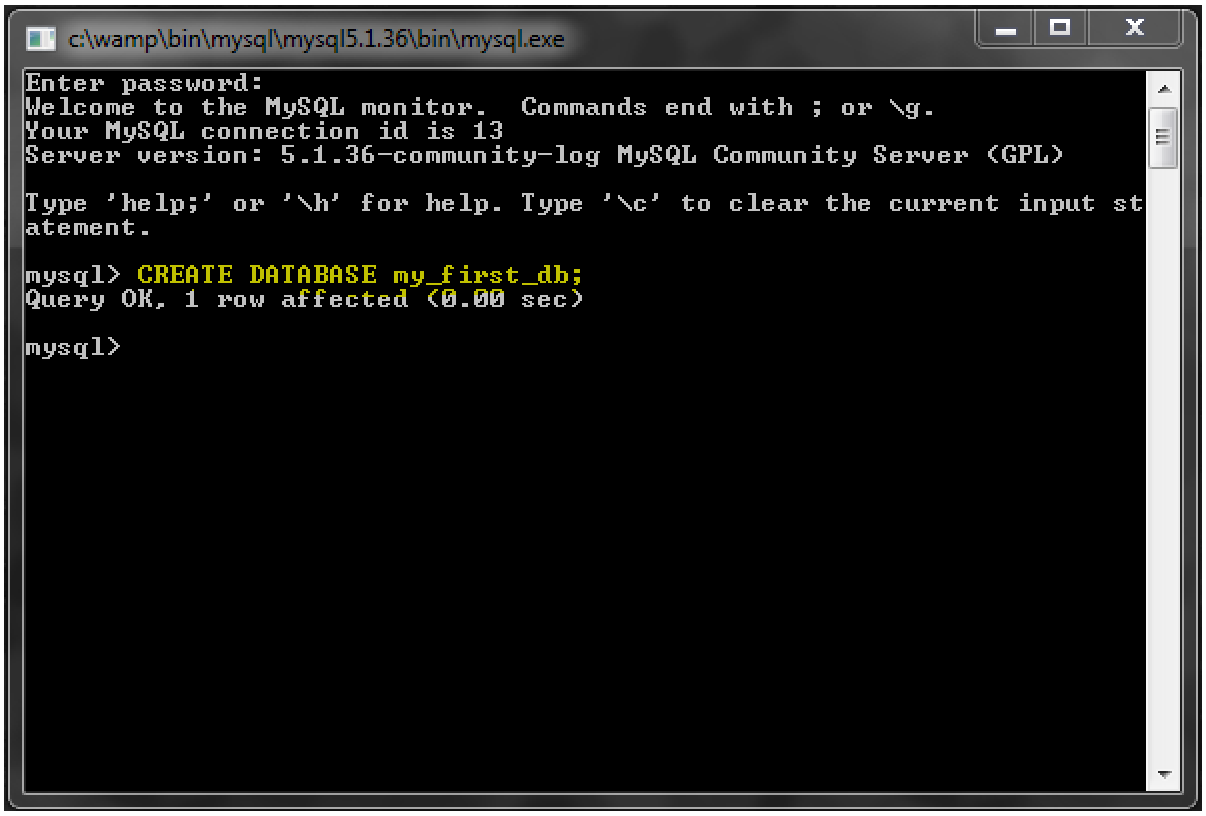
CREATE DATABASE: Creating a Database

Here comes our very first query. We are going to be creating a database to work with.

First, open up your MySQL Console and login. For WAMP, the default password is blank. For MAMP, the password should be 'root' by default.

After logging in, type this query and hit enter:

```
CREATE DATABASE my_first_db;
```



```
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 13
Server version: 5.1.36-community-log MySQL Community Server (GPL)

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

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

mysql>
```

Note that semicolon (;) is added at the end of the query, just like at the end of lines of code.

Also, the special words 'CREATE DATABASE' are case insensitive, along with all special words in SQL. But for the sake of readability, we will be writing them in uppercase.

Optional: Character Set and Collation

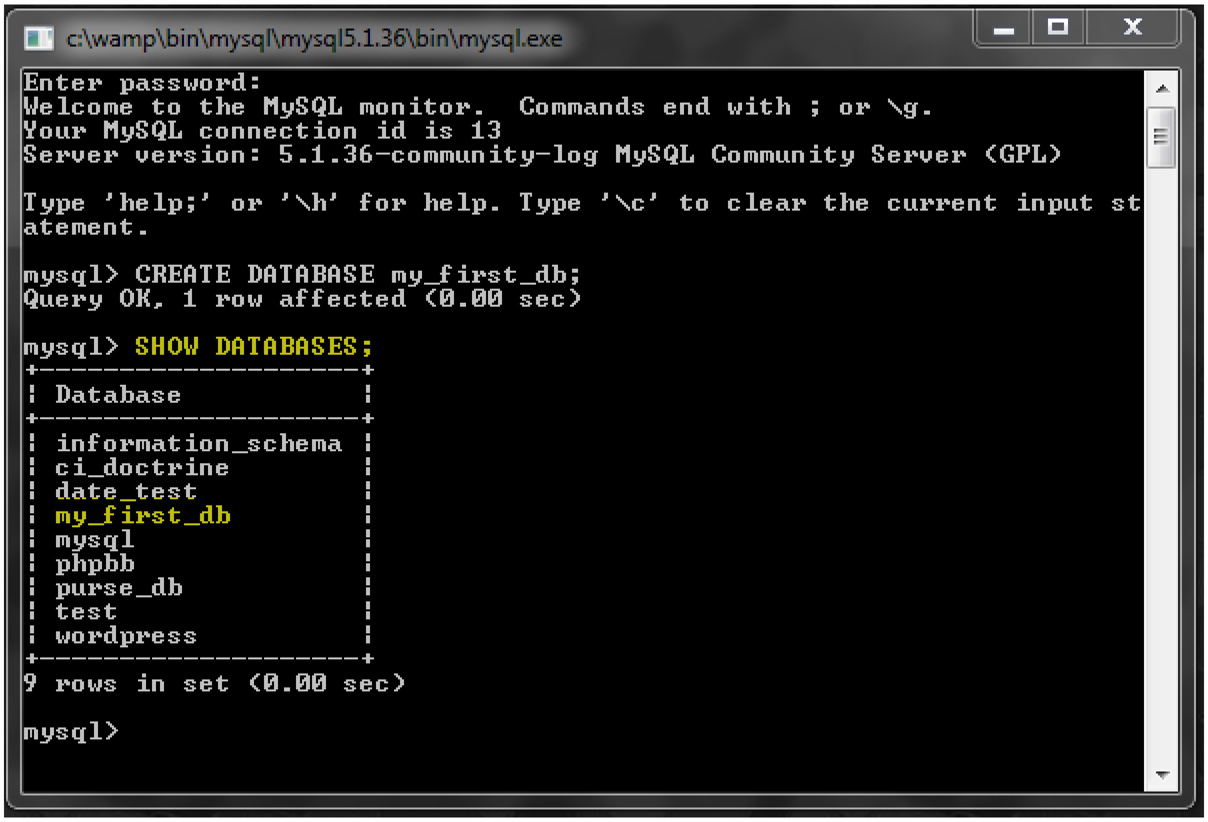
If you would like to set the default character set and collation, you can write the same query like this instead:

```
CREATE DATABASE my_first_db DEFAULT CHARACTER SET utf8 COLLATE
utf8_general_ci;
```

Here is a list of supported character sets and collations in MySQL.

SHOW DATABASES: List All Databases

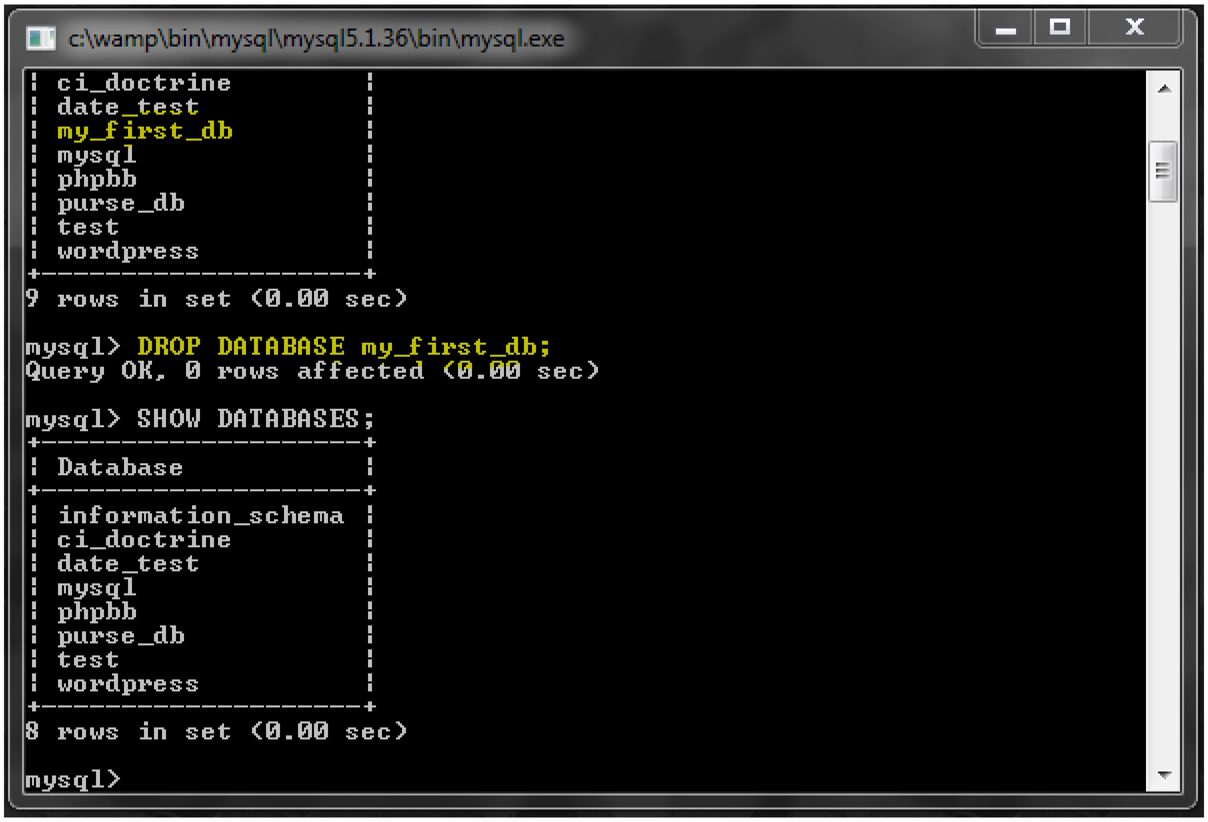
This query is used to get a list of all databases you have.



```
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 13
Server version: 5.1.36-community-log MySQL Community Server (GPL)

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

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

mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| ci_doctrine        |
| date_test          |
| my_first_db        |
| mysql              |
| phpbb              |
| purse_db           |
| test               |
| wordpress          |
+--------------------+
9 rows in set (0.00 sec)

mysql>
```

DROP DATABASE: Delete a Database

You can delete an existing database with this query.

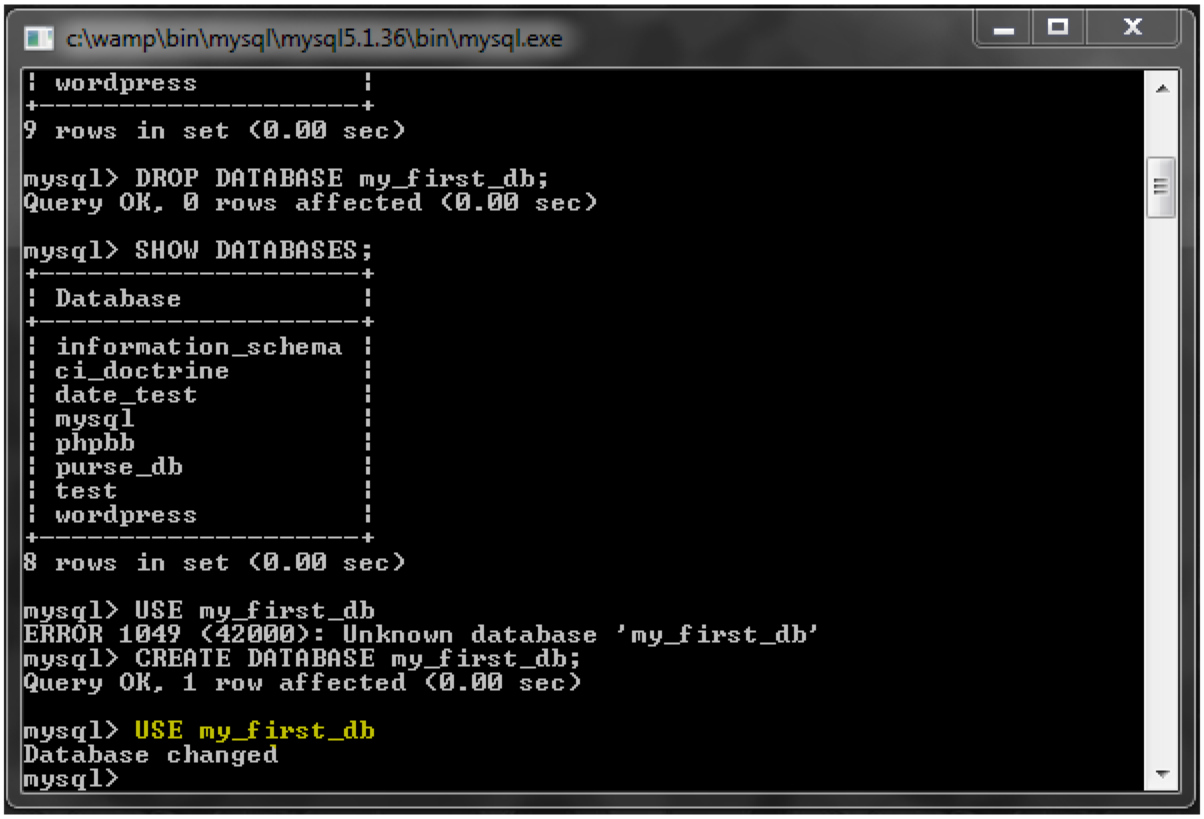


```
| ci_doctrine        |
| date_test          |
| my_first_db        |
| mysql              |
| phpbb              |
| purse_db           |
| test               |
| wordpress          |
+--------------------+
9 rows in set (0.00 sec)

mysql> DROP DATABASE my_first_db;
Query OK, 0 rows affected (0.00 sec)

mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| ci_doctrine        |
| date_test          |
| mysql              |
| phpbb              |
| purse_db           |
| test               |
| wordpress          |
+--------------------+
8 rows in set (0.00 sec)

mysql>
```

Be careful with this query, because it gives you no warnings. If you have tables and data under the database, they will all be deleted instantly.

USE: Selecting a Database

This technically is not a query. It is a 'statement' and does not require a semicolon at the end.



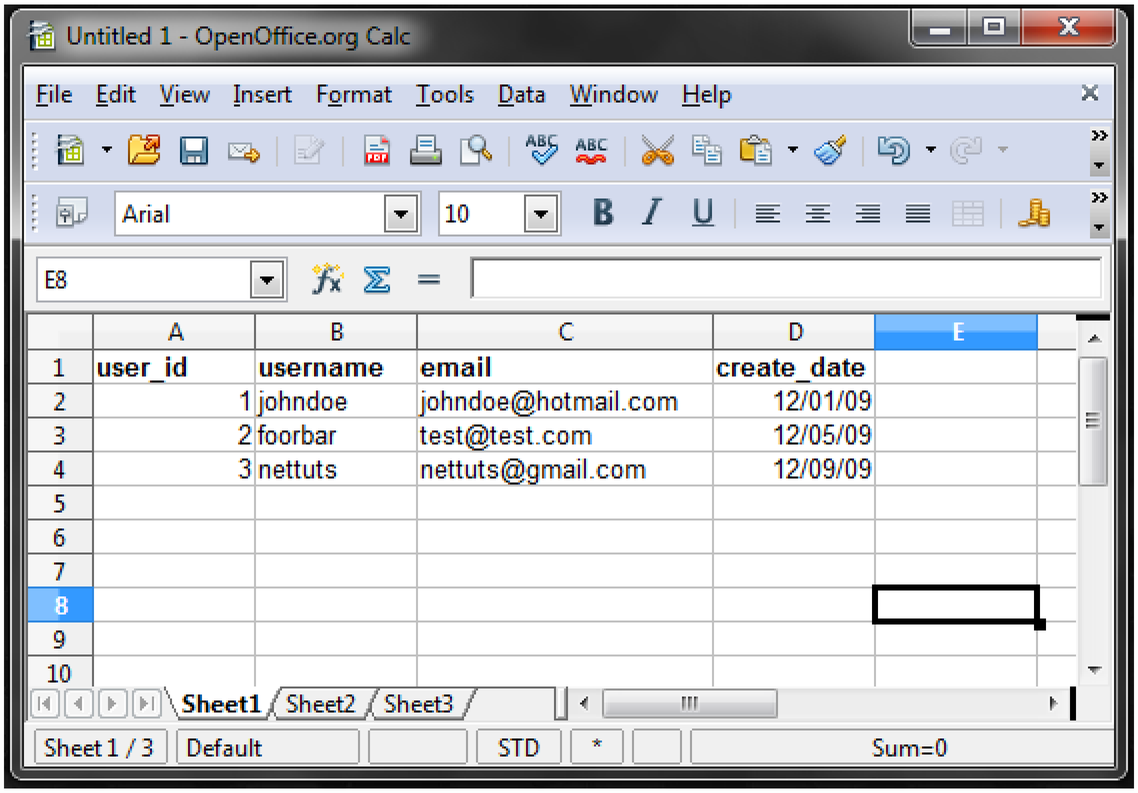
c:\wamp\bin\mysql\mysql5.1.36\bin\mysql.exe

```
| wordpress          |
+--------------------+
9 rows in set (0.00 sec)

mysql> DROP DATABASE my_first_db;
Query OK, 0 rows affected (0.00 sec)

mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| ci_doctrine        |
| date_test          |
| mysql              |
| phpbb              |
| purse_db           |
| test               |
| wordpress          |
+--------------------+
8 rows in set (0.00 sec)

mysql> USE my_first_db
ERROR 1049 (42000): Unknown database 'my_first_db'
mysql> CREATE DATABASE my_first_db;
Query OK, 1 row affected (0.00 sec)

mysql> USE my_first_db
Database changed
mysql>
```

It tells MySQL to select a default database to work with, for the current session. Now we are ready to create tables and do other things under this database.

What is a Database Table?

You can think of a database table like a spreadsheet or csv file that holds structured data.


Untitled 1 - OpenOffice.org Calc
File Edit View Insert Format Tools Data Window Help
Arial
10
E8
Sheet1 Sheet2 Sheet3
Sheet 1 / 3
Default
STD
Sum=0

	A	B	C	D	E
1	user_id	username	email	create_date	
2	1	johndoe	johndoe@hotmail.com	12/01/09	
3	2	foorbar	test@test.com	12/05/09	
4	3	nettuts	nettuts@gmail.com	12/09/09	
5					
6					
7					
8					
9					
10					

Just like in this example, tables have column names, and rows of data. With SQL queries we can create these tables. We can also add, read, update and delete the data.

CREATE TABLE: Creating a Table

With this query we can create tables in the database. Unfortunately the MySQL documentation is not very friendly for new learners. The structure of this type of query can get very complex, but we will start with an easy one.

The following query will create a table with 2 columns.

```
CREATE TABLE users (
    username VARCHAR(20),
    create_date DATE
);
```

Note that we are able to write a query in multiple lines, and even use tabs for indentation.

First line is easy. We just create a table named 'users'. Following that, in parantheses, we have a list table columns separated by commas. After each column name, we have a data type, such as VARCHAR or DATE.

VARCHAR(20) means that the column is a string type, and can be a maximum of 20 characters long. DATE is also a data type that is specifically used for storing dates, in this format: 'YYYY-MM-DD'.

PRIMARY KEY

Before we run that query, we should also include a column for 'user_id', which will be a PRIMARY KEY. Without getting too much into the details, you can think of a PRIMARY KEY as a way to identify each row of data in a table.

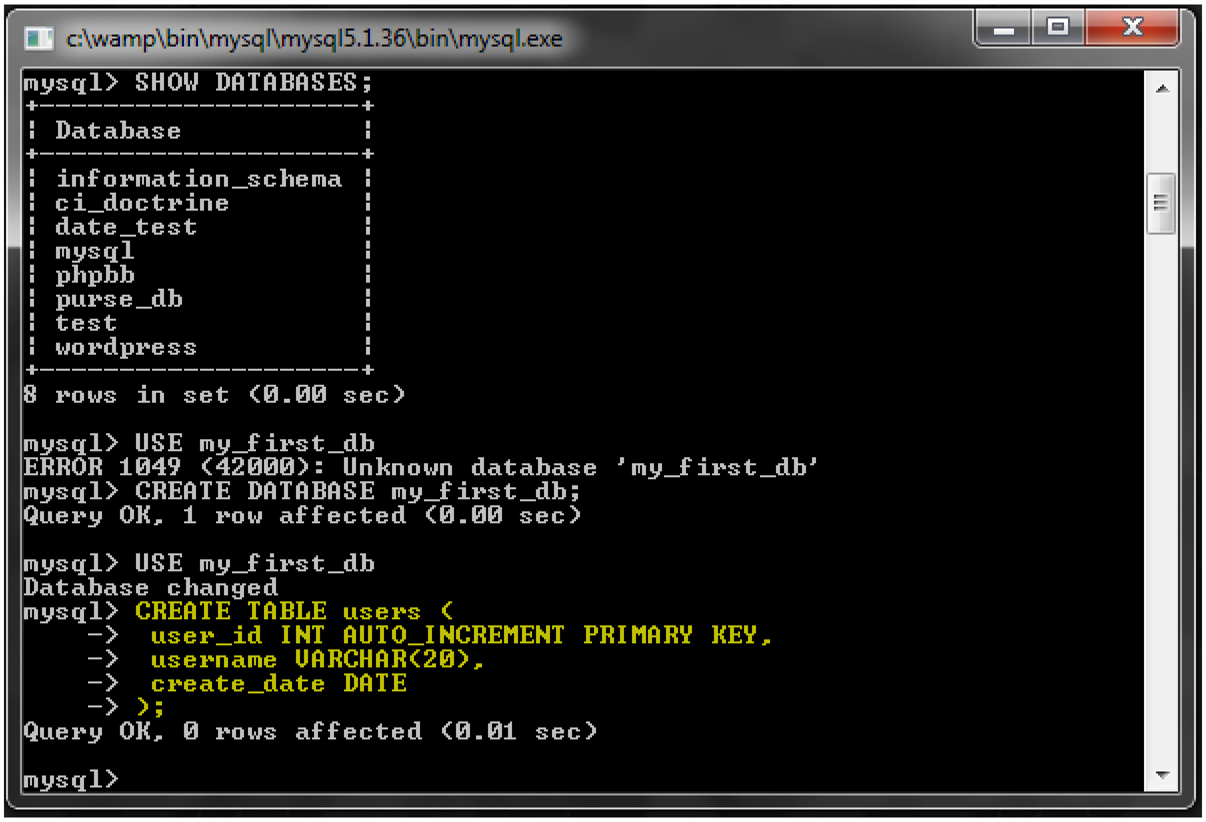
Now the query becomes:

```
CREATE TABLE users (
    user_id INT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(20),
    create_date DATE
);
```

INT makes this a 32bit integer type (i.e. numeric). AUTO_INCREMENT automatically generates a new id number every time we add new rows of data. It is not required, but makes it much more convenient.

This column does not have to be an integer, but it is the most commonly used type. Having a PRIMARY KEY column also is not required, but it is strongly recommended for good database design and performance.

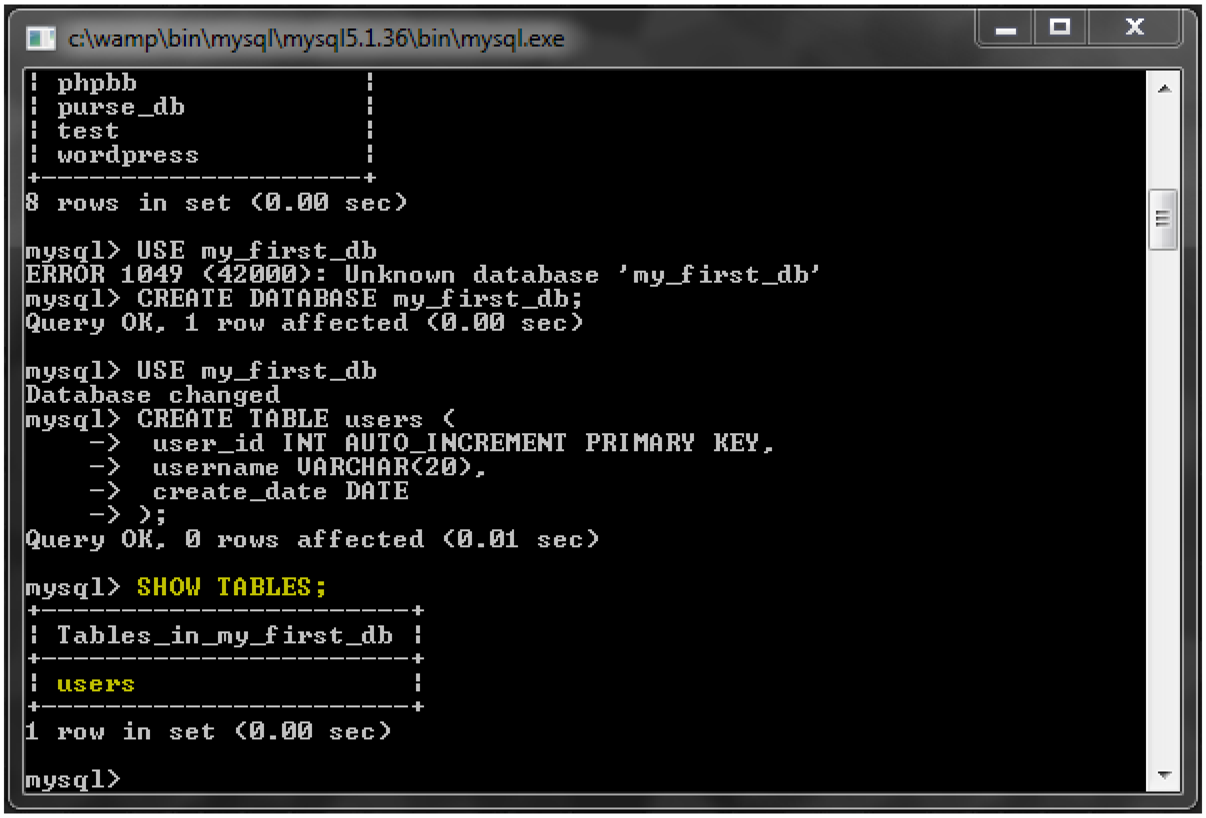
Let's run the query:



```
mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| ci_doctrine        |
| date_test          |
| mysql              |
| phpbb              |
| purse_db           |
| test               |
| wordpress          |
+--------------------+
8 rows in set (0.00 sec)

mysql> USE my_first_db
ERROR 1049 (42000): Unknown database 'my_first_db'
mysql> CREATE DATABASE my_first_db;
Query OK, 1 row affected (0.00 sec)

mysql> USE my_first_db
Database changed
mysql> CREATE TABLE users (
    ->  user_id INT AUTO_INCREMENT PRIMARY KEY,
    ->  username VARCHAR(20),
    ->  create_date DATE
    -> );
Query OK, 0 rows affected (0.01 sec)

mysql>
```

SHOW TABLES: List All Tables

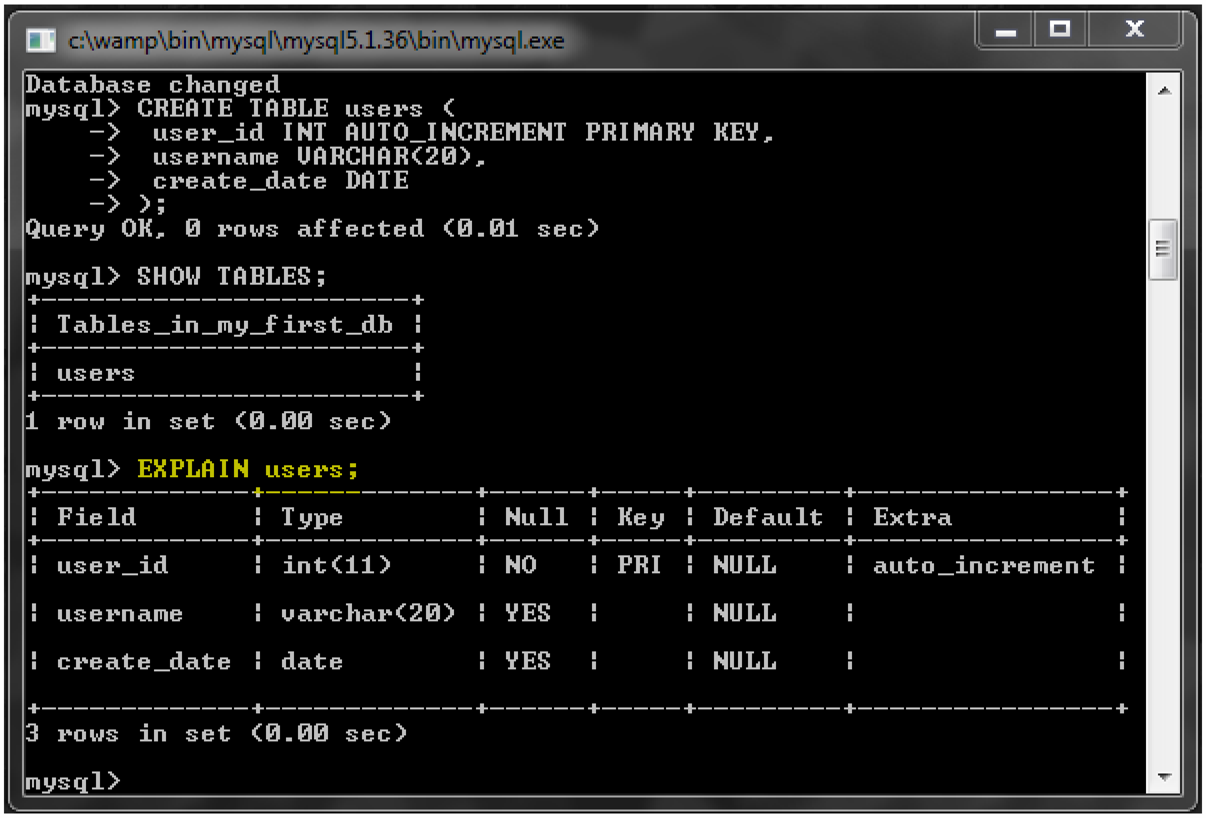
This query allows you to get a list of tables that are currently in the database.



```
| phpbb                |
| purse_db             |
| test                 |
| wordpress            |
+----------------------+
8 rows in set (0.00 sec)

mysql> USE my_first_db
ERROR 1049 (42000): Unknown database 'my_first_db'
mysql> CREATE DATABASE my_first_db;
Query OK, 1 row affected (0.00 sec)

mysql> USE my_first_db
Database changed
mysql> CREATE TABLE users (
    ->  user_id INT AUTO_INCREMENT PRIMARY KEY,
    ->  username VARCHAR(20),
    ->  create_date DATE
    -> );
Query OK, 0 rows affected (0.01 sec)

mysql> SHOW TABLES;
+-----------------------+
| Tables_in_my_first_db |
+-----------------------+
| users                 |
+-----------------------+
1 row in set (0.00 sec)

mysql>
```

EXPLAIN: Show Table Structure

To see the structure of an existing table, you can use this query.

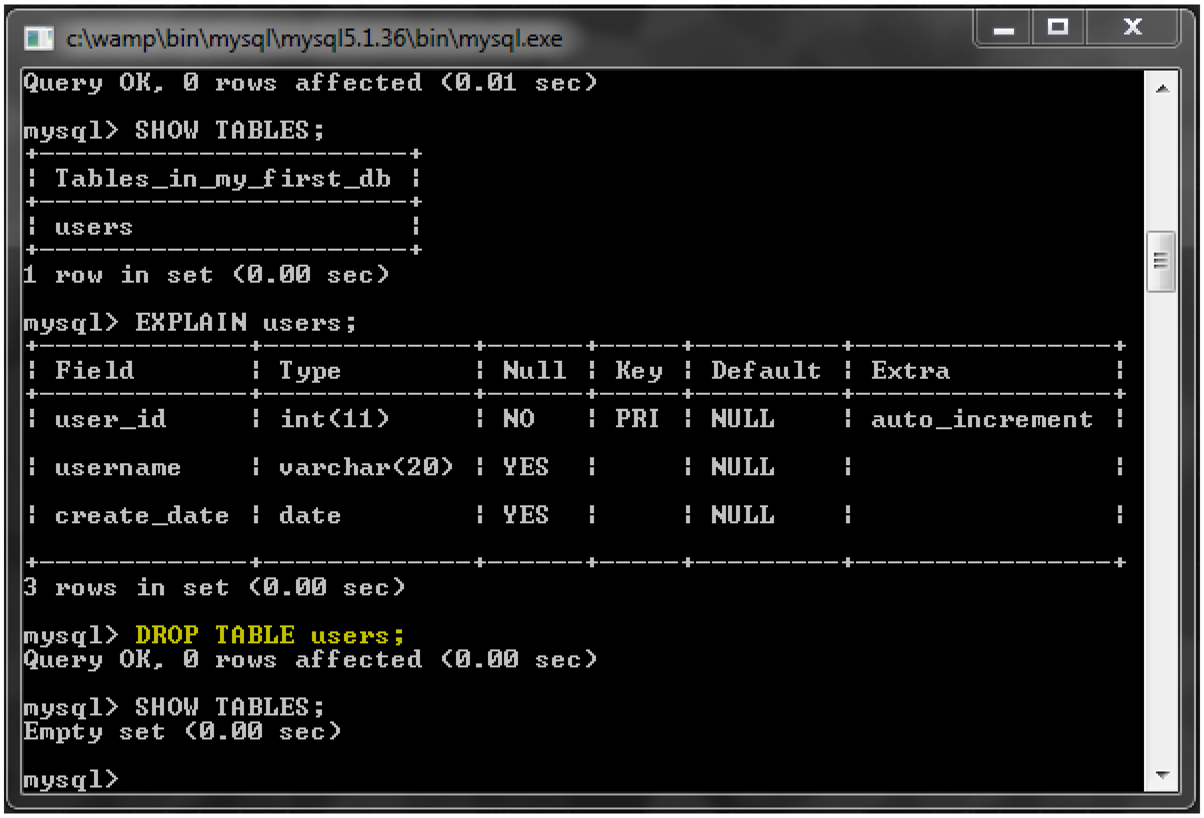


```
Database changed
mysql> CREATE TABLE users (
    ->  user_id INT AUTO_INCREMENT PRIMARY KEY,
    ->  username VARCHAR(20),
    ->  create_date DATE
    -> );
Query OK, 0 rows affected (0.01 sec)

mysql> SHOW TABLES;
+-----------------------+
| Tables_in_my_first_db |
+-----------------------+
| users                 |
+-----------------------+
1 row in set (0.00 sec)

mysql> EXPLAIN users;
+-------------+-------------+------+-----+---------+----------------+
| Field       | Type        | Null | Key | Default | Extra          |
+-------------+-------------+------+-----+---------+----------------+
| user_id     | int(11)     | NO   | PRI | NULL    | auto_increment |
| username    | varchar(20) | YES  |     | NULL    |                |
| create_date | date        | YES  |     | NULL    |                |
+-------------+-------------+------+-----+---------+----------------+
3 rows in set (0.00 sec)

mysql>
```

Fields (aka. columns) are listed in the results, with their properties.

DROP TABLE: Delete a Table

Just like DROP DATABASES, this query deletes a table and its contents, without a warning.



```
Query OK, 0 rows affected (0.01 sec)

mysql> SHOW TABLES;
+------------------------+
| Tables_in_my_first_db  |
+------------------------+
| users                  |
+------------------------+
1 row in set (0.00 sec)

mysql> EXPLAIN users;
+-------------+-------------+------+-----+---------+----------------+
| Field       | Type        | Null | Key | Default | Extra          |
+-------------+-------------+------+-----+---------+----------------+
| user_id     | int(11)     | NO   | PRI | NULL    | auto_increment |
| username    | varchar(20) | YES  |     | NULL    |                |
| create_date | date        | YES  |     | NULL    |                |
+-------------+-------------+------+-----+---------+----------------+
3 rows in set (0.00 sec)

mysql> DROP TABLE users;
Query OK, 0 rows affected (0.00 sec)

mysql> SHOW TABLES;
Empty set (0.00 sec)

mysql>
```

ALTER TABLE: Modify a Table

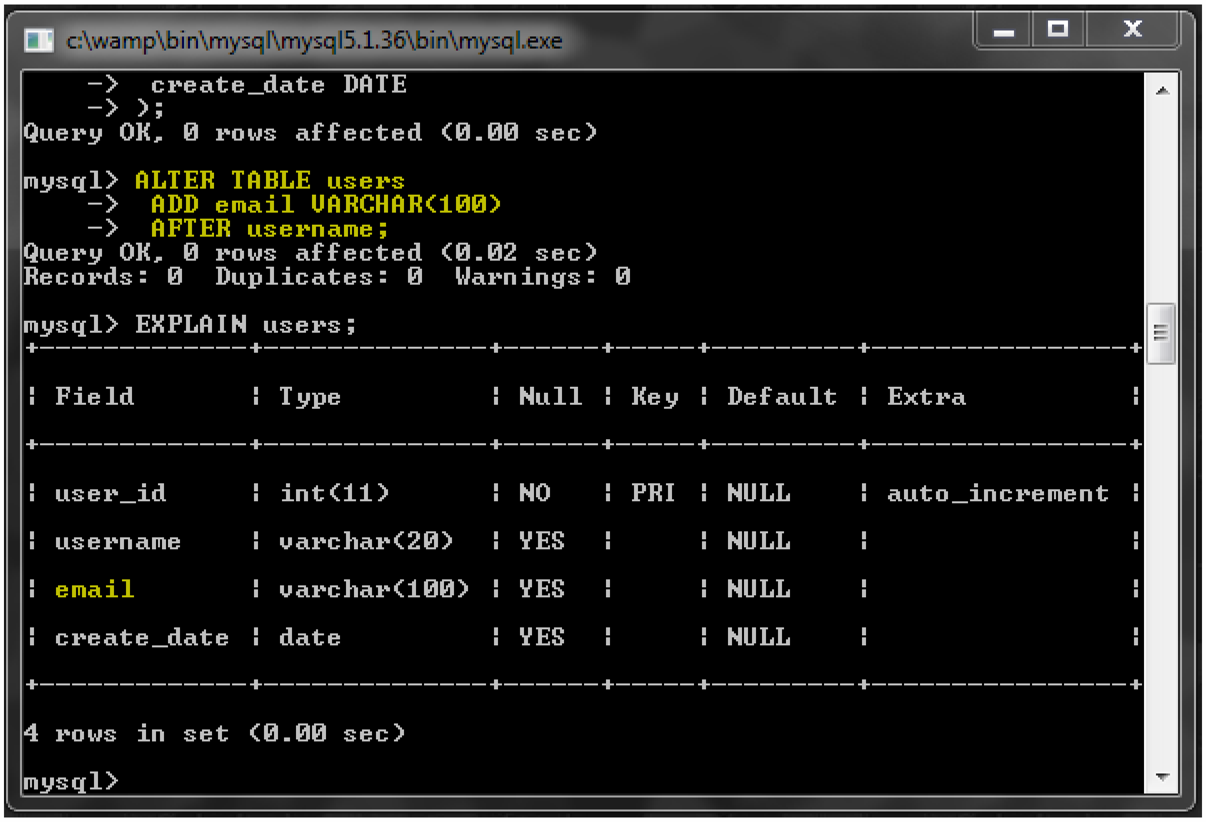
This query also can have quite a complex structure because of the multitude of changes it can perform on a table. Let's look at some simple examples.

(Make sure to re-create the table we just dropped or the following queries obviously won't work.)

Add a Column

```
ALTER TABLE users
    ADD email VARCHAR(100)
    AFTER username;
```

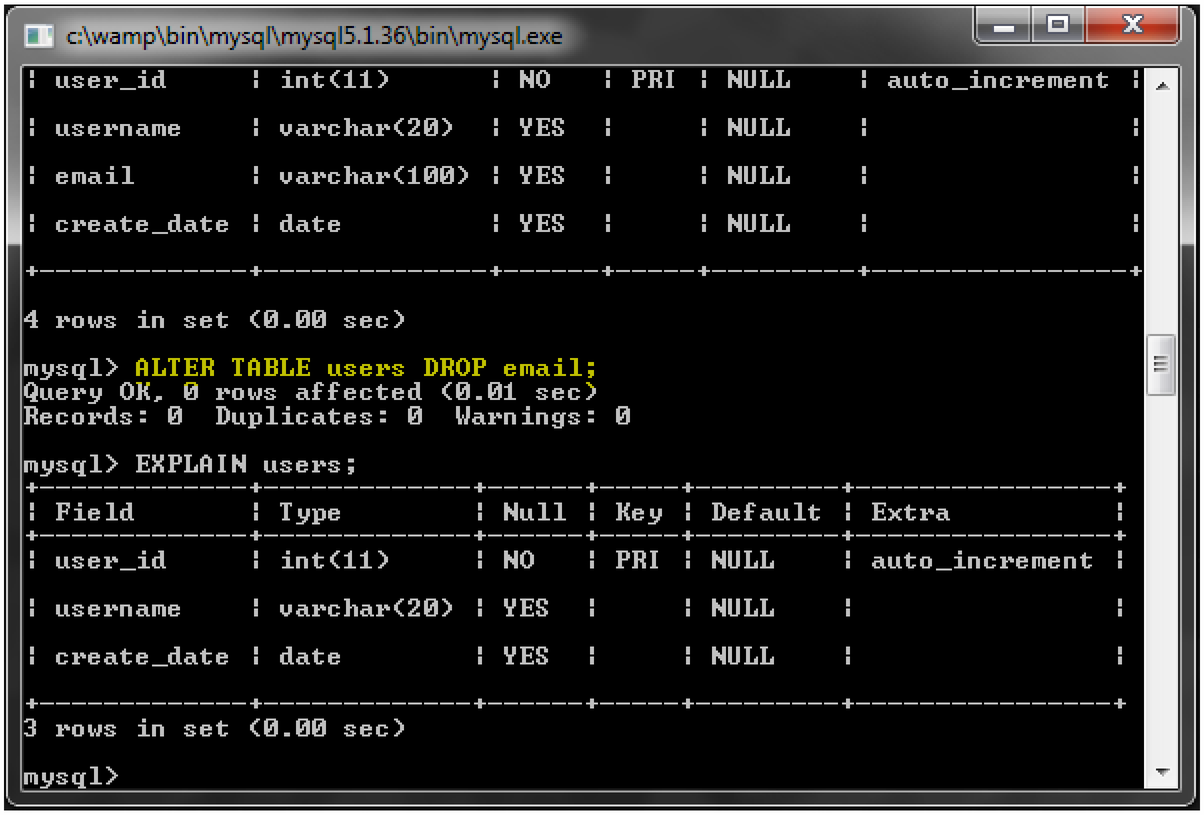
Thanks to the readability of SQL, I don't think that query even needs an explanation.



c:\wamp\bin\mysql\mysql5.1.36\bin\mysql.exe

```
    -> create_date DATE
    -> );
Query OK, 0 rows affected (0.00 sec)

mysql> ALTER TABLE users
    ->  ADD email VARCHAR(100)
    ->  AFTER username;
Query OK, 0 rows affected (0.02 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> EXPLAIN users;
+-------------+--------------+------+-----+---------+----------------+
| Field       | Type         | Null | Key | Default | Extra          |
+-------------+--------------+------+-----+---------+----------------+
| user_id     | int(11)      | NO   | PRI | NULL    | auto_increment |
| username    | varchar(20)  | YES  |     | NULL    |                |
| email       | varchar(100) | YES  |     | NULL    |                |
| create_date | date         | YES  |     | NULL    |                |
+-------------+--------------+------+-----+---------+----------------+
4 rows in set (0.00 sec)

mysql>
```

Remove a Column



```
c:\wamp\bin\mysql\mysql5.1.36\bin\mysql.exe
| user_id     | int(11)      | NO   | PRI | NULL    | auto_increment |
| username    | varchar(20)  | YES  |     | NULL    |                |
| email       | varchar(100) | YES  |     | NULL    |                |
| create_date | date         | YES  |     | NULL    |                |
+-------------+--------------+------+-----+---------+----------------+
4 rows in set (0.00 sec)

mysql> ALTER TABLE users DROP email;
Query OK, 0 rows affected (0.01 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> EXPLAIN users;
+-------------+-------------+------+-----+---------+----------------+
| Field       | Type        | Null | Key | Default | Extra          |
+-------------+-------------+------+-----+---------+----------------+
| user_id     | int(11)     | NO   | PRI | NULL    | auto_increment |
| username    | varchar(20) | YES  |     | NULL    |                |
| create_date | date        | YES  |     | NULL    |                |
+-------------+-------------+------+-----+---------+----------------+
3 rows in set (0.00 sec)

mysql>
```

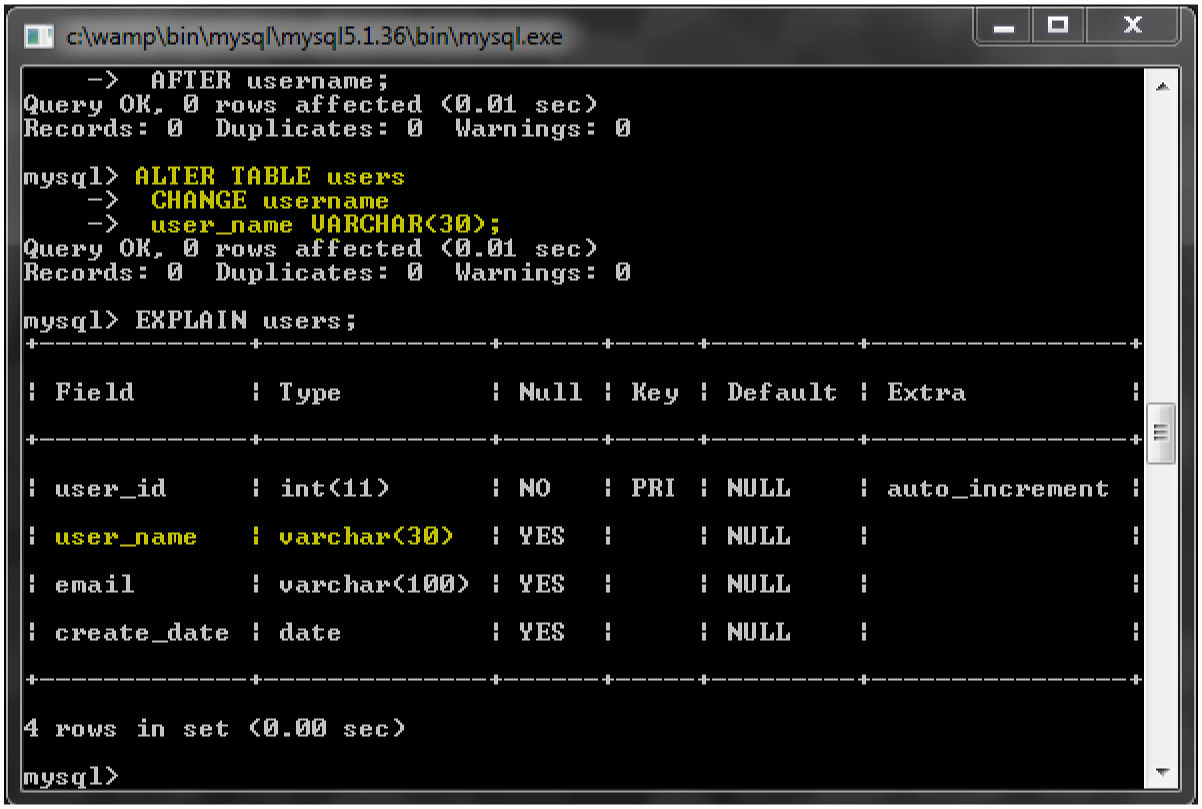
That was also very simple. But use it with caution as it permanently removes data without a warning.

Re-add the email column because we are going to be using it later:

```
ALTER TABLE users
    ADD email VARCHAR(100)
    AFTER username;
```

Modify a Column

Sometimes you may want to change the properties of a column, so you don't have to delete and recreate it.

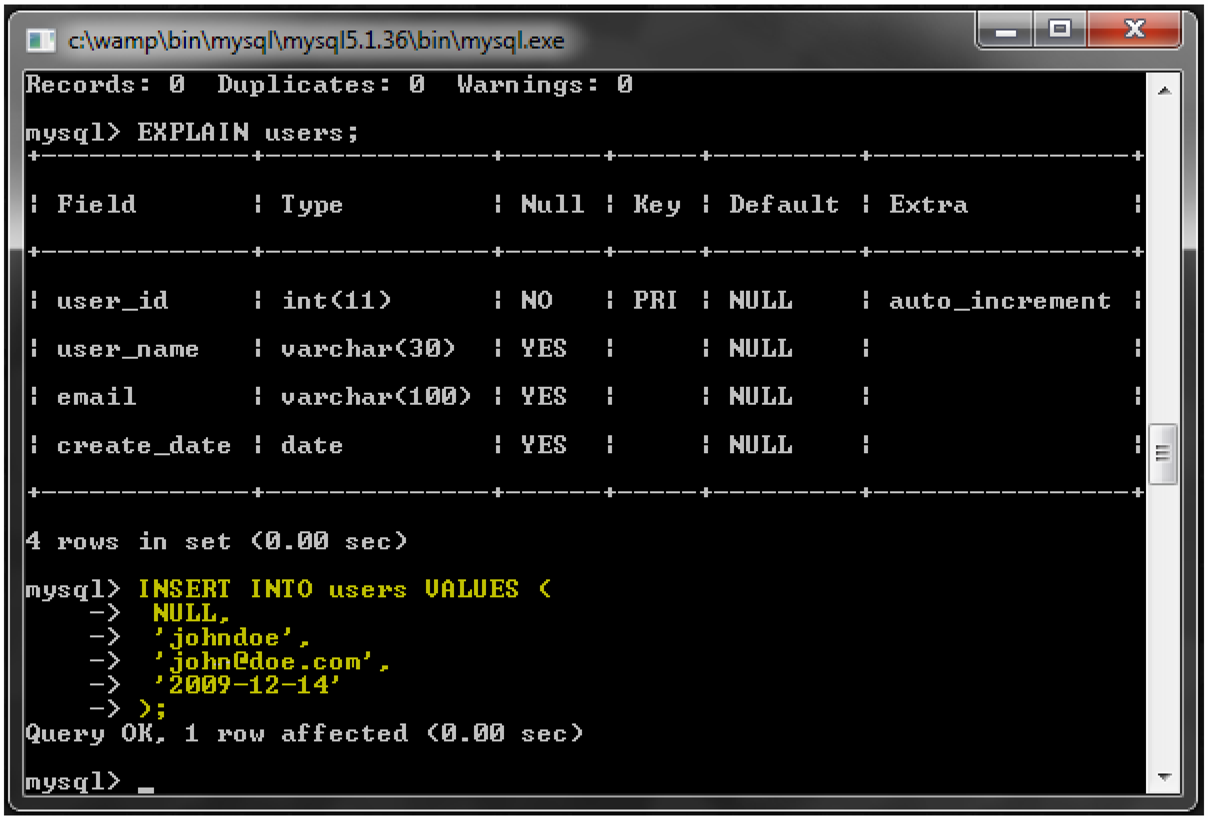


```
    -> AFTER username;
Query OK, 0 rows affected (0.01 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> ALTER TABLE users
    -> CHANGE username
    -> user_name VARCHAR(30);
Query OK, 0 rows affected (0.01 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> EXPLAIN users;
+-------------+--------------+------+-----+---------+----------------+
| Field       | Type         | Null | Key | Default | Extra          |
+-------------+--------------+------+-----+---------+----------------+
| user_id     | int(11)      | NO   | PRI | NULL    | auto_increment |
| user_name   | varchar(30)  | YES  |     | NULL    |                |
| email       | varchar(100) | YES  |     | NULL    |                |
| create_date | date         | YES  |     | NULL    |                |
+-------------+--------------+------+-----+---------+----------------+
4 rows in set (0.00 sec)

mysql>
```

That renamed the username column to 'user_name' and changed the type from VARCHAR(20) to VARCHAR(30). A change like this should not disturb any of the existing data in the table.

INSERT: Add Data to a Table

Let's add some data into the table using this query.



c:\wamp\bin\mysql\mysql5.1.36\bin\mysql.exe

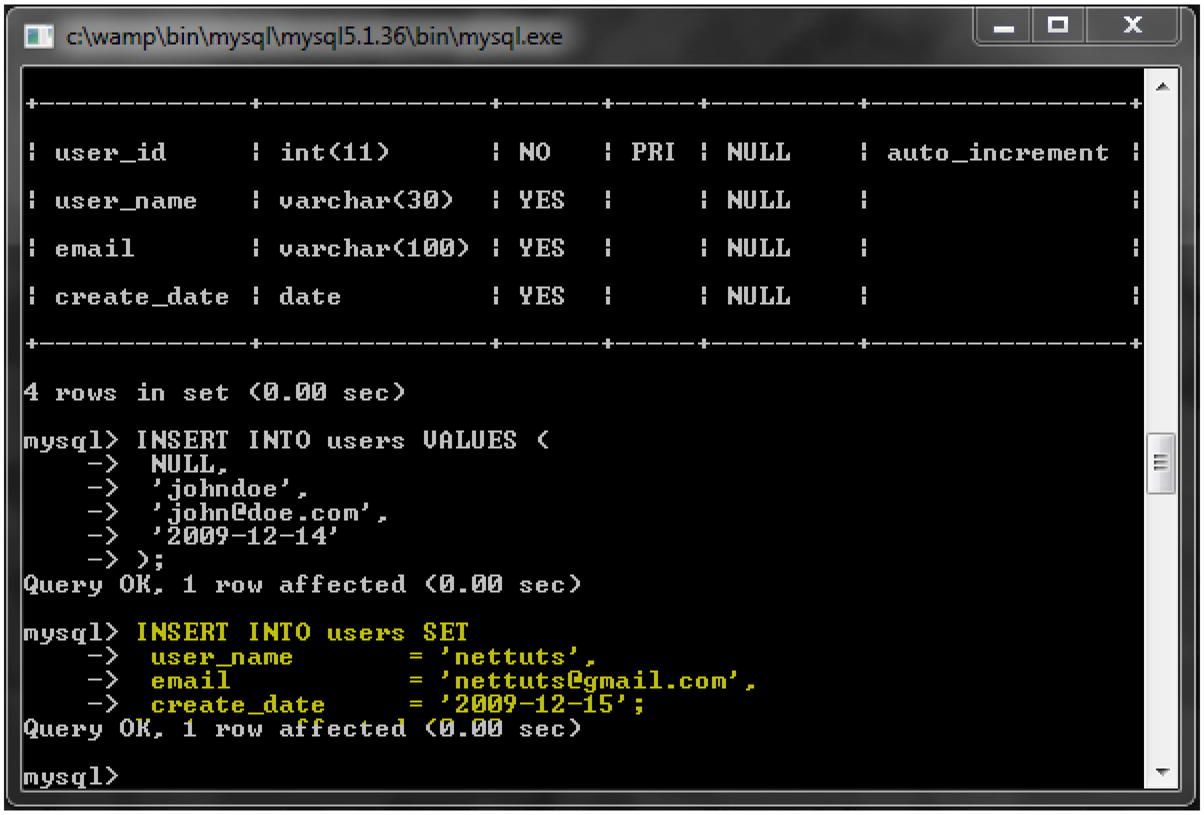
```
Records: 0  Duplicates: 0  Warnings: 0

mysql> EXPLAIN users;
+-------------+--------------+------+-----+---------+----------------+
| Field       | Type         | Null | Key | Default | Extra          |
+-------------+--------------+------+-----+---------+----------------+
| user_id     | int(11)      | NO   | PRI | NULL    | auto_increment |
| user_name   | varchar(30)  | YES  |     | NULL    |                |
| email       | varchar(100) | YES  |     | NULL    |                |
| create_date | date         | YES  |     | NULL    |                |
+-------------+--------------+------+-----+---------+----------------+
4 rows in set (0.00 sec)

mysql> INSERT INTO users VALUES (
    ->  NULL,
    ->  'johndoe',
    ->  'john@doe.com',
    ->  '2009-12-14'
    -> );
Query OK, 1 row affected (0.00 sec)

mysql> _
```

As you can see, VALUES() contains the list of field values, separated by commas. The string values are enclosed in single quotes. And the values need to be in the order of the columns that were defined when we created the table.

Note that the first value is NULL for the PRIMARY KEY field we called 'user_id'. We do this so that an id is automatically generated, because the column is set to AUTO_INCREMENT. When entering a row of data for the first time, the id will be 1. Next inserted row will be 2 and so on...

Alternate Syntax

Here is another syntax for inserting rows.



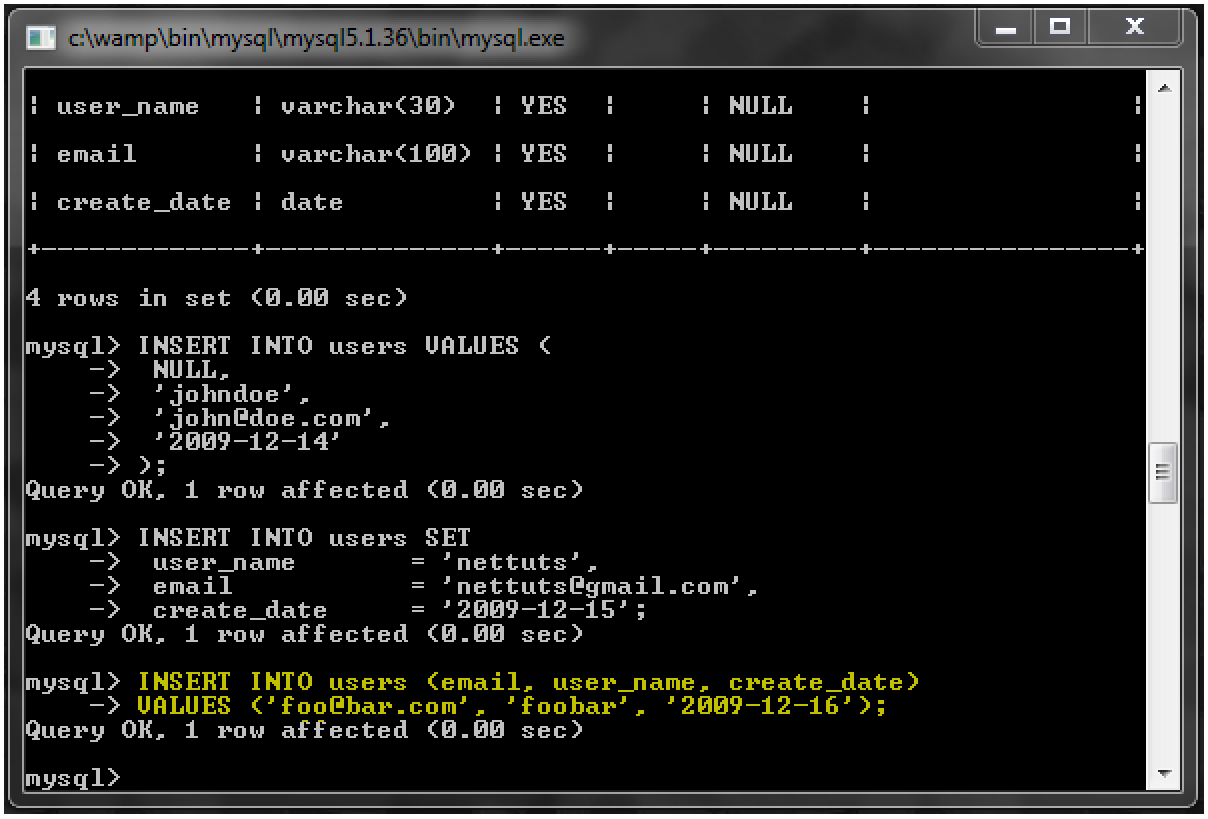
```
+-------------+--------------+------+-----+---------+----------------+
| user_id     | int(11)      | NO   | PRI | NULL    | auto_increment |
| user_name   | varchar(30)  | YES  |     | NULL    |                |
| email       | varchar(100) | YES  |     | NULL    |                |
| create_date | date         | YES  |     | NULL    |                |
+-------------+--------------+------+-----+---------+----------------+
4 rows in set (0.00 sec)

mysql> INSERT INTO users VALUES (
    ->  NULL,
    ->  'johndoe',
    ->  'john@doe.com',
    ->  '2009-12-14'
    -> );
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO users SET
    ->  user_name       = 'nettuts',
    ->  email           = 'nettuts@gmail.com',
    ->  create_date     = '2009-12-15';
Query OK, 1 row affected (0.00 sec)

mysql>
```

This time we are using the keyword SET instead of VALUES, and it is not followed by paratheses. There are a few things to note here:

- A column can be omitted. For example we did not assign a value to user_id, which will default to the AUTO_INCREMENT functionality. If you omit a VARCHAR column, it would default to an empty string (unless a different default value was specified during table creation).
- Each column has to be referenced by its name. Because of this, they can be in any order, unlike the previous syntax.

Alternate Syntax 2

Here is yet another syntax.

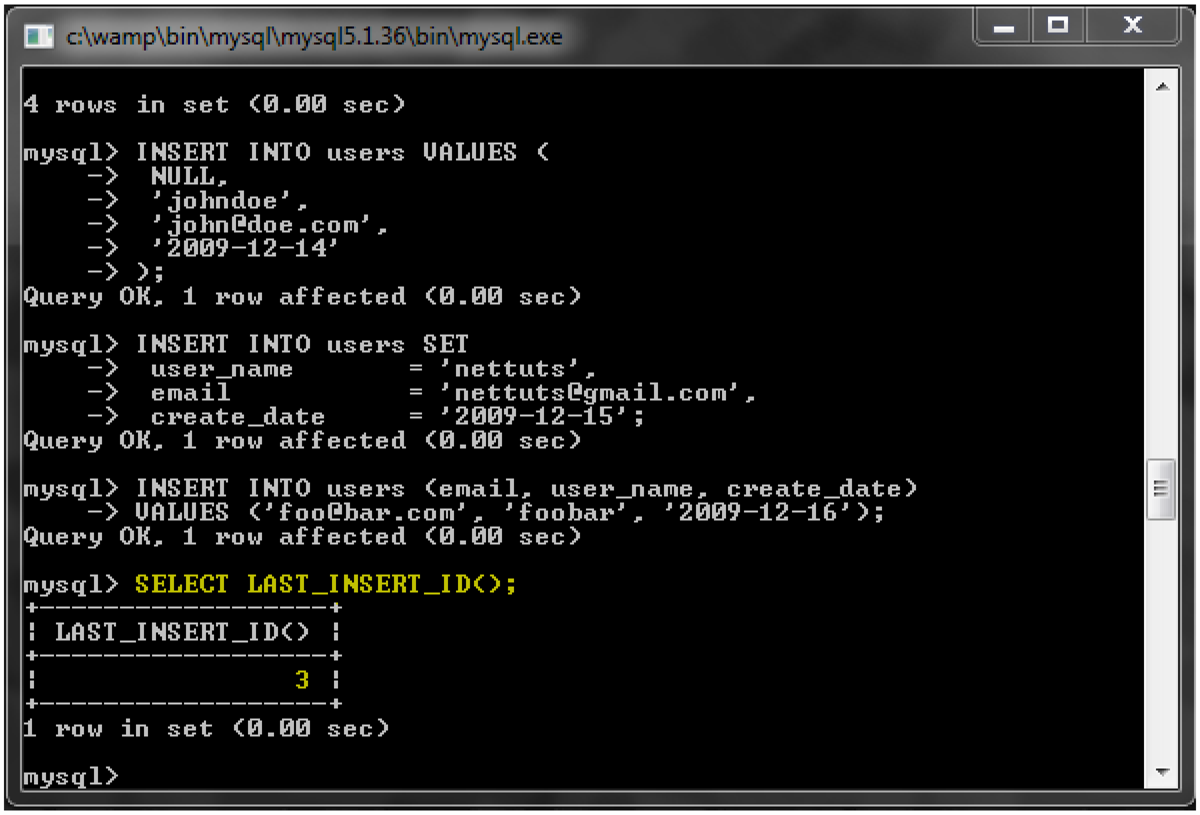


```
c:\wamp\bin\mysql\mysql5.1.36\bin\mysql.exe
| user_name   | varchar(30)  | YES  |     | NULL    |               |
| email       | varchar(100) | YES  |     | NULL    |               |
| create_date | date         | YES  |     | NULL    |               |
+-------------+--------------+------+-----+---------+---------------+
4 rows in set (0.00 sec)

mysql> INSERT INTO users VALUES (
    ->  NULL,
    ->  'johndoe',
    ->  'john@doe.com',
    ->  '2009-12-14'
    -> );
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO users SET
    ->  user_name        = 'nettuts',
    ->  email            = 'nettuts@gmail.com',
    ->  create_date      = '2009-12-15';
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO users (email, user_name, create_date)
    -> VALUES ('foo@bar.com', 'foobar', '2009-12-16');
Query OK, 1 row affected (0.00 sec)

mysql>
```

Again, since each column is referenced by name, they can be in any order.

LAST_INSERT_ID()

You can use this query to get the AUTO_INCREMENT id for the last inserted row, in the current session.

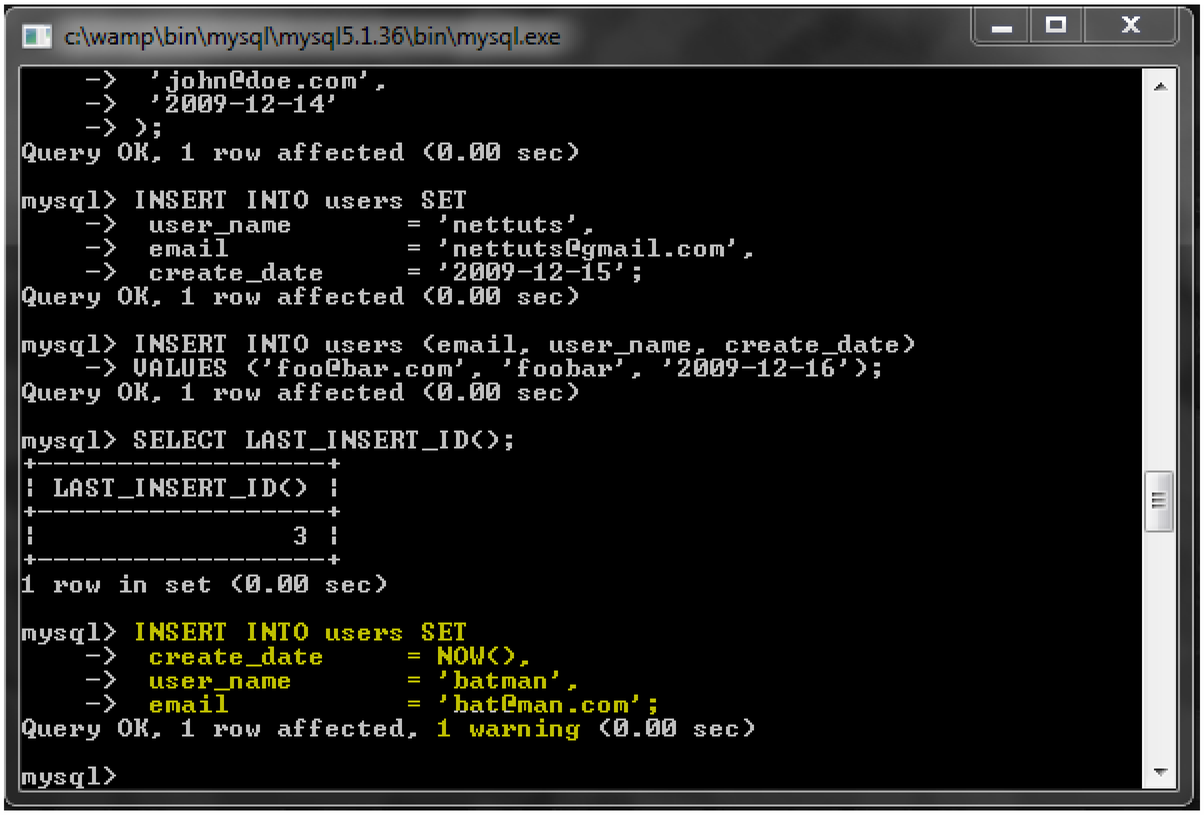


```
4 rows in set (0.00 sec)

mysql> INSERT INTO users VALUES (
    ->  NULL,
    ->  'johndoe',
    ->  'john@doe.com',
    ->  '2009-12-14'
    -> );
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO users SET
    ->  user_name        = 'nettuts',
    ->  email            = 'nettuts@gmail.com',
    ->  create_date      = '2009-12-15';
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO users (email, user_name, create_date)
    -> VALUES ('foo@bar.com', 'foobar', '2009-12-16');
Query OK, 1 row affected (0.00 sec)

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

mysql>
```

NOW()

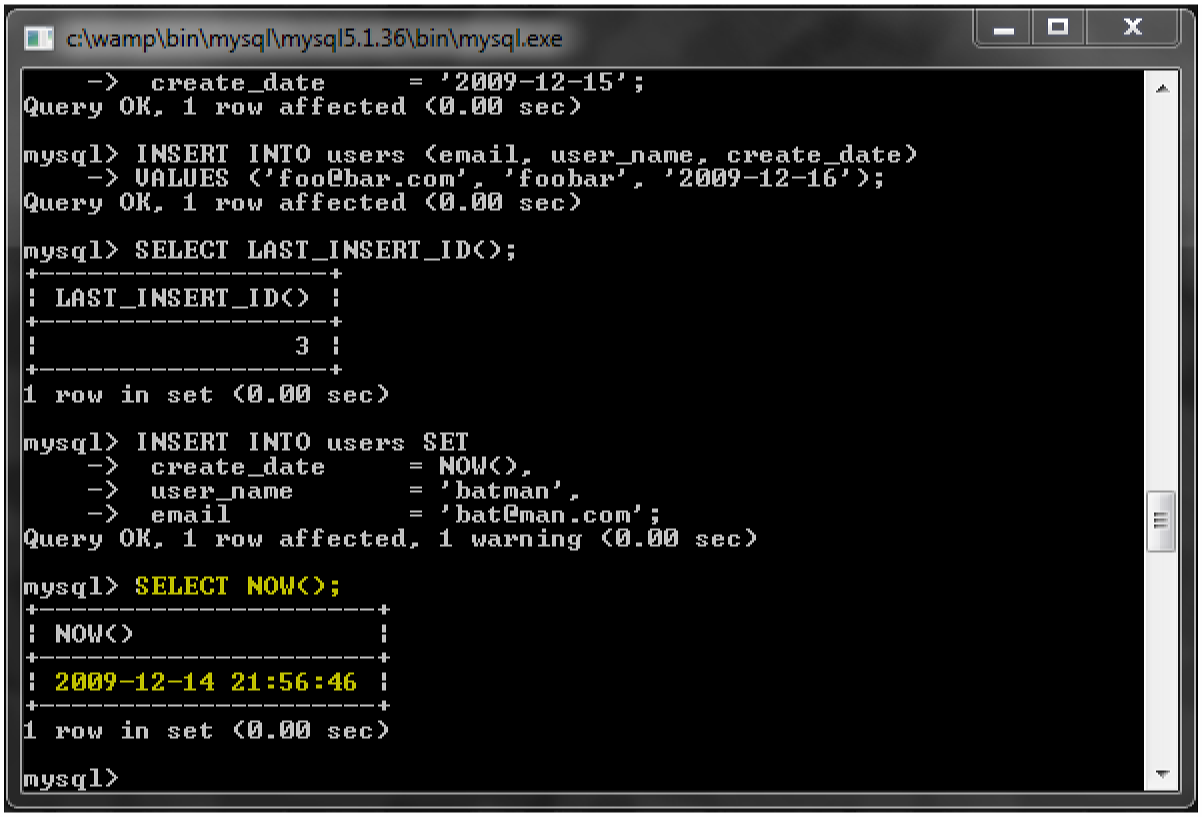
I think it is a good time to demonstrate how you can use a MySQL function inside your queries.

The NOW() function returns the current date. So you can use it to automatically set a DATE column to the current day while inserting a new row.



```
    -> 'john@doe.com',
    -> '2009-12-14'
    -> );
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO users SET
    -> user_name        = 'nettuts',
    -> email            = 'nettuts@gmail.com',
    -> create_date      = '2009-12-15';
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO users (email, user_name, create_date)
    -> VALUES ('foo@bar.com', 'foobar', '2009-12-16');
Query OK, 1 row affected (0.00 sec)

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

mysql> INSERT INTO users SET
    -> create_date      = NOW(),
    -> user_name        = 'batman',
    -> email            = 'bat@man.com';
Query OK, 1 row affected, 1 warning (0.00 sec)

mysql>
```

Note that we received a warning from MySQL, but it is not a big deal. The reason is that NOW() actually returns time information as well.



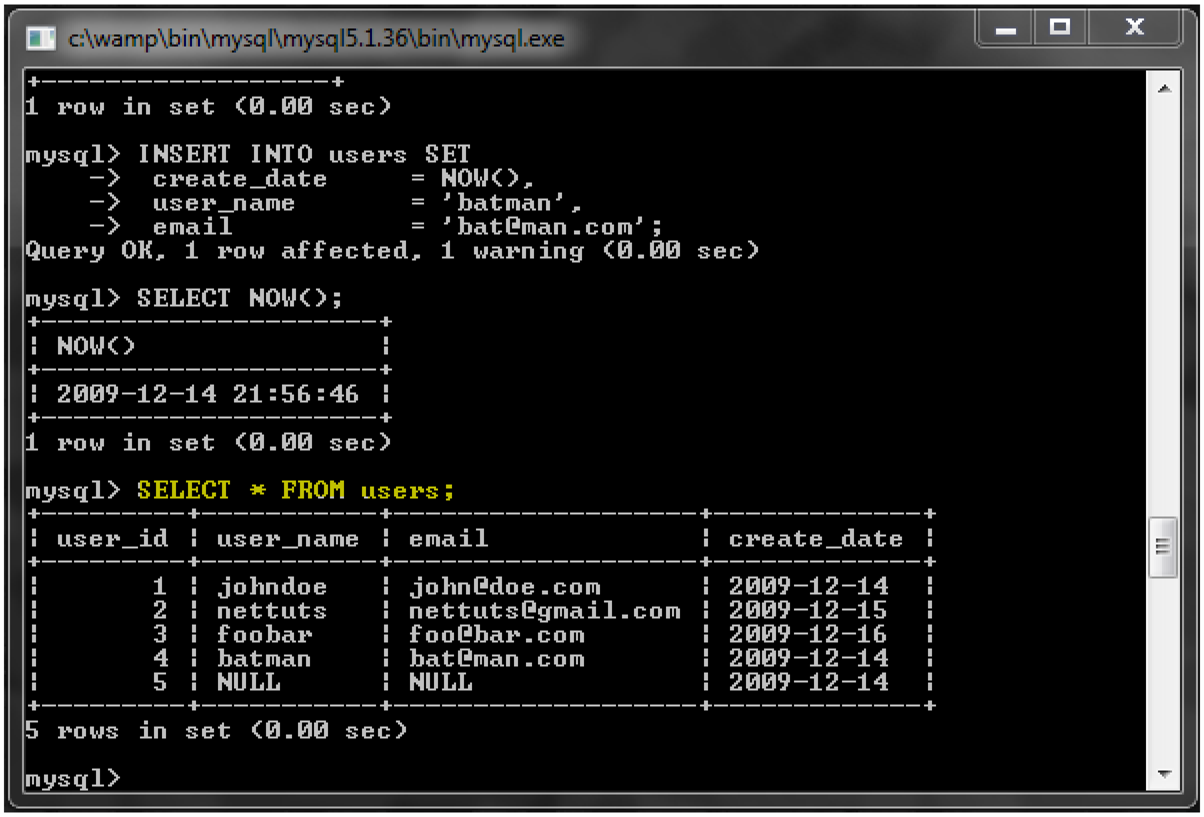
```
    -> create_date      = '2009-12-15';
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO users (email, user_name, create_date)
    -> VALUES ('foo@bar.com', 'foobar', '2009-12-16');
Query OK, 1 row affected (0.00 sec)

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

mysql> INSERT INTO users SET
    -> create_date      = NOW(),
    -> user_name        = 'batman',
    -> email            = 'bat@man.com';
Query OK, 1 row affected, 1 warning (0.00 sec)

mysql> SELECT NOW();
+---------------------+
| NOW()               |
+---------------------+
| 2009-12-14 21:56:46 |
+---------------------+
1 row in set (0.00 sec)

mysql>
```

But the create_date column we created only contains the date, and not the time, therefor the returned data was truncated. We could use the CURDATE() function instead, which returns just the date, but the data stored at the end would be the same either way.

SELECT: Read Data from a Table

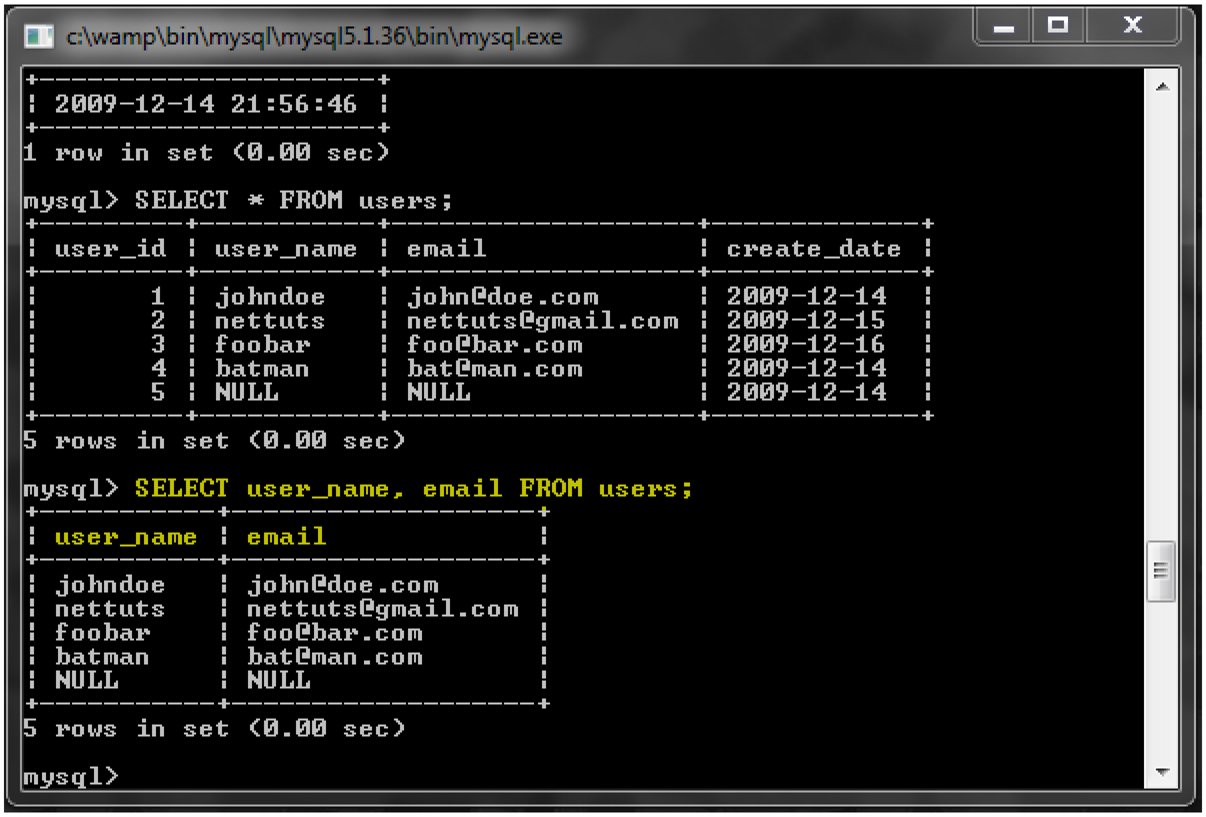
Obviously the data we added would be useless unless we can read it. This is where the SELECT query comes in.

Here is the simplest possible SELECT query for reading from a table:



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

mysql> INSERT INTO users SET
    -> create_date      = NOW(),
    -> user_name        = 'batman',
    -> email            = 'bat@man.com';
Query OK, 1 row affected, 1 warning (0.00 sec)

mysql> SELECT NOW();
+---------------------+
| NOW()               |
+---------------------+
| 2009-12-14 21:56:46 |
+---------------------+
1 row in set (0.00 sec)

mysql> SELECT * FROM users;
+---------+-----------+--------------------+-------------+
| user_id | user_name | email              | create_date |
+---------+-----------+--------------------+-------------+
|       1 | johndoe   | john@doe.com       | 2009-12-14  |
|       2 | nettuts   | nettuts@gmail.com  | 2009-12-15  |
|       3 | foobar    | foo@bar.com        | 2009-12-16  |
|       4 | batman    | bat@man.com        | 2009-12-14  |
|       5 | NULL      | NULL               | 2009-12-14  |
+---------+-----------+--------------------+-------------+
5 rows in set (0.00 sec)

mysql>
```

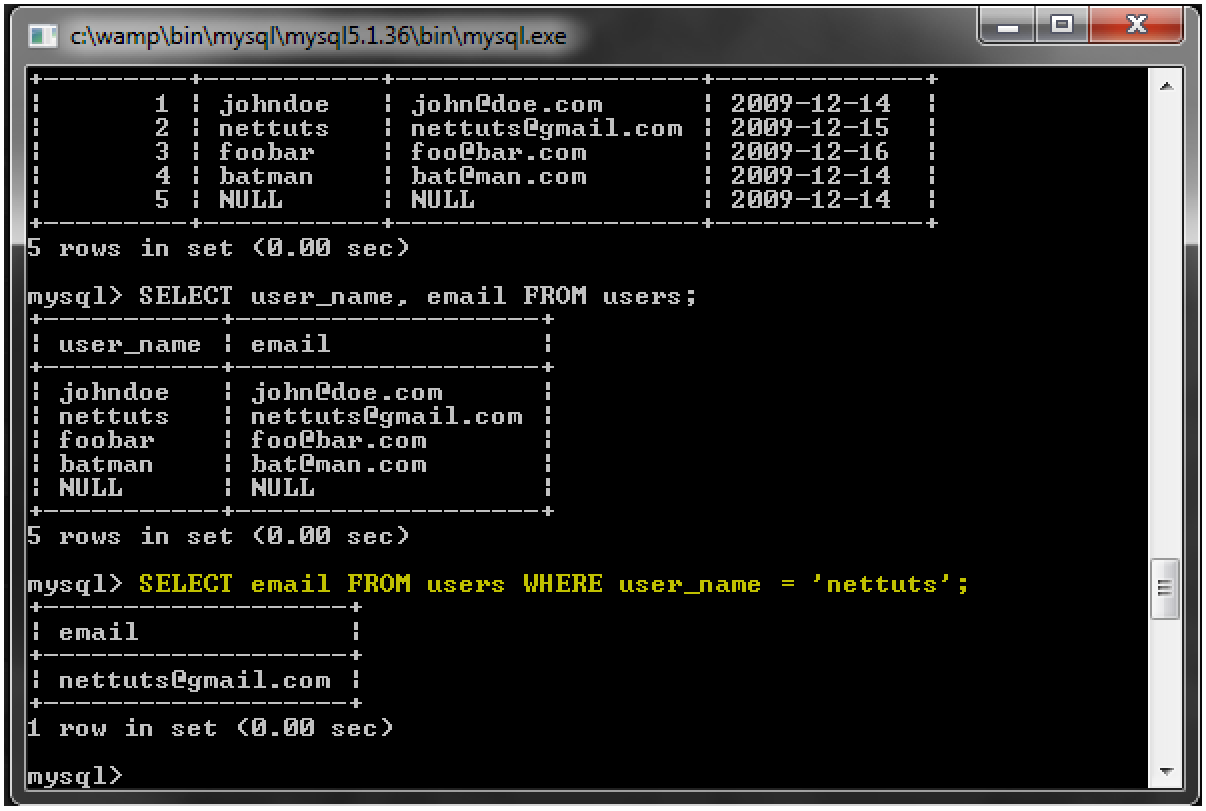
In this case, the asterisk (*) means that we asked to fetch all the columns from the table. If you want only specific columns, the query would look like this:


c:\wamp\bin\mysql\mysql5.1.36\bin\mysql.exe

```
+---------------------+
| 2009-12-14 21:56:46 |
+---------------------+
1 row in set (0.00 sec)

mysql> SELECT * FROM users;
+---------+-----------+-------------------+-------------+
| user_id | user_name | email             | create_date |
+---------+-----------+-------------------+-------------+
|       1 | johndoe   | john@doe.com      | 2009-12-14  |
|       2 | nettuts   | nettuts@gmail.com | 2009-12-15  |
|       3 | foobar    | foo@bar.com       | 2009-12-16  |
|       4 | batman    | bat@man.com       | 2009-12-14  |
|       5 | NULL      | NULL              | 2009-12-14  |
+---------+-----------+-------------------+-------------+
5 rows in set (0.00 sec)

mysql> SELECT user_name, email FROM users;
+-----------+-------------------+
| user_name | email             |
+-----------+-------------------+
| johndoe   | john@doe.com      |
| nettuts   | nettuts@gmail.com |
| foobar    | foo@bar.com       |
| batman    | bat@man.com       |
| NULL      | NULL              |
+-----------+-------------------+
5 rows in set (0.00 sec)

mysql>
```

WHERE Clause

More often than not, we are only interested in some of the rows, and not all. For example, let's say we want the email address for the user 'nettuts'.

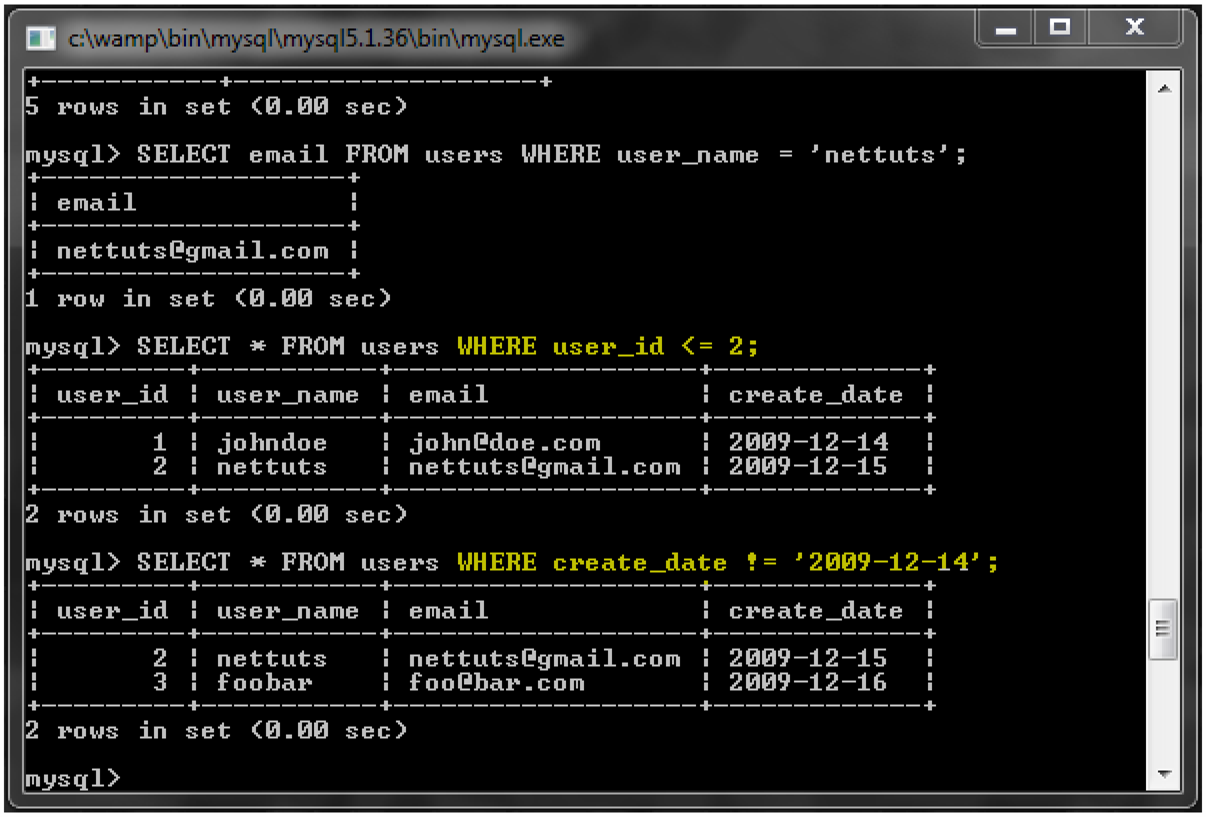


```
+----------+------------+--------------------+-------------+
|        1 | johndoe    | john@doe.com       | 2009-12-14  |
|        2 | nettuts    | nettuts@gmail.com  | 2009-12-15  |
|        3 | foobar     | foo@bar.com        | 2009-12-16  |
|        4 | batman     | bat@man.com        | 2009-12-14  |
|        5 | NULL       | NULL               | 2009-12-14  |
+----------+------------+--------------------+-------------+
5 rows in set (0.00 sec)

mysql> SELECT user_name, email FROM users;
+-----------+-------------------+
| user_name | email             |
+-----------+-------------------+
| johndoe   | john@doe.com      |
| nettuts   | nettuts@gmail.com |
| foobar    | foo@bar.com       |
| batman    | bat@man.com       |
| NULL      | NULL              |
+-----------+-------------------+
5 rows in set (0.00 sec)

mysql> SELECT email FROM users WHERE user_name = 'nettuts';
+-------------------+
| email             |
+-------------------+
| nettuts@gmail.com |
+-------------------+
1 row in set (0.00 sec)

mysql>
```

Think of it like an IF statement. WHERE allows you to put conditions in the query for the results you are looking for.

Note that for the equality condition, only a single equal sign is used (=), instead of double (==) which you might be used to from programming.

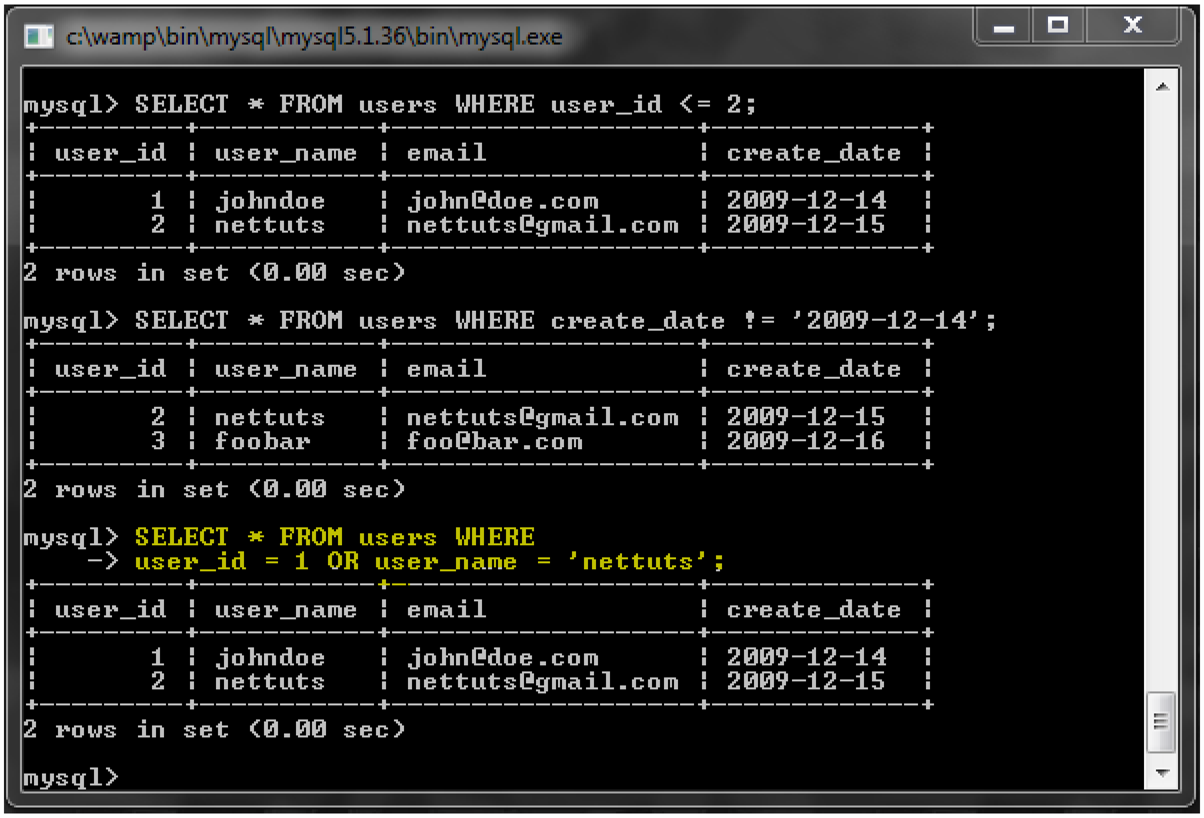
You can use other comparison conditions too:



c:\wamp\bin\mysql\mysql5.1.36\bin\mysql.exe

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

mysql> SELECT email FROM users WHERE user_name = 'nettuts';
+-------------------+
| email             |
+-------------------+
| nettuts@gmail.com |
+-------------------+
1 row in set (0.00 sec)

mysql> SELECT * FROM users WHERE user_id <= 2;
+---------+-----------+-------------------+-------------+
| user_id | user_name | email             | create_date |
+---------+-----------+-------------------+-------------+
|       1 | johndoe   | john@doe.com      | 2009-12-14  |
|       2 | nettuts   | nettuts@gmail.com | 2009-12-15  |
+---------+-----------+-------------------+-------------+
2 rows in set (0.00 sec)

mysql> SELECT * FROM users WHERE create_date != '2009-12-14';
+---------+-----------+-------------------+-------------+
| user_id | user_name | email             | create_date |
+---------+-----------+-------------------+-------------+
|       2 | nettuts   | nettuts@gmail.com | 2009-12-15  |
|       3 | foobar    | foo@bar.com       | 2009-12-16  |
+---------+-----------+-------------------+-------------+
2 rows in set (0.00 sec)

mysql>
```

AND and OR can be used to combine conditions:

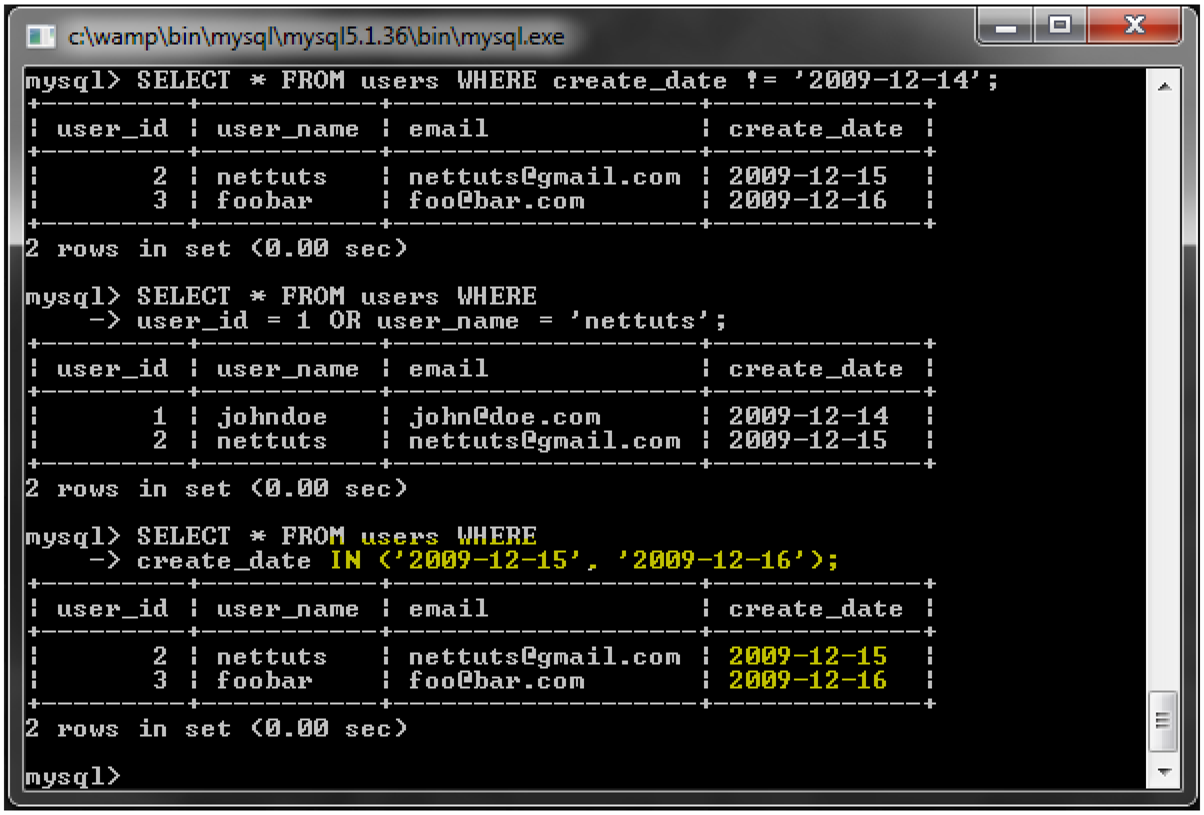

c:\wamp\bin\mysql\mysql5.1.36\bin\mysql.exe

```
mysql> SELECT * FROM users WHERE user_id <= 2;
+---------+-----------+---------------------+-------------+
| user_id | user_name | email               | create_date |
+---------+-----------+---------------------+-------------+
|       1 | johndoe   | john@doe.com        | 2009-12-14  |
|       2 | nettuts   | nettuts@gmail.com   | 2009-12-15  |
+---------+-----------+---------------------+-------------+
2 rows in set (0.00 sec)

mysql> SELECT * FROM users WHERE create_date != '2009-12-14';
+---------+-----------+---------------------+-------------+
| user_id | user_name | email               | create_date |
+---------+-----------+---------------------+-------------+
|       2 | nettuts   | nettuts@gmail.com   | 2009-12-15  |
|       3 | foobar    | foo@bar.com         | 2009-12-16  |
+---------+-----------+---------------------+-------------+
2 rows in set (0.00 sec)

mysql> SELECT * FROM users WHERE
    -> user_id = 1 OR user_name = 'nettuts';
+---------+-----------+---------------------+-------------+
| user_id | user_name | email               | create_date |
+---------+-----------+---------------------+-------------+
|       1 | johndoe   | john@doe.com        | 2009-12-14  |
|       2 | nettuts   | nettuts@gmail.com   | 2009-12-15  |
+---------+-----------+---------------------+-------------+
2 rows in set (0.00 sec)

mysql>
```

Note that numeric values do not have to be inside quotes.

IN()

This is useful for matching multiple values.

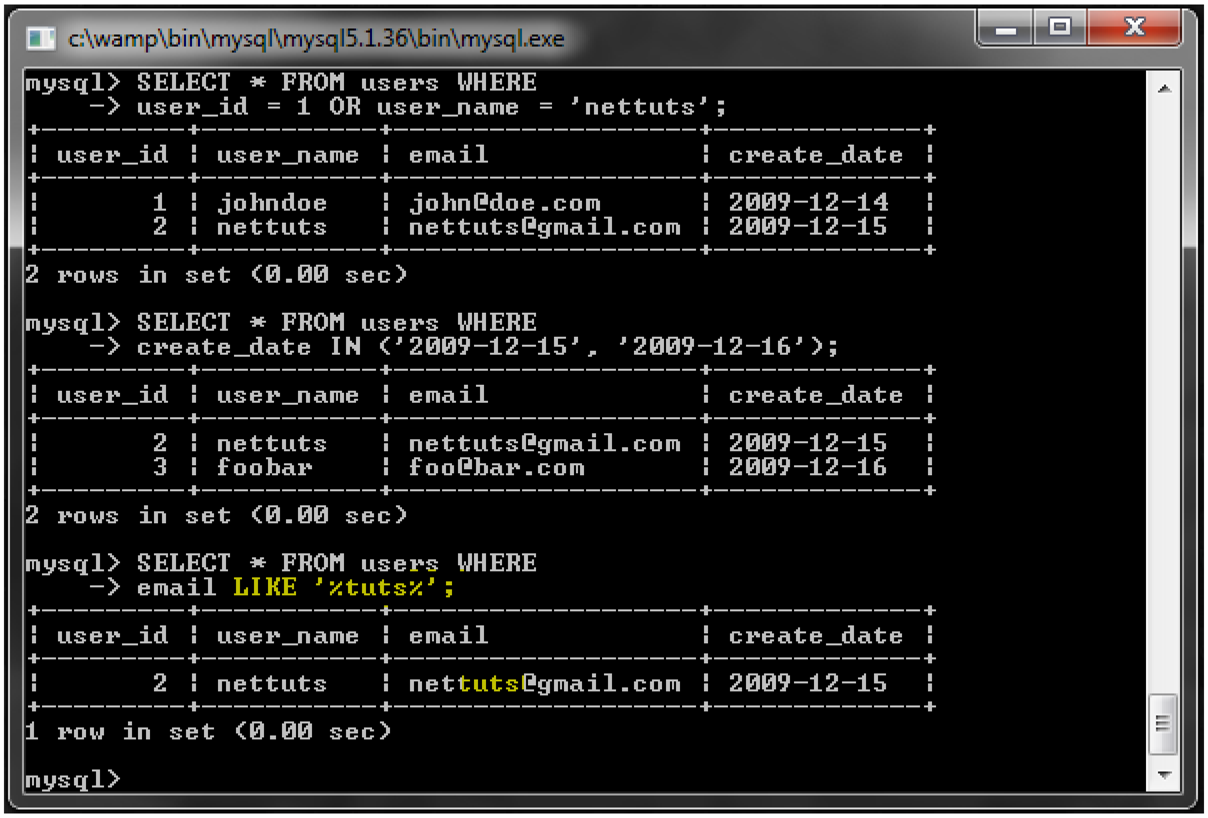


c:\wamp\bin\mysql\mysql5.1.36\bin\mysql.exe

```
mysql> SELECT * FROM users WHERE create_date != '2009-12-14';
+---------+-----------+-------------------+-------------+
| user_id | user_name | email             | create_date |
+---------+-----------+-------------------+-------------+
|       2 | nettuts   | nettuts@gmail.com | 2009-12-15  |
|       3 | foobar    | foo@bar.com       | 2009-12-16  |
+---------+-----------+-------------------+-------------+
2 rows in set (0.00 sec)

mysql> SELECT * FROM users WHERE
    -> user_id = 1 OR user_name = 'nettuts';
+---------+-----------+-------------------+-------------+
| user_id | user_name | email             | create_date |
+---------+-----------+-------------------+-------------+
|       1 | johndoe   | john@doe.com      | 2009-12-14  |
|       2 | nettuts   | nettuts@gmail.com | 2009-12-15  |
+---------+-----------+-------------------+-------------+
2 rows in set (0.00 sec)

mysql> SELECT * FROM users WHERE
    -> create_date IN ('2009-12-15', '2009-12-16');
+---------+-----------+-------------------+-------------+
| user_id | user_name | email             | create_date |
+---------+-----------+-------------------+-------------+
|       2 | nettuts   | nettuts@gmail.com | 2009-12-15  |
|       3 | foobar    | foo@bar.com       | 2009-12-16  |
+---------+-----------+-------------------+-------------+
2 rows in set (0.00 sec)

mysql>
```

LIKE

This allows you to do wildcard searches.

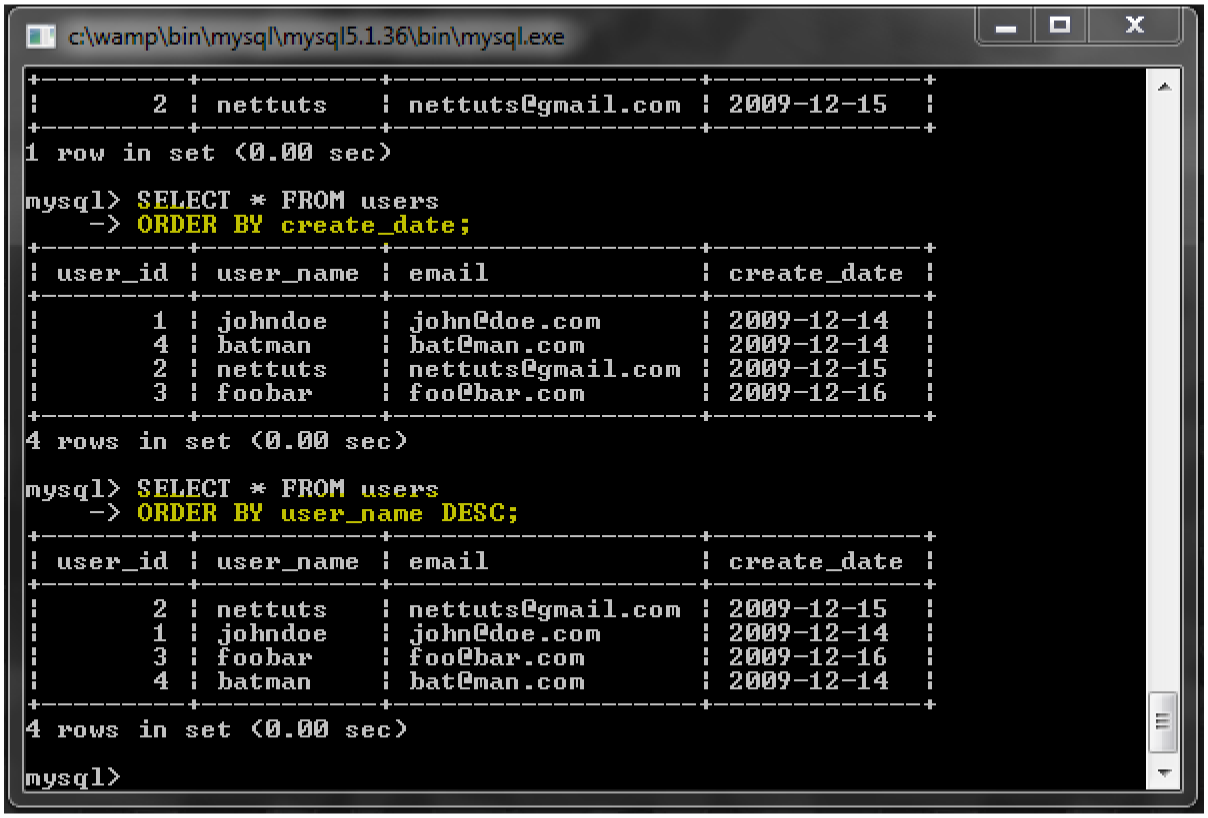

c:\wamp\bin\mysql\mysql5.1.36\bin\mysql.exe

```
mysql> SELECT * FROM users WHERE
    -> user_id = 1 OR user_name = 'nettuts';
+---------+-----------+-------------------+-------------+
| user_id | user_name | email             | create_date |
+---------+-----------+-------------------+-------------+
|       1 | johndoe   | john@doe.com      | 2009-12-14  |
|       2 | nettuts   | nettuts@gmail.com | 2009-12-15  |
+---------+-----------+-------------------+-------------+
2 rows in set (0.00 sec)

mysql> SELECT * FROM users WHERE
    -> create_date IN ('2009-12-15', '2009-12-16');
+---------+-----------+-------------------+-------------+
| user_id | user_name | email             | create_date |
+---------+-----------+-------------------+-------------+
|       2 | nettuts   | nettuts@gmail.com | 2009-12-15  |
|       3 | foobar    | foo@bar.com       | 2009-12-16  |
+---------+-----------+-------------------+-------------+
2 rows in set (0.00 sec)

mysql> SELECT * FROM users WHERE
    -> email LIKE '%tuts%';
+---------+-----------+-------------------+-------------+
| user_id | user_name | email             | create_date |
+---------+-----------+-------------------+-------------+
|       2 | nettuts   | nettuts@gmail.com | 2009-12-15  |
+---------+-----------+-------------------+-------------+
1 row in set (0.00 sec)

mysql>
```

Percentage sign (%) is used as the wildcard.

ORDER BY Clause

If you want the results to be returned in a specific order, use this clause:

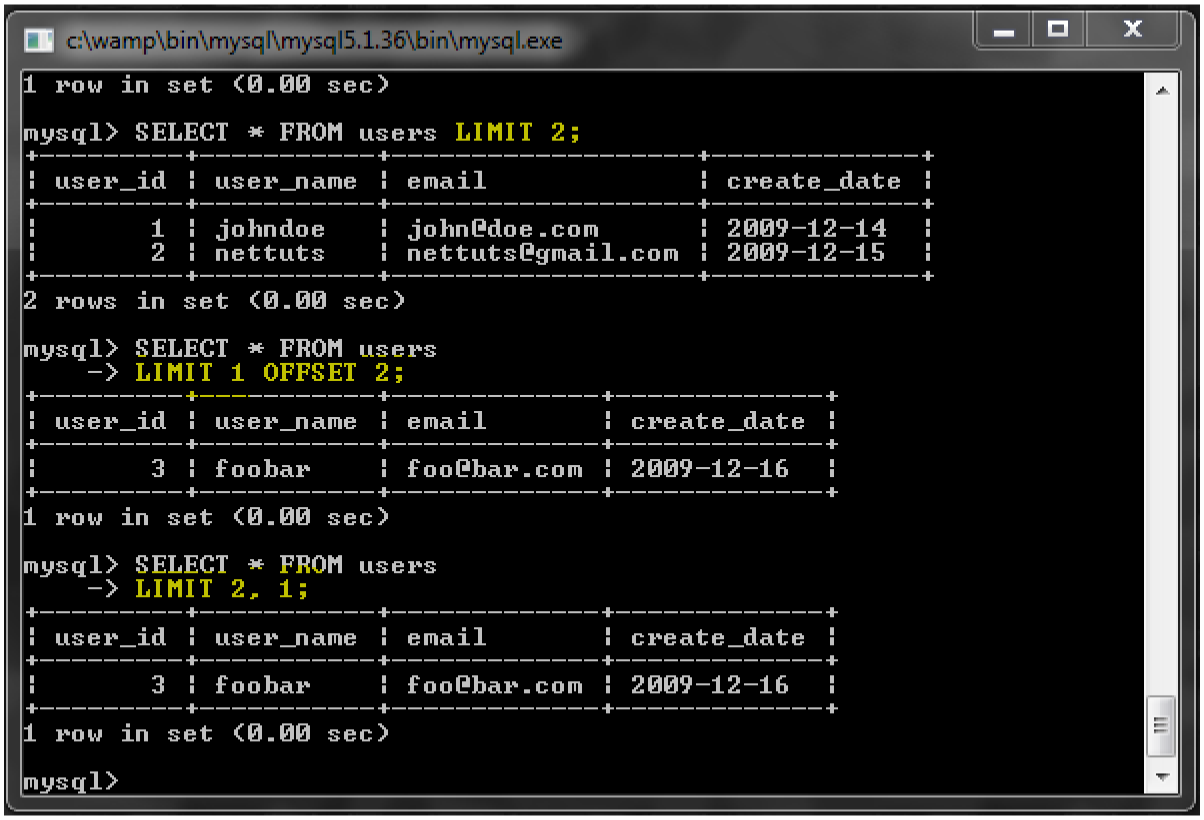


```
+---------+-----------+--------------------+-------------+
|       2 | nettuts   | nettuts@gmail.com  | 2009-12-15  |
+---------+-----------+--------------------+-------------+
1 row in set (0.00 sec)

mysql> SELECT * FROM users
    -> ORDER BY create_date;
+---------+-----------+--------------------+-------------+
| user_id | user_name | email              | create_date |
+---------+-----------+--------------------+-------------+
|       1 | johndoe   | john@doe.com       | 2009-12-14  |
|       4 | batman    | bat@man.com        | 2009-12-14  |
|       2 | nettuts   | nettuts@gmail.com  | 2009-12-15  |
|       3 | foobar    | foo@bar.com        | 2009-12-16  |
+---------+-----------+--------------------+-------------+
4 rows in set (0.00 sec)

mysql> SELECT * FROM users
    -> ORDER BY user_name DESC;
+---------+-----------+--------------------+-------------+
| user_id | user_name | email              | create_date |
+---------+-----------+--------------------+-------------+
|       2 | nettuts   | nettuts@gmail.com  | 2009-12-15  |
|       1 | johndoe   | john@doe.com       | 2009-12-14  |
|       3 | foobar    | foo@bar.com        | 2009-12-16  |
|       4 | batman    | bat@man.com        | 2009-12-14  |
+---------+-----------+--------------------+-------------+
4 rows in set (0.00 sec)

mysql>
```

The default order is ASC (i.e. ascending). You can add DESC to reverse order it.

LIMIT ... OFFSET ...

You can limit the number of returned results.

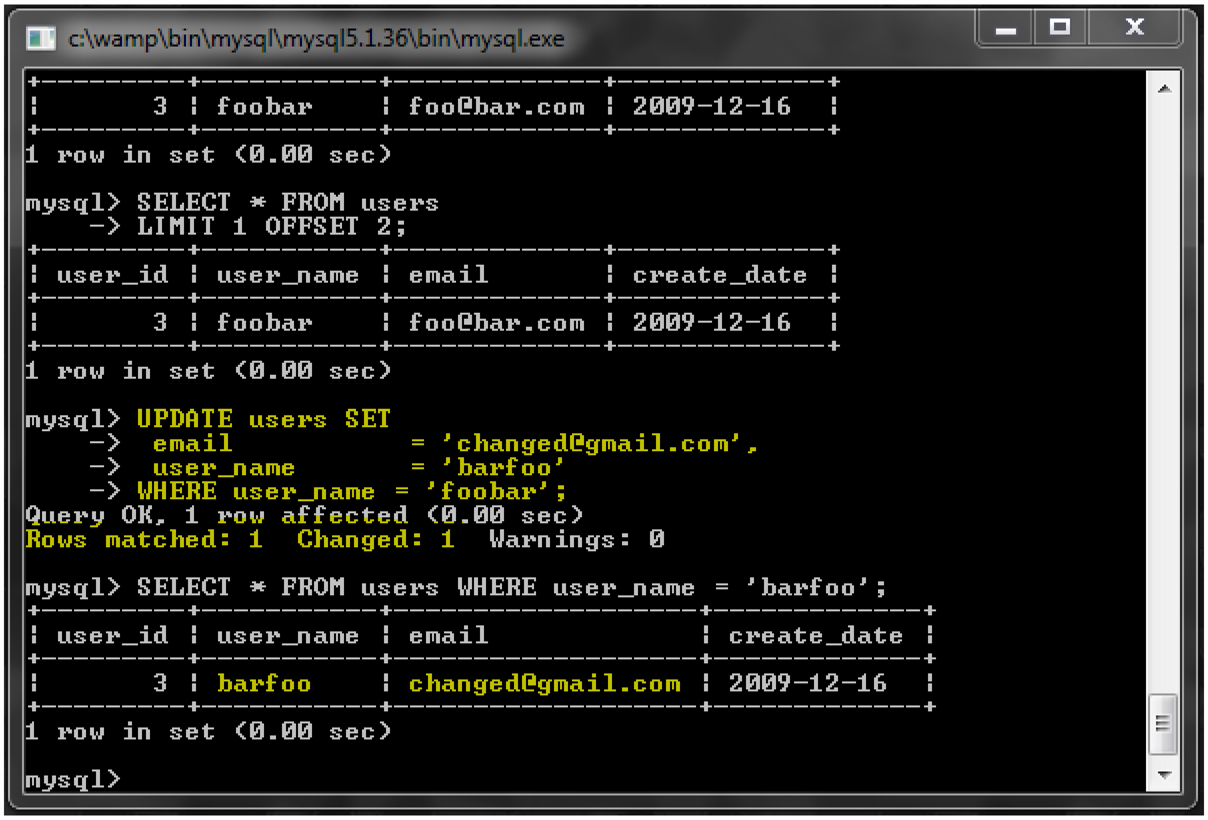

c:\wamp\bin\mysql\mysql5.1.36\bin\mysql.exe

```
1 row in set (0.00 sec)

mysql> SELECT * FROM users LIMIT 2;
+---------+-----------+-------------------+-------------+
| user_id | user_name | email             | create_date |
+---------+-----------+-------------------+-------------+
|       1 | johndoe   | john@doe.com      | 2009-12-14  |
|       2 | nettuts   | nettuts@gmail.com | 2009-12-15  |
+---------+-----------+-------------------+-------------+
2 rows in set (0.00 sec)

mysql> SELECT * FROM users
    -> LIMIT 1 OFFSET 2;
+---------+-----------+-------------+-------------+
| user_id | user_name | email       | create_date |
+---------+-----------+-------------+-------------+
|       3 | foobar    | foo@bar.com | 2009-12-16  |
+---------+-----------+-------------+-------------+
1 row in set (0.00 sec)

mysql> SELECT * FROM users
    -> LIMIT 2, 1;
+---------+-----------+-------------+-------------+
| user_id | user_name | email       | create_date |
+---------+-----------+-------------+-------------+
|       3 | foobar    | foo@bar.com | 2009-12-16  |
+---------+-----------+-------------+-------------+
1 row in set (0.00 sec)

mysql>
```

LIMIT 2 just gets the first 2 results. LIMIT 1 OFFSET 2 gets 1 result, after the first 2 results. LIMIT 2, 1 means the same thing, but note that the first number is the offset and the second number is the limit.

UPDATE: Update Data in a Table

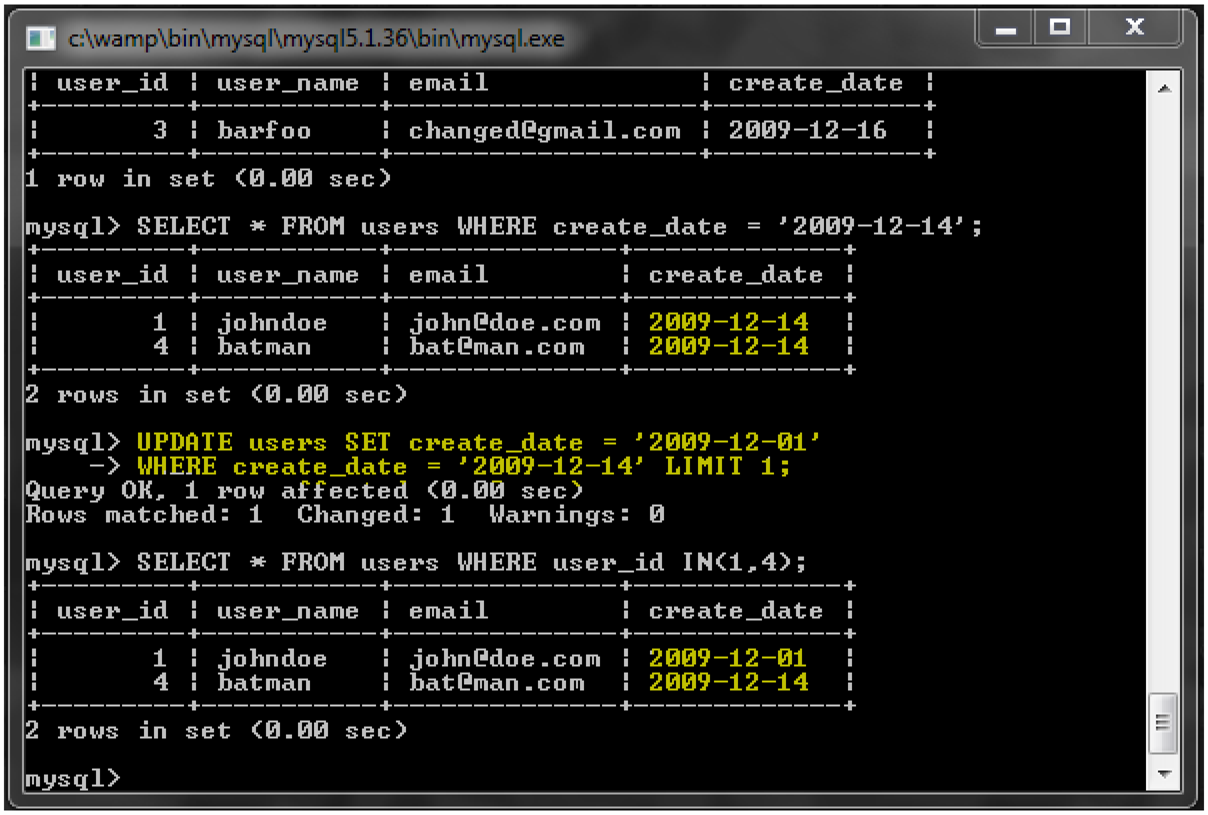
This query is used for updating the data in a table.



```
+---------+-----------+-------------+------------+
|       3 | foobar    | foo@bar.com | 2009-12-16 |
+---------+-----------+-------------+------------+
1 row in set (0.00 sec)

mysql> SELECT * FROM users
    -> LIMIT 1 OFFSET 2;
+---------+-----------+-------------+------------+
| user_id | user_name | email       | create_date |
+---------+-----------+-------------+------------+
|       3 | foobar    | foo@bar.com | 2009-12-16 |
+---------+-----------+-------------+------------+
1 row in set (0.00 sec)

mysql> UPDATE users SET
    ->  email             = 'changed@gmail.com',
    ->  user_name         = 'barfoo'
    -> WHERE user_name = 'foobar';
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> SELECT * FROM users WHERE user_name = 'barfoo';
+---------+-----------+-------------------+-------------+
| user_id | user_name | email             | create_date |
+---------+-----------+-------------------+-------------+
|       3 | barfoo    | changed@gmail.com | 2009-12-16  |
+---------+-----------+-------------------+-------------+
1 row in set (0.00 sec)

mysql>
```

Most of the time, it is used with a WHERE clause, because you would want only specific rows to be updated. If a WHERE clause is not provided, all rows would be updated with the same changes.

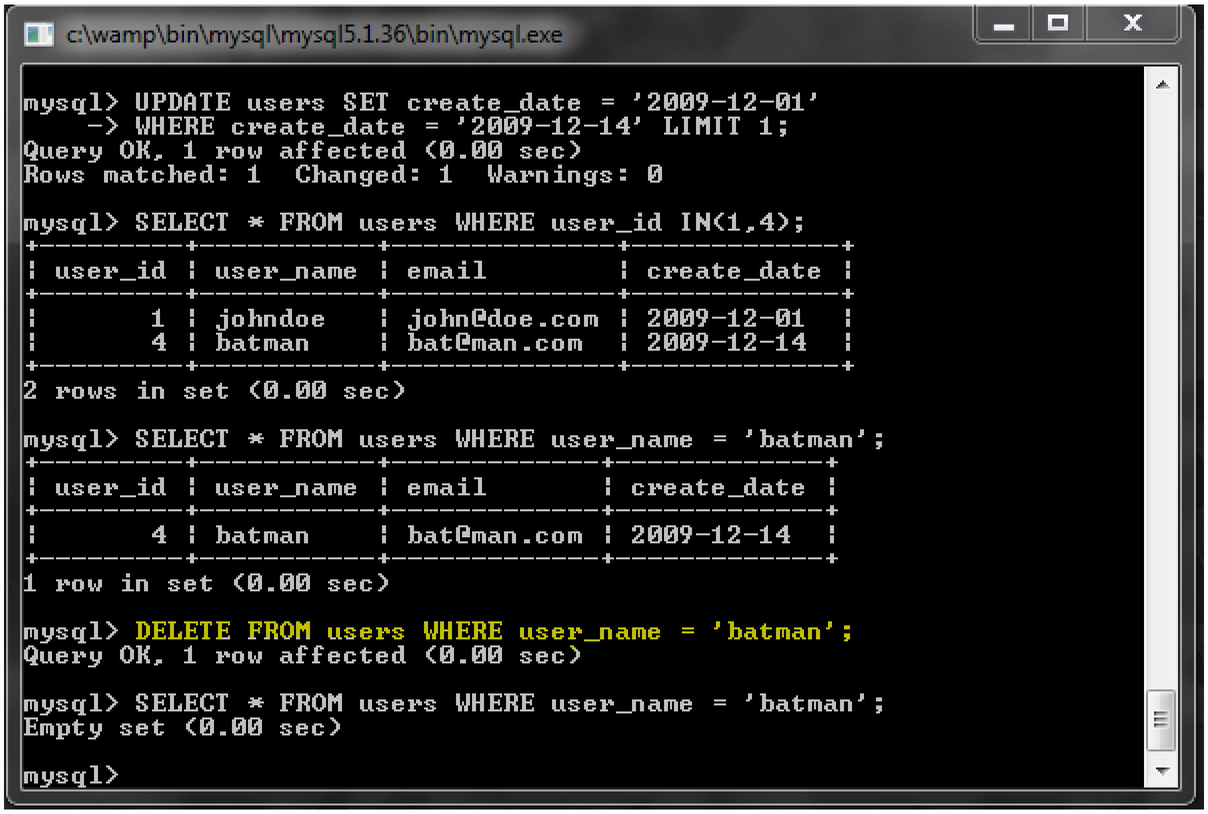
You can also use a LIMIT clause to limit the number of rows to be updated.



```
| user_id | user_name | email              | create_date |
+---------+-----------+--------------------+-------------+
|       3 | barfoo    | changed@gmail.com  | 2009-12-16  |
+---------+-----------+--------------------+-------------+
1 row in set (0.00 sec)

mysql> SELECT * FROM users WHERE create_date = '2009-12-14';
+---------+-----------+---------------+-------------+
| user_id | user_name | email         | create_date |
+---------+-----------+---------------+-------------+
|       1 | johndoe   | john@doe.com  | 2009-12-14  |
|       4 | batman    | bat@man.com   | 2009-12-14  |
+---------+-----------+---------------+-------------+
2 rows in set (0.00 sec)

mysql> UPDATE users SET create_date = '2009-12-01'
    -> WHERE create_date = '2009-12-14' LIMIT 1;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> SELECT * FROM users WHERE user_id IN(1,4);
+---------+-----------+---------------+-------------+
| user_id | user_name | email         | create_date |
+---------+-----------+---------------+-------------+
|       1 | johndoe   | john@doe.com  | 2009-12-01  |
|       4 | batman    | bat@man.com   | 2009-12-14  |
+---------+-----------+---------------+-------------+
2 rows in set (0.00 sec)

mysql>
```

DELETE: Delete Data from a Table

Just like UPDATE, this query is also usually used with a WHERE clause.



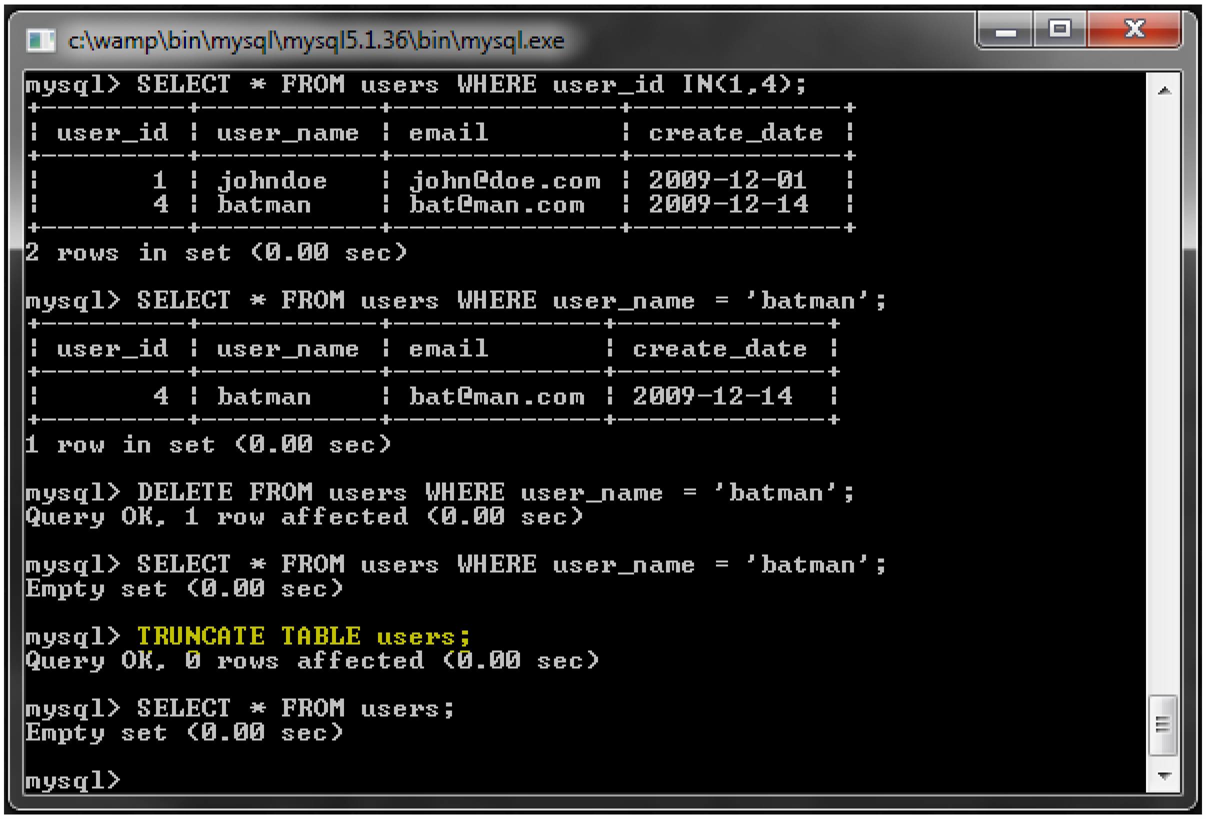
```
mysql> UPDATE users SET create_date = '2009-12-01'
    -> WHERE create_date = '2009-12-14' LIMIT 1;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> SELECT * FROM users WHERE user_id IN(1,4);
+---------+-----------+--------------+-------------+
| user_id | user_name | email        | create_date |
+---------+-----------+--------------+-------------+
|       1 | johndoe   | john@doe.com | 2009-12-01  |
|       4 | batman    | bat@man.com  | 2009-12-14  |
+---------+-----------+--------------+-------------+
2 rows in set (0.00 sec)

mysql> SELECT * FROM users WHERE user_name = 'batman';
+---------+-----------+-------------+-------------+
| user_id | user_name | email       | create_date |
+---------+-----------+-------------+-------------+
|       4 | batman    | bat@man.com | 2009-12-14  |
+---------+-----------+-------------+-------------+
1 row in set (0.00 sec)

mysql> DELETE FROM users WHERE user_name = 'batman';
Query OK, 1 row affected (0.00 sec)

mysql> SELECT * FROM users WHERE user_name = 'batman';
Empty set (0.00 sec)

mysql>
```

TRUNCATE TABLE

To delete the entire contents of a table, you can just do this:

```
DELETE FROM users;
```

But it is usually more performance efficient to use TRUNCATE instead.



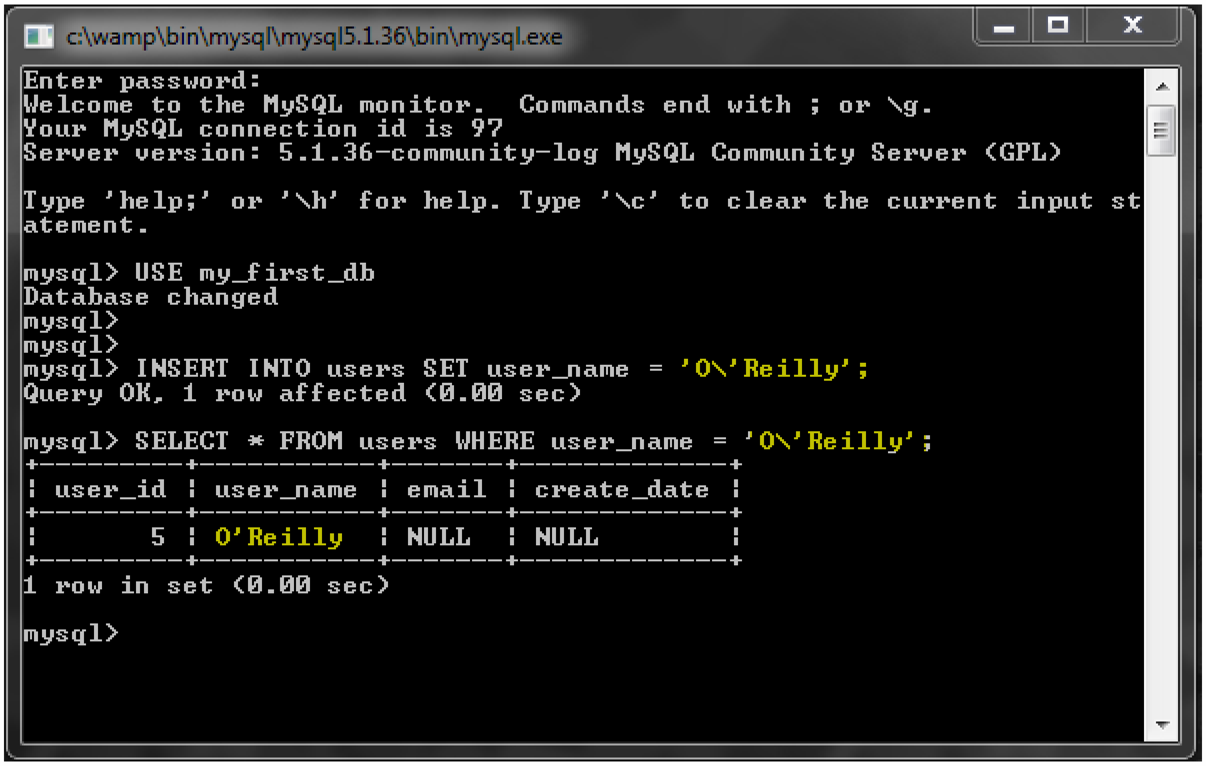
```
mysql> SELECT * FROM users WHERE user_id IN(1,4);
+---------+-----------+--------------+-------------+
| user_id | user_name | email        | create_date |
+---------+-----------+--------------+-------------+
|       1 | johndoe   | john@doe.com | 2009-12-01  |
|       4 | batman    | bat@man.com  | 2009-12-14  |
+---------+-----------+--------------+-------------+
2 rows in set (0.00 sec)

mysql> SELECT * FROM users WHERE user_name = 'batman';
+---------+-----------+-------------+-------------+
| user_id | user_name | email       | create_date |
+---------+-----------+-------------+-------------+
|       4 | batman    | bat@man.com | 2009-12-14  |
+---------+-----------+-------------+-------------+
1 row in set (0.00 sec)

mysql> DELETE FROM users WHERE user_name = 'batman';
Query OK, 1 row affected (0.00 sec)

mysql> SELECT * FROM users WHERE user_name = 'batman';
Empty set (0.00 sec)

mysql> TRUNCATE TABLE users;
Query OK, 0 rows affected (0.00 sec)

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

mysql>
```

TRUNCATE also resets the AUTO_INCREMENT numbers so a new row will again have the id 1. But this does not happen with a DELETE query, and the counter keeps going up.

Escaping String Values and Special Words

String Values

Certain characters need to be escaped, otherwise you can have problems.



```
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 97
Server version: 5.1.36-community-log MySQL Community Server (GPL)

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

mysql> USE my_first_db
Database changed
mysql>
mysql>
mysql> INSERT INTO users SET user_name = 'O\'Reilly';
Query OK, 1 row affected (0.00 sec)

mysql> SELECT * FROM users WHERE user_name = 'O\'Reilly';
+---------+-----------+-------+-------------+
| user_id | user_name | email | create_date |
+---------+-----------+-------+-------------+
|       5 | O'Reilly  | NULL  | NULL        |
+---------+-----------+-------+-------------+
1 row in set (0.00 sec)

mysql>
```

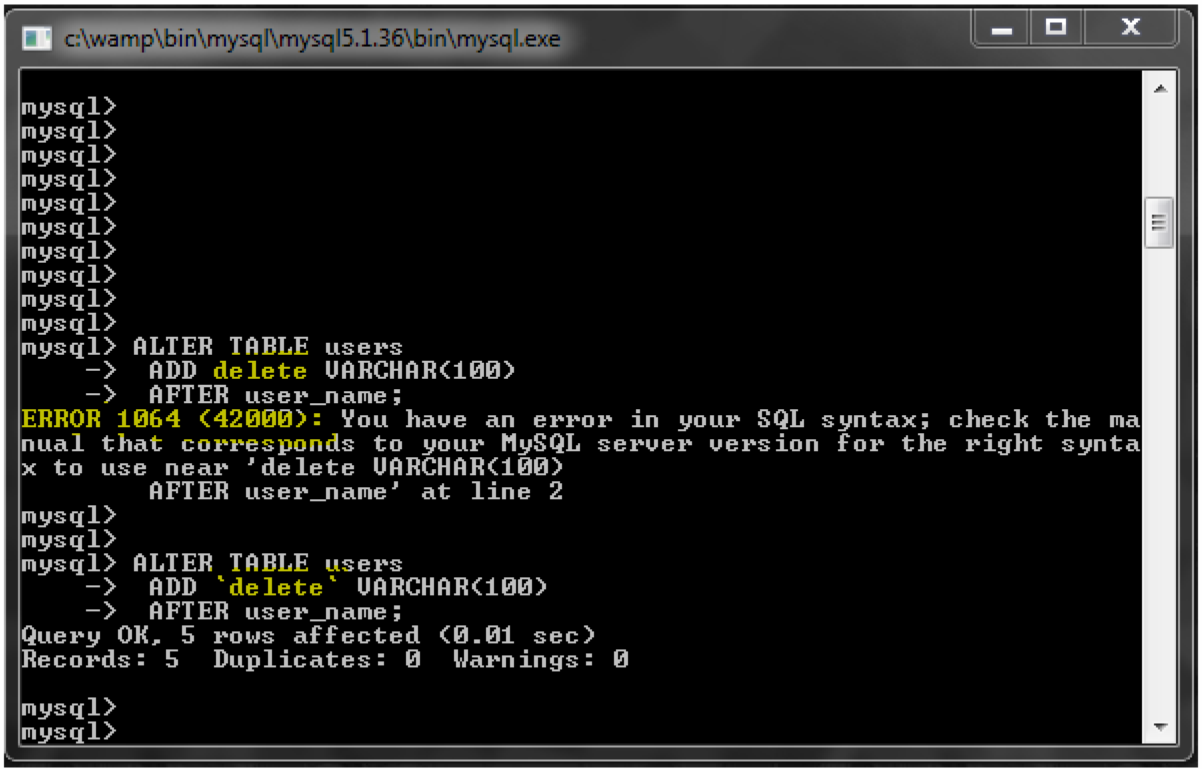
Backslash (\) is used for escaping.

This is also very important for security reasons. Any user input going into the database needs to be properly escaped. In PHP, you use the mysql_real_escape_string() function or use prepared statements since they do escaping automatically.

Special Words

Since MySQL has many special words like SELECT or UPDATE, you can prevent collision by putting quotes around your table and column names. But these are not the regular quotes; you need to use the backtick (`) character.

Let's say you want to add a column named 'delete' for some reason:



c:\wamp\bin\mysql\mysql5.1.36\bin\mysql.exe

```
mysql>
mysql>
mysql>
mysql>
mysql>
mysql>
mysql>
mysql>
mysql>
mysql>
mysql> ALTER TABLE users
    ->  ADD delete VARCHAR(100)
    ->  AFTER user_name;
ERROR 1064 (42000): You have an error in your SQL syntax; check the ma
nual that corresponds to your MySQL server version for the right synta
x to use near 'delete VARCHAR(100)
        AFTER user_name' at line 2
mysql>
mysql>
mysql> ALTER TABLE users
    ->  ADD `delete` VARCHAR(100)
    ->  AFTER user_name;
Query OK, 5 rows affected (0.01 sec)
Records: 5  Duplicates: 0  Warnings: 0

mysql>
mysql>
```