MySQL Database Tutorial

(Adapted from http://www.analysisandsolutions.com/code/mybasic.htm)

Start the Client

```
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1 to server version: 4.1.16-nt-max

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql -u root -p
```

Creating a Simple Database and Displaying Structure

Instruct MySQL to setup a new database

```
mysql 4.1.16-nt-max> create database database01;
Database "database01" created.
```

Open the database

```
mysql 4.1.16-nt-max> use database01
Database changed
```

Create a table

```
mysql 4.1.16-nt-max> create table table01 (field01 integer, field02
char(10));
Query OK, 0 rows affected (0.00 sec)
```

⚠ Enclose entire list of field names between one pair of parentheses.
⚠ Commas are used between each field.
ⓘ A space may be used after the comma between fields.
⚠ A comma is *not* used after last field.
⚠ This, and all SQL statements, are concluded by a semicolon ";".

List the tables

```
mysql 4.1.16-nt-max> show tables;
+----------------------+
| Tables in database01 |
+----------------------+
| table01              |
| table02              |
+----------------------+
```

List the fields in a table

```
mysql 4.1.16-nt-max> show columns from table01;
+---------+----------+------+-----+---------+-------+
| Field   | Type     | Null | Key | Default | Extra |
+---------+----------+------+-----+---------+-------+
| field01 | int(11)  | YES  |     |         |       |
| field02 | char(10) | YES  |     |         |       |
+---------+----------+------+-----+---------+-------+
```

Congratulations! Pretty straightforward, eh?

Putting Data into a Table

Insert a record

```
mysql 4.1.16-nt-max> insert into table01 (field01, field02) values
(1, 'first');
Query OK, 1 row affected (0.00 sec)
```

 Enclose entire list of field names between one pair of parentheses.
 Enclose the values to be inserted between *another* pair of parentheses.
 Commas are used between each field and between each value.
 A space may be used after the comma between fields.

List all the records in a table

```
mysql 4.1.16-nt-max> select * from table01;
+---------+---------+
| field01 | field02 |
+---------+---------+
|       1 | first   |
+---------+---------+
```

Adding Fields

...one field at a time

```
mysql 4.1.16-nt-max> alter table table01 add column field03
char(20);
Query OK, 1 row affected (0.04 sec)
Records: 1  Duplicates: 0  Warnings: 0
```

...more than one at a time

```
mysql 4.1.16-nt-max> alter table table01 add column field04 date,
add column field05 time;
Query OK, 1 row affected (0.04 sec)
Records: 1  Duplicates: 0  Warnings: 0
```

⚠The "add column" must be restated for each column.
⚠Commas are used between each add column statement.
ⓘA space may be used after these commas.

ⓘThe MySQL Manual fully explains each possible column data type.

Did it work?

```
mysql 4.1.16-nt-max> select * from table01;
+---------+---------+---------+---------+---------+
| field01 | field02 | field03 | field04 | field05 |
+---------+---------+---------+---------+---------+
|       1 | first   | NULL    | NULL    | NULL    |
+---------+---------+---------+---------+---------+
```

Now we're getting somewhere!

Multi-line Command Entry

The MySQL command line interface allows you to put a statement on one line or spread it across multiple lines. There's no difference in syntax between the two. Using multiple lines allows you to break down the SQL statement into steps you may more easily comprehend.

In multiple line mode, the interpreter appends each line to the prior lines. This continues until you enter a semicolon ";" to close out the SQL statement. Once the semicolon is typed in and you hit enter, the statement is executed.

Here's an example of the same exact SQL statement entered both ways:

Single Line Entry

```
mysql 4.1.16-nt-max> create table table33 (field01 integer,field02
char(30));
```

Multiple Line Entry

```
mysql 4.1.16-nt-max> create table table33
-> (field01
-> integer,
```

```
-> field02
-> char(30));
```

⚠Don't break up words:

Valid	Invalid

`mysql 4.1.16-nt-max> create table table33`
`-> (field01`
`-> integer,`
`-> field02`
`-> char(30));`
	`mysql 4.1.16-nt-max> create table table33`
	`-> (field01 inte`
	`-> ger,`
	`-> field02`
	`-> char(30));`

⚠When inserting or updating records, do not spread a field's string across multiple lines, otherwise the line breaks are stored in the record:

Standard Operation	Line Break Stored in Record

`mysql 4.1.16-nt-max> insert into table33 (field02)`
`-> values`
`-> ('Who thought of foo?');`
	`mysql 4.1.16-nt-max> insert into table33 (field02)`
	`-> values`
	`-> ('Pooh thought`
	`-> of foo.');`

Results

```
mysql 4.1.16-nt-max> select * from table33;
+---------+---------------------+
| field01 | field02             |
+---------+---------------------+
|    NULL | Who thought of foo? |
|    NULL | Pooh thought
of foo. |
+---------+---------------------+
```

Insert Some More Records into the Table

Add this record

```
mysql 4.1.16-nt-max> insert into table01
(field01,field02,field03,field04,field05) values
-> (2, 'second', 'another', '1999-10-23', '10:30:00');
Query OK, 1 row affected (0.00 sec)
```

Quotes must go around text values.

Standard date format is "yyyy-mm-dd".
Standard time format is "hh:mm:ss".
Quotes are required around the standard date and time formats, noted above.
Dates may also be entered as "yyyymmdd" and times as "hhmmss". If entered in this format, values don't need to be quoted.

Numeric values do *not* need to be quoted. This holds true regardless of the data type a column is formatted to contain (e.g. text, date, time, integer).

MySQL has a useful command buffer. The buffer stores the SQL statements you've entered thus far. Using it keeps you from having to retype the same commands over and over. Let's use this next step as an example.

Add another record using the command buffer (and optional date and time formats)

1. Hit the up arrow key twice.
2. Hit the ENTER key.
3. Type in the new values between a pair parentheses and stick a closing semicolon on the end.
   ```
   (3, 'a third', 'more foo for you', 19991024, 103004);
   ```
4. Hit the ENTER key.

Voilà!

Is it in there?

```
mysql 4.1.16-nt-max> select * from table01;
+---------+----------+------------------+------------+----------+
| field01 | field02  | field03          | field04    | field05  |
+---------+----------+------------------+------------+----------+
|       1 | first    | NULL             | NULL       | NULL     |
|       2 | second   | another          | 1999-10-23 | 10:30:00 |
|       3 | a third  | more foo for you | 1999-10-24 | 10:30:01 |
+---------+----------+------------------+------------+----------+
```

It's in there!

Now, we're almost done...

Updating Existing Records

Modify one field at a time

⚠Again, be careful with syntax. Quote marks need to go around text but not around numbers.

```
mysql 4.1.16-nt-max> update table01 set field03='new info' where
field01=1;
Query OK, 1 row affected (0.00 sec)
```

Deleting Records

The delete command

```
mysql 4.1.16-nt-max> delete from table01 where field01=3;
Query OK, 1 row affected (0.01 sec)
mysql 4.1.16-nt-max> select * from table01;
+---------+---------+----------+------------+----------+
| field01 | field02 | field03  | field04    | field05  |
+---------+---------+----------+------------+----------+
|       1 | first   | new info | 1999-10-22 | 15:29:01 |
|       2 | second  | another  | 1999-10-23 | 15:29:01 |
+---------+---------+----------+------------+----------+
```

Time to Call it Quits

```
mysql 4.1.16-nt-max> quit
Bye
```