Startup MySQL

```
[randy@jupiter ~]$ mysql

Welcome to the MySQL monitor.  Commands end with ; or \g.

Your MySQL connection id is 20 to server version: 4.1.11-Max

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

mysql>

Common command line switches to mysql client:

-u = specifies user name to connect as.

-h = specifies a remote MySQL server to connect to.

-p = prompt for a password.
```

Accessing a database

```
mysql> show databases;
+------------+
| Database   |
+------------+
| cim7       |
| egroupware |
| landesk    |
| mpn2       |
| mysql      |
| test       |
+------------+
6 rows in set (0.05 sec)

     mysql> use cim7;
     Database changed
```

Create a database and table

```
mysql> create database database01;

Query OK, 1 row affected (0.05 sec)

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

List the fields in a table

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

Congratulations! That wasn't all that hard, was it?

Putting Data into a Table

Inserting a record

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

On to the next task!...

Adding Fields

...one field at a time

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

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> create table table33 (field01 integer,field02 char(30));
```

Multiple Line Entry

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

⚠ Don't break up words:

Valid	Invalid

`mysql> create table table33`
`-> (field01`
`-> integer,`
`-> field02`
`-> char(30));`
	`mysql> 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> insert into table33 (field02)`
`-> values`
`-> ('Who thought of foo?');`
	`mysql> insert into table33 (field02)`
	`-> values`
	`-> ('Pooh thought`
	`-> of foo.');`

Results

```
mysql> 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> 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.

Is it in there?

```
mysql> 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> update table01 set field03='new info' where field01=1;

Query OK, 1 row affected (0.00 sec)
```

Change multiple fields at once

⚠Remember to put commas between each field you're updating.

```
mysql> update table01 set field04=19991022, field05=062218 where field01=1;
Query OK, 1 row affected (0.00 sec)
```

So, what's up with our data?

```
mysql> select * from table01;
+---------+-----------+------------------+------------+----------+
| field01 | field02   | field03          | field04    | field05  |
+---------+-----------+------------------+------------+----------+
|       1 | first     | new info         | 1999-10-22 | 06:22:18 |
|       2 | second    | another          | 1999-10-23 | 10:30:00 |
|       3 | third one | more foo for you | 1999-10-24 | 10:30:01 |
+---------+-----------+------------------+------------+----------+
```

Update multiple records in one stroke

```
mysql> update table01 set field05=152901 where field04>19990101;
Query OK, 3 rows affected (0.00 sec)
```

So it now says...

```
mysql> 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 |
|       3 | third one | more foo for you | 1999-10-24 | 15:29:01 |
+---------+-----------+------------------+------------+----------+
```

Deleting Records

The delete command

```
mysql> delete from table01 where field01=3;
Query OK, 1 row affected (0.01 sec)
```

```
mysql> 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> quit
Bye
```

Security Basics

By default, the database user `root` has full control and NO PASSWORD!

To use SET PASSWORD on Linux, do this:

```
shell> mysql -u root
mysql> SET PASSWORD FOR ''@'localhost' = PASSWORD('newpwd');
mysql> SET PASSWORD FOR ''@'host_name' = PASSWORD('newpwd');
```

In the second SET PASSWORD statement, replace *host_name* with the name of the server host. This is the name that is specified in the Host column of the non-localhost record for root in the user table. If you don't know what hostname this is, issue the following statement before using SET PASSWORD:

```
mysql> SELECT Host, User FROM mysql.user;
```

Look for the record that has root in the User column and something other than localhost in the Host column. Then use that Host value in the second SET PASSWORD statement.

The other way to assign passwords to the anonymous accounts is by using UPDATE to modify the user table directly. Connect to the server as root and issue an UPDATE statement that assigns a value to the Password column of the appropriate user table records. The procedure is the same for Windows and Unix. The following UPDATE statement assigns a password to both anonymous accounts at once:

```
shell> mysql -u root
mysql> UPDATE mysql.user SET Password = PASSWORD('newpwd')
    ->     WHERE User = '';
mysql> FLUSH PRIVILEGES;
```

After you update the passwords in the user table directly using UPDATE, you must tell the server to re-read the grant tables with FLUSH PRIVILEGES. Otherwise, the change goes unnoticed until you restart the server.

If you prefer to remove the anonymous accounts instead, do so as follows:

```
shell> mysql -u root
mysql> DELETE FROM mysql.user WHERE User = '';
mysql> FLUSH PRIVILEGES;
```

To assign passwords using SET PASSWORD, connect to the server as root and issue two SET PASSWORD statements. Be sure to encrypt the password using the PASSWORD() function.

```
shell> mysql -u root

mysql> SET PASSWORD FOR 'root'@'localhost' = PASSWORD('newpwd');
mysql> SET PASSWORD FOR 'root'@'host_name' = PASSWORD('newpwd');
```

In the second SET PASSWORD statement, replace *host_name* with the name of the server host. This is the same hostname that you used when you assigned the anonymous account passwords.

To assign passwords to the root accounts using **mysqladmin**, execute the following commands:

```
shell> mysqladmin -u root password "newpwd"
shell> mysqladmin -u root -h host_name password "newpwd"
```

After the passwords have been set, you must supply the appropriate password whenever you connect to the server. For example, if you want to use **mysqladmin** to shut down the server, you can do so using this command:

```
shell> mysqladmin -u root -p shutdown
Enter password: (enter root password here)
```

References

Documentation: http://dev.mysql.com/doc/

Downloads: http://dev.mysql.com/downloads/

Primary Website: http://www.mysql.com

ODBC Driver: http://dev.mysql.com/downloads/connector/odbc/3.51.html

For a more in-depth introduction to MySQL Basics, download the MySQL Administrators Guide and go through Chapter 3.

Another Online Tutorial: http://php.about.com/od/mysqlbasics/ss/mysql101_2.htm