Getting Started

In this section, you'll learn how to:

- **Install MySQL on a Linux computer**
- **Start MySQL**
- **Create a new database**
- **Create a table**
- **Create a record**
- **Run a query**

What is MySQL?

MySQL is the world's most popular open-source database program.

MySQL is more like Microsoft SQL Server (a server-based database program) than Access (mainly for desktop users). With MySQL running on a server, you can easily use it for business systems or database-driven websites.

Easy to use and configure, MySQL is also capable of industrial-strength applications. Depending on the computer it's installed on, MySQL can hold several terabytes of information per table.

Install MySQL on a Linux computer

During Linux installation

1. **Obtain a copy of Linux.**

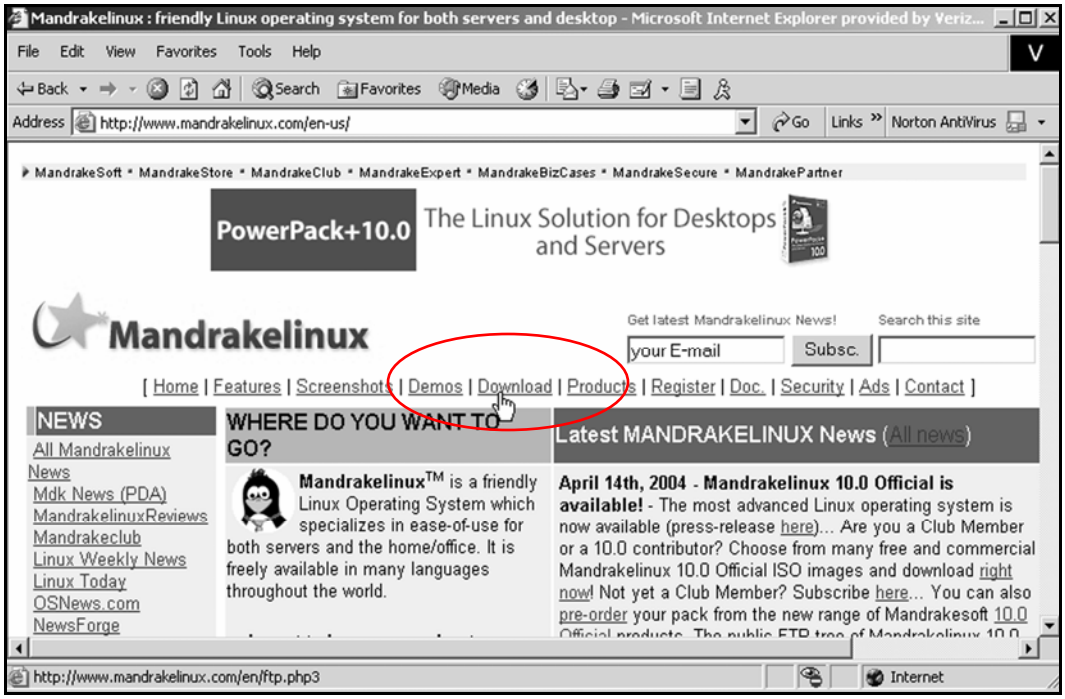
   Tip: *A good version of Linux to use with this book is Mandrake Linux.*

   You can buy a copy on CDs at:

   www.mandrakestore.com

   You can also download a copy at:

   www.mandrakelinux.com

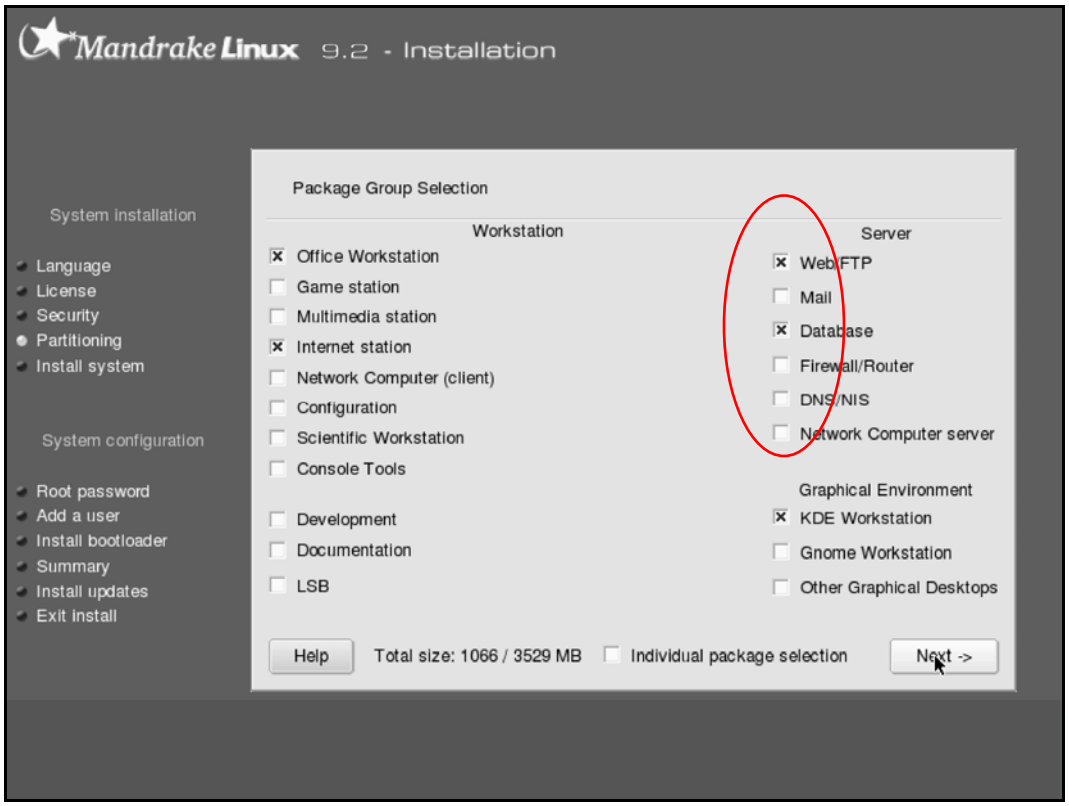


   If you download a copy, burn the three ISO files onto CDs.

2. **Begin installing Mandrake Linux.**

3. **When the Package Selection screen appears, make sure the Web/FTP and Database options are selected:**

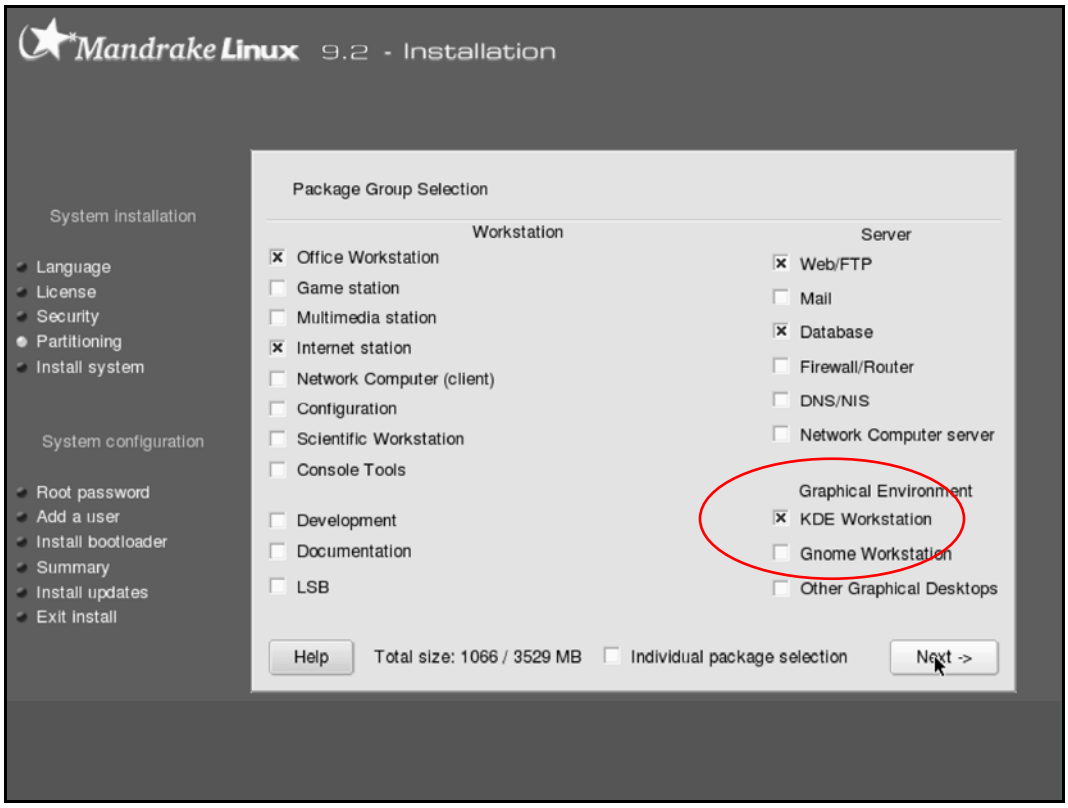
Server
Web/FTP
Database



This will install MySQL on your computer, along with a copy of the Apache Web server.

4. Under **Graphical Environment**, make sure **KDE Workstation** is selected:

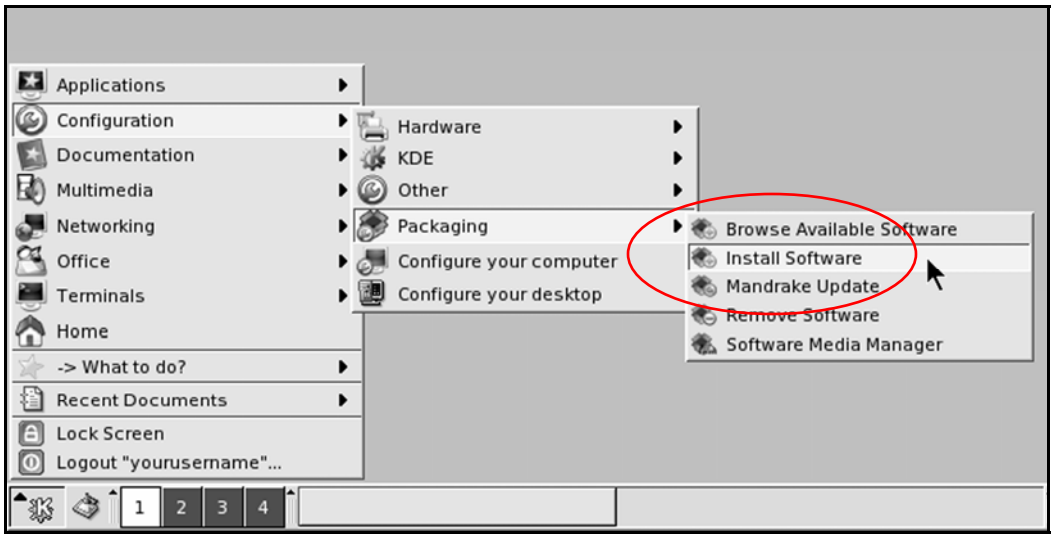
Graphical Environment
KDE Workstation



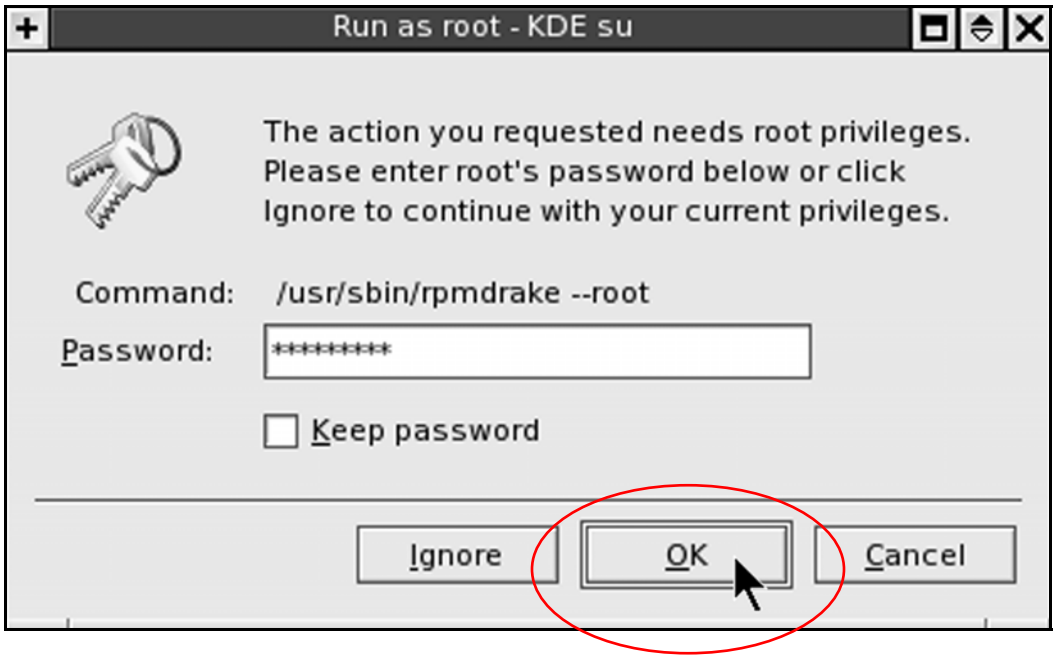
5. Continue with the Mandrake Linux installation.

On an existing Linux computer

1. Click the  icon, then **Configuration**, then **Packaging**, then **Install Software**.



2. When the **Run as Root** window appears, type the Root password in the **Password** box.

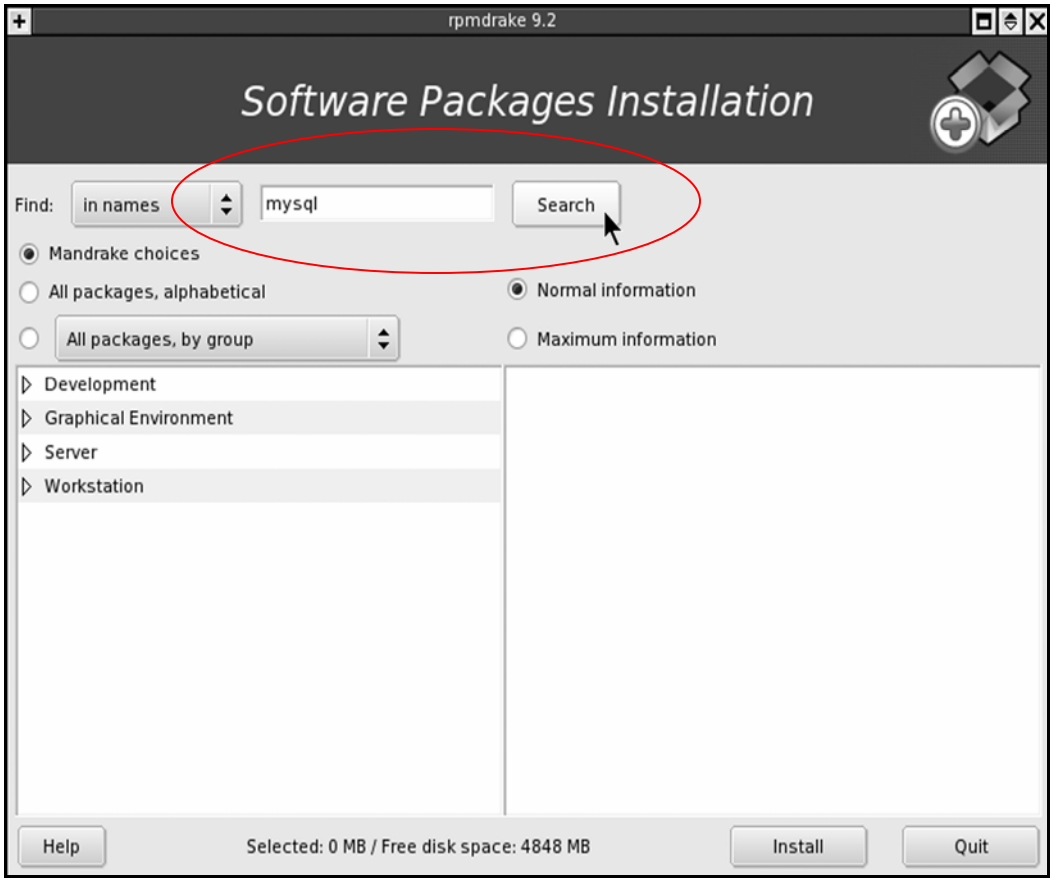


3. When the **Software Packages Installation** window appears, type:

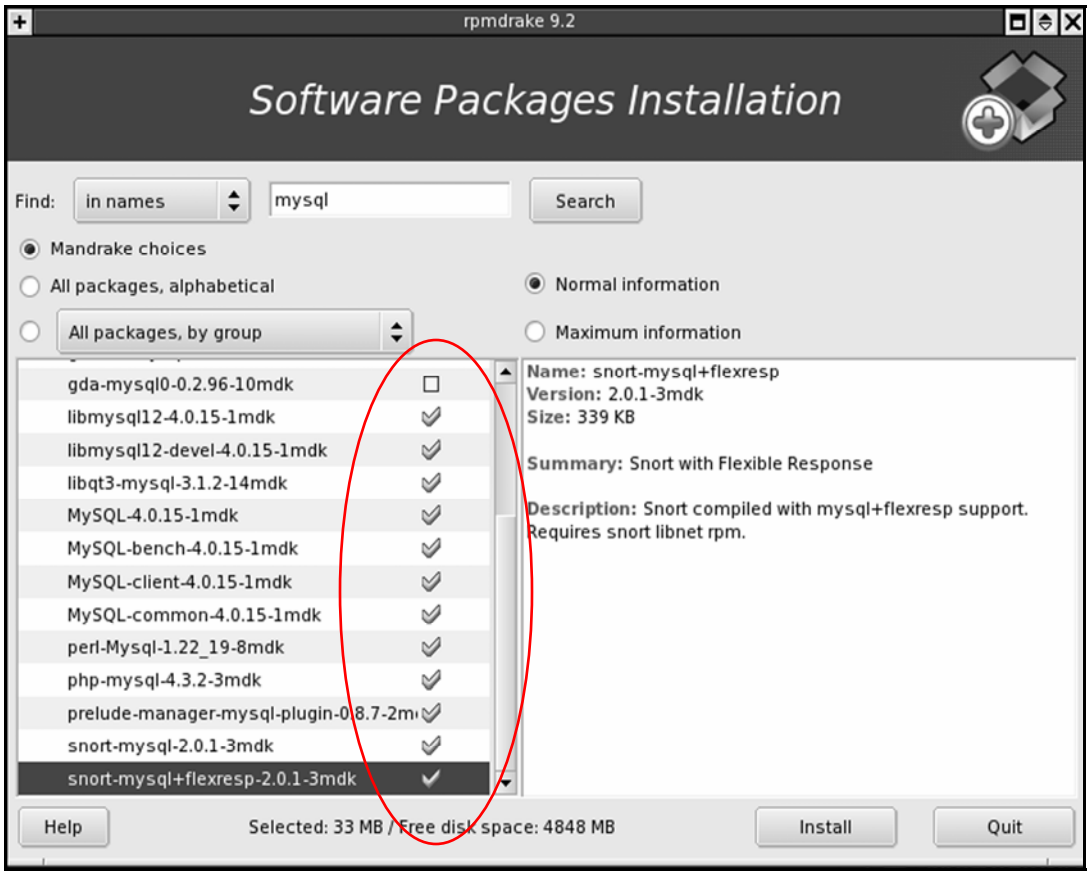
mysql

in the Search box.

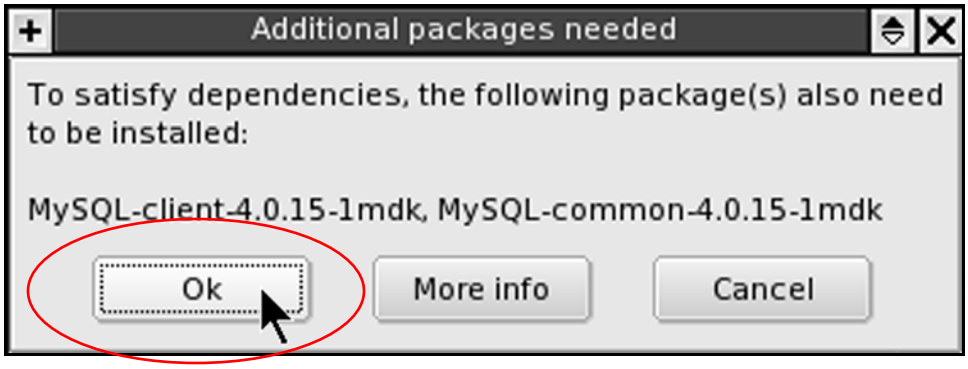
Then click the Search button.



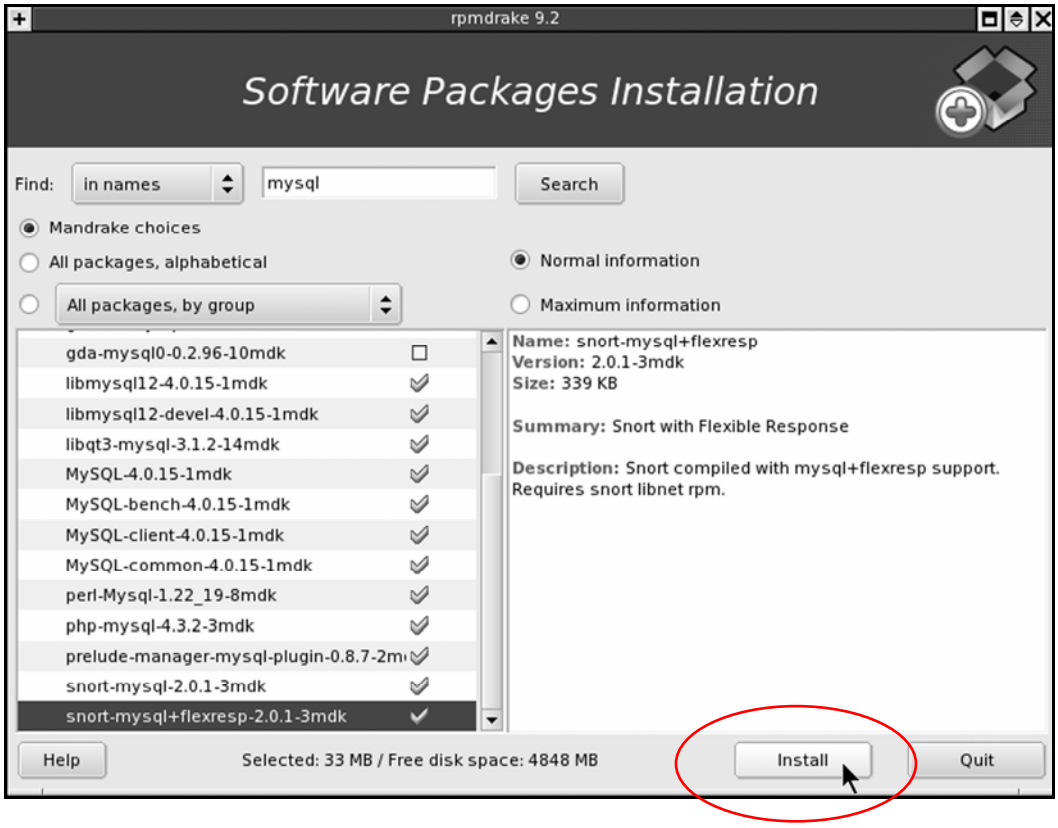
4. In the **Packages** list, check **MySQL-4.0.15-1mdk** and everything beneath it.



5. Whenever the **Additional packages needed** window appears, click the Ok button.



6. Click the Install button.

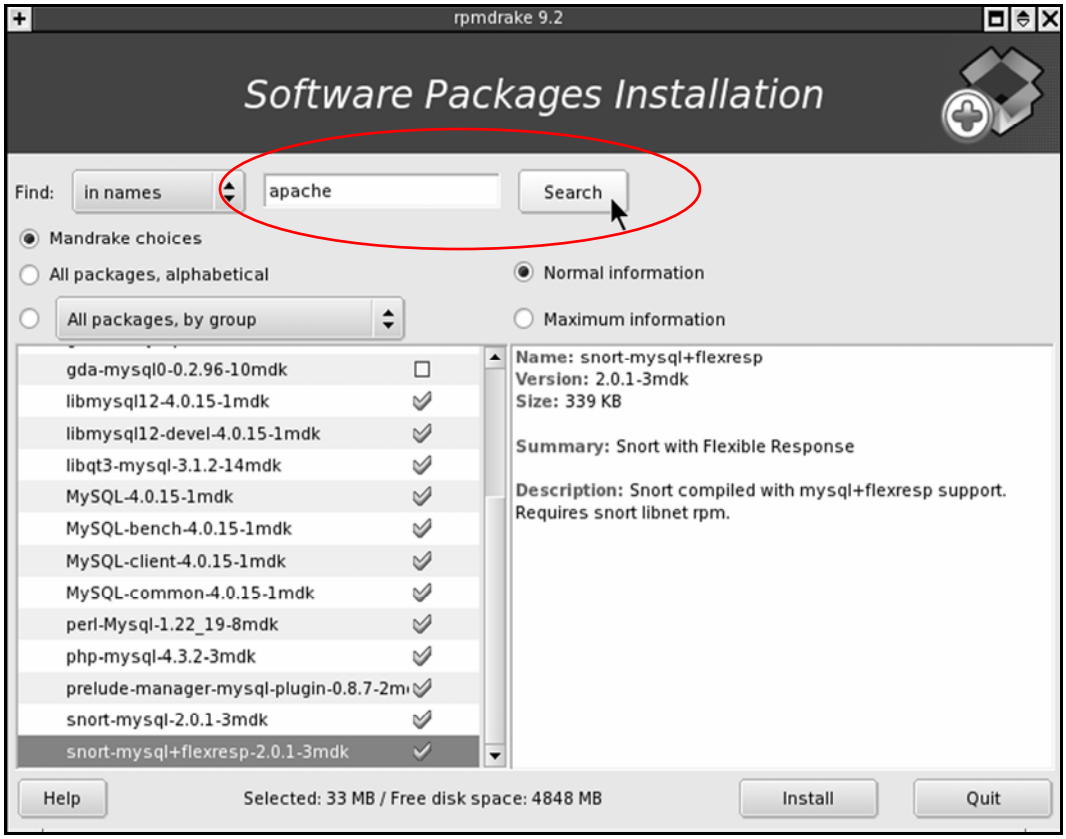


7. Insert the Linux CDs as directed.

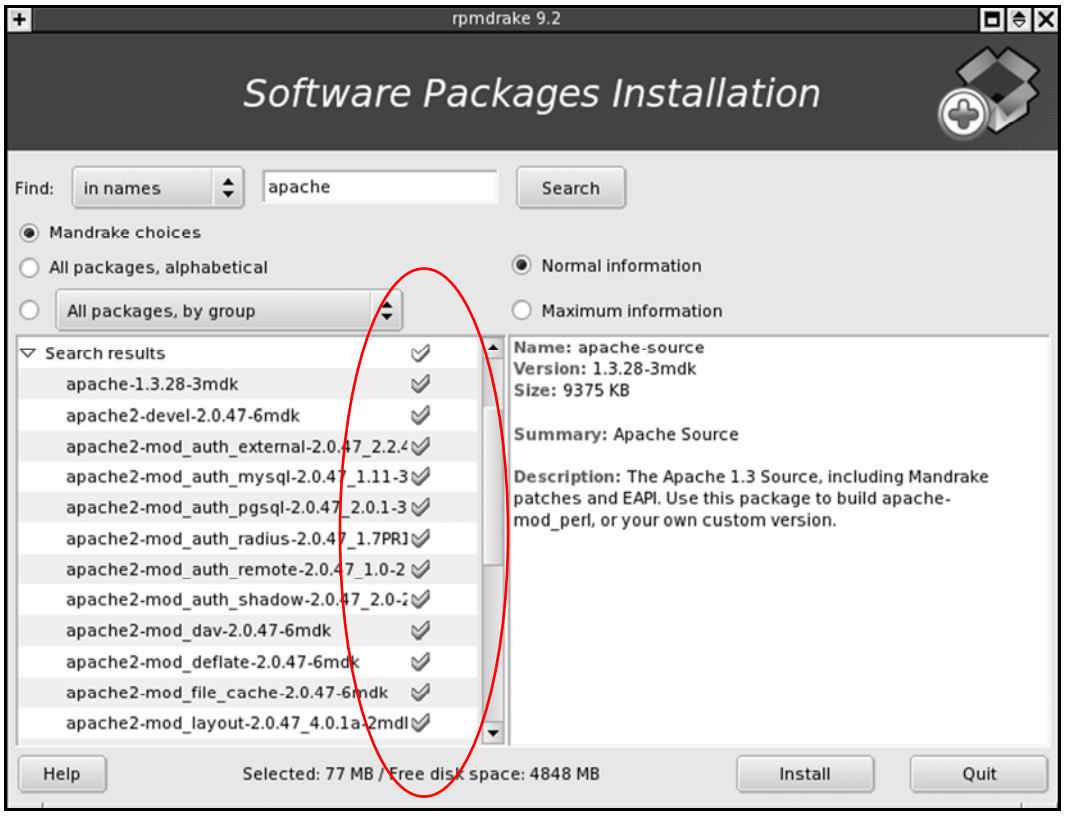
8. In the Search box, type:

apache

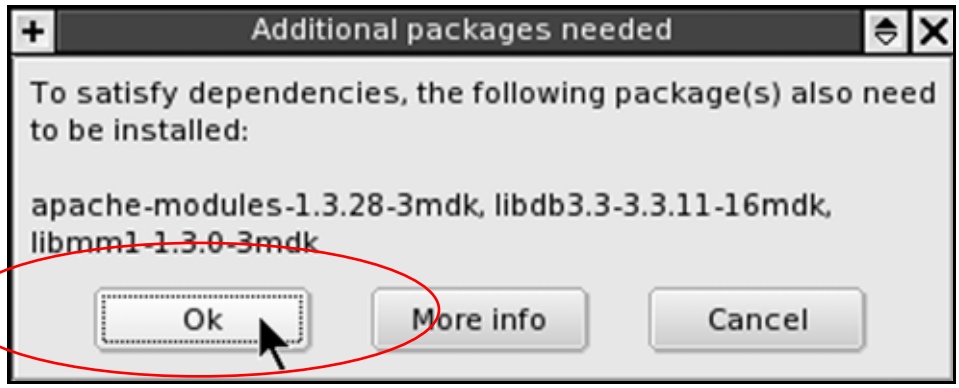
and click the Search button.



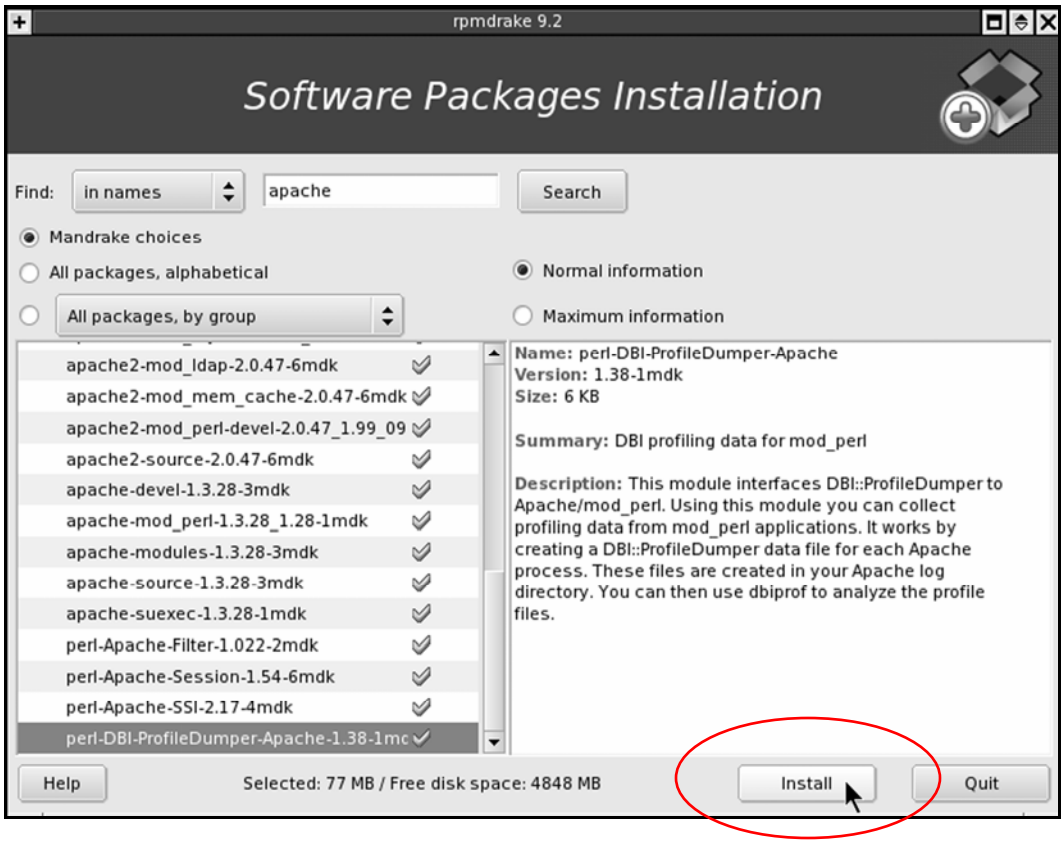
9. In the Packages list, check **apache-1.3.28-3mdk** and everything beneath it.



10. Whenever the **Additional packages needed** window appears, click the Ok button.

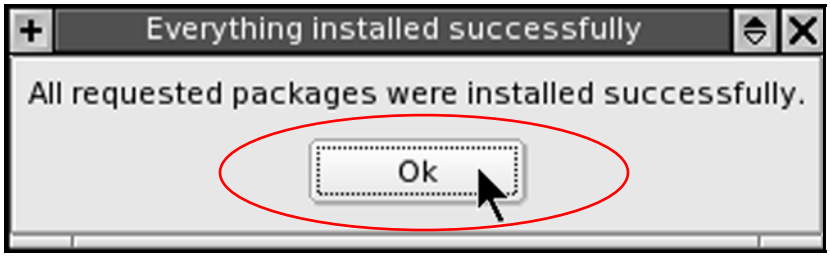


11. Click the Install button.

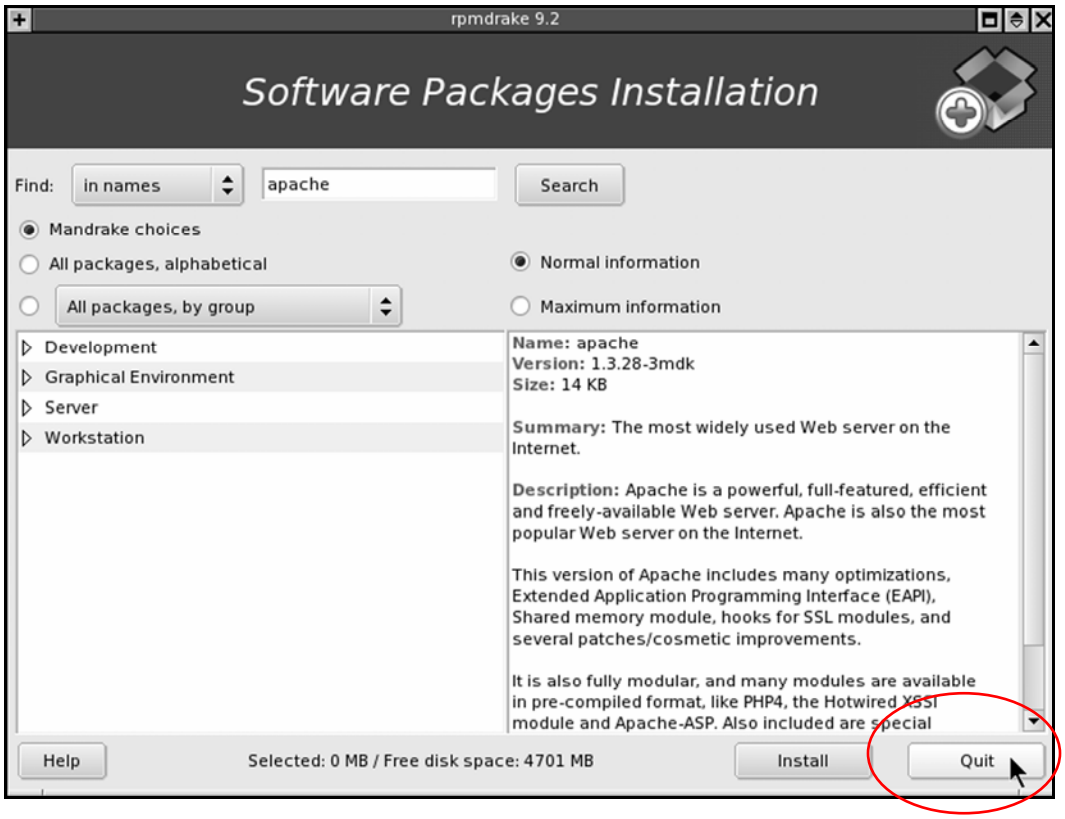


12. Insert the Linux CDs as directed.

13. When the installation is complete, click the Ok button.



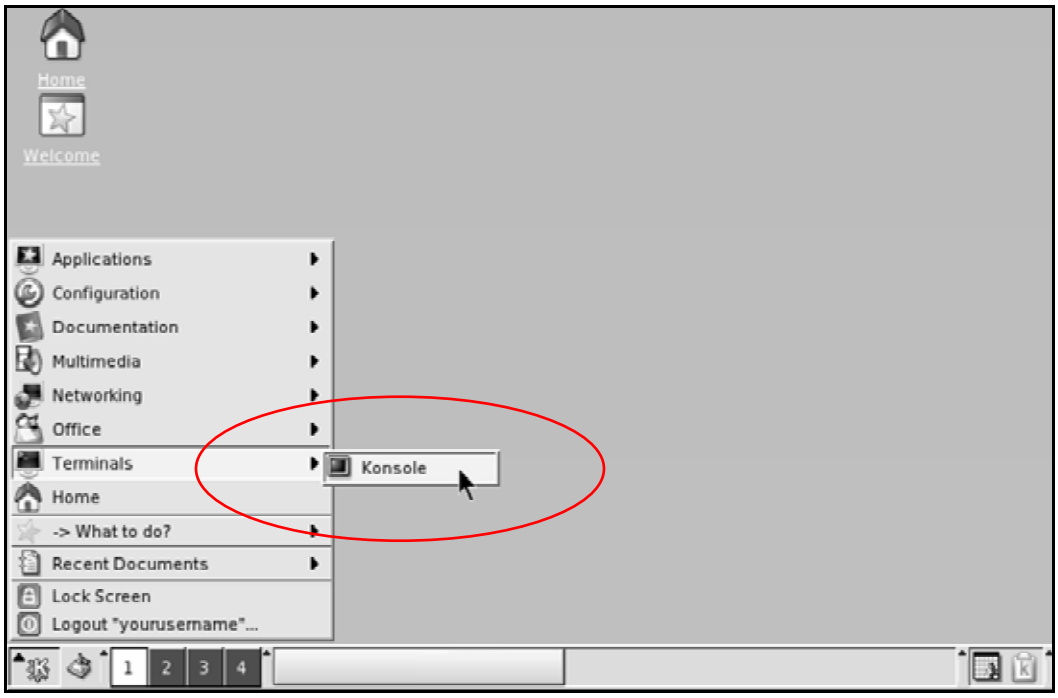
14. Click the Quit button.



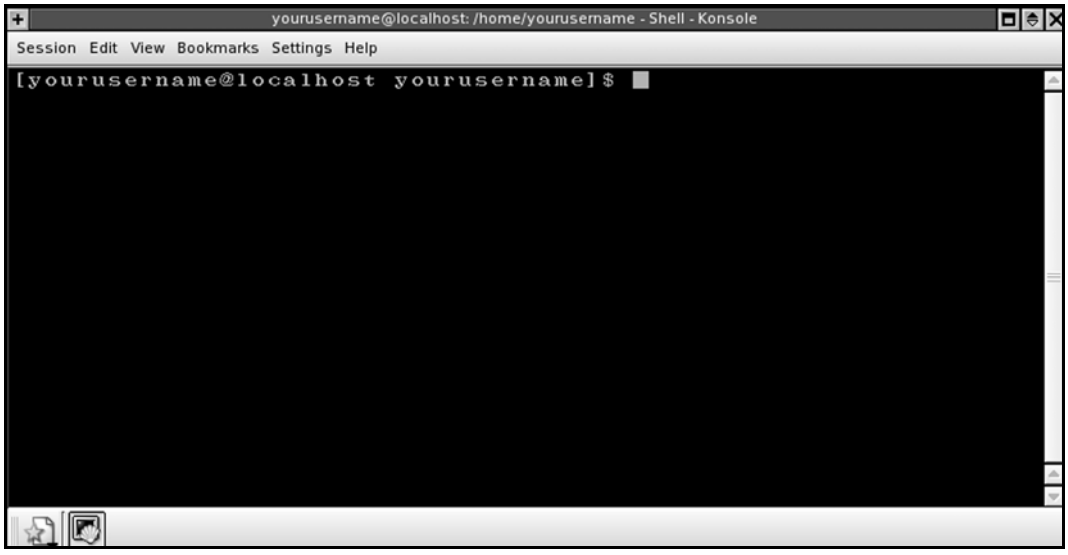
Start MySQL

1. Log in to your Linux computer using your user account.

2. Click the  icon, then **Terminals**, then **Konsole**.

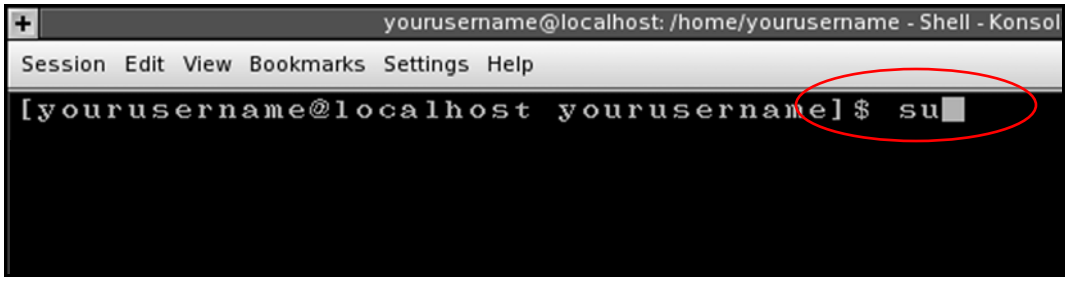


3. When the **Konsole** window opens, it should look like this:



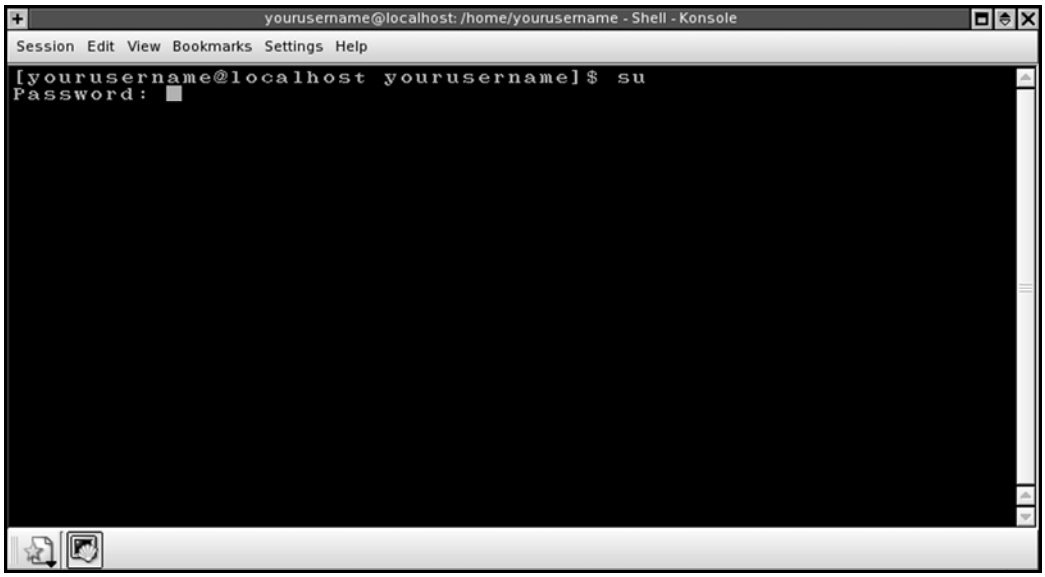
3. At the prompt, type:

```
su
```



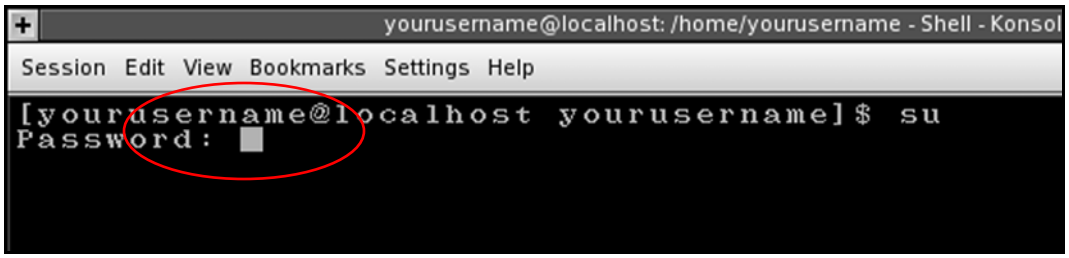
then press the **ENTER** key on your keyboard.

The window should look like this:



4. At the `Password` prompt, type:

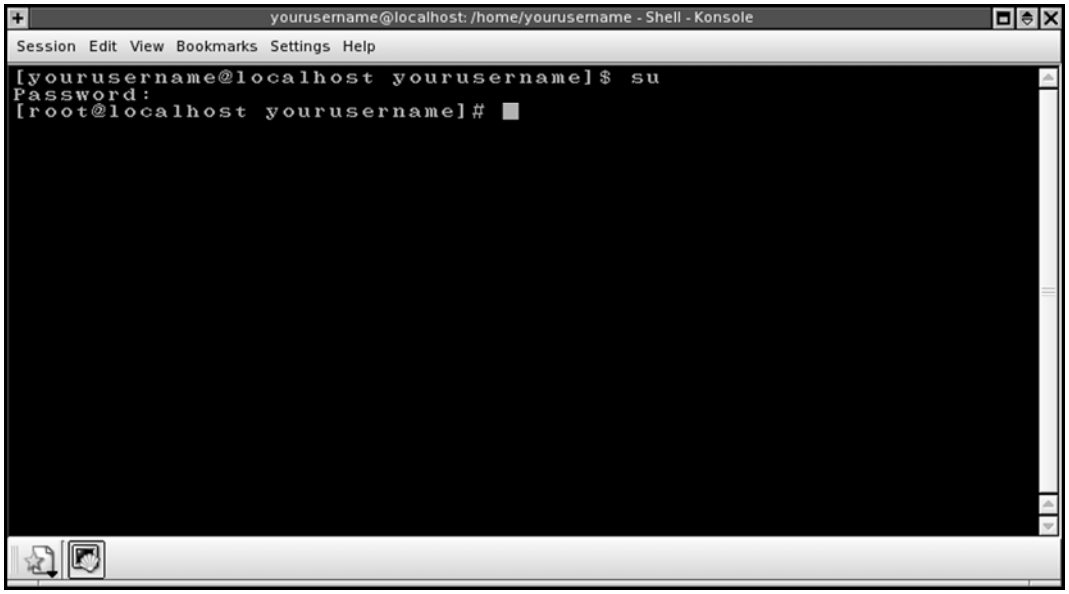
`Your Root user password`



Not this particular string, of course, but the actual Root password for the Linux computer.

Then press the **ENTER** key.

The window should look like this:

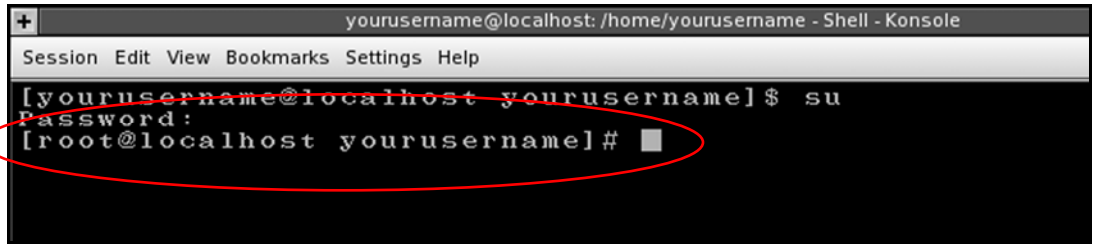


Tip: Notice the prompt has changed from

[yourusername@localhost yourusername]$

to

[yourusername@localhost yourusername]#



There's now a **#** at the end of the prompt.

$ **means you're giving Linux commands as a regular user. # means you're giving commands as the Root user.**

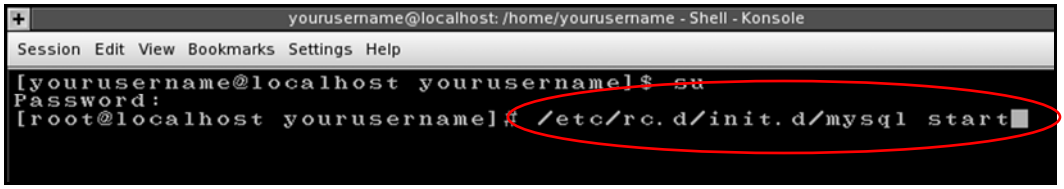
On any Linux computer, there are regular users and the Root user. Giving the *su* command allows you to give commands as the "Super User," or Root user, of the computer.

As the Root user, you can add/delete/modify any file on the computer. A regular user can't do this.

The Root user has the power to really mess up the computer, so you should only work as the Root user when necessary.

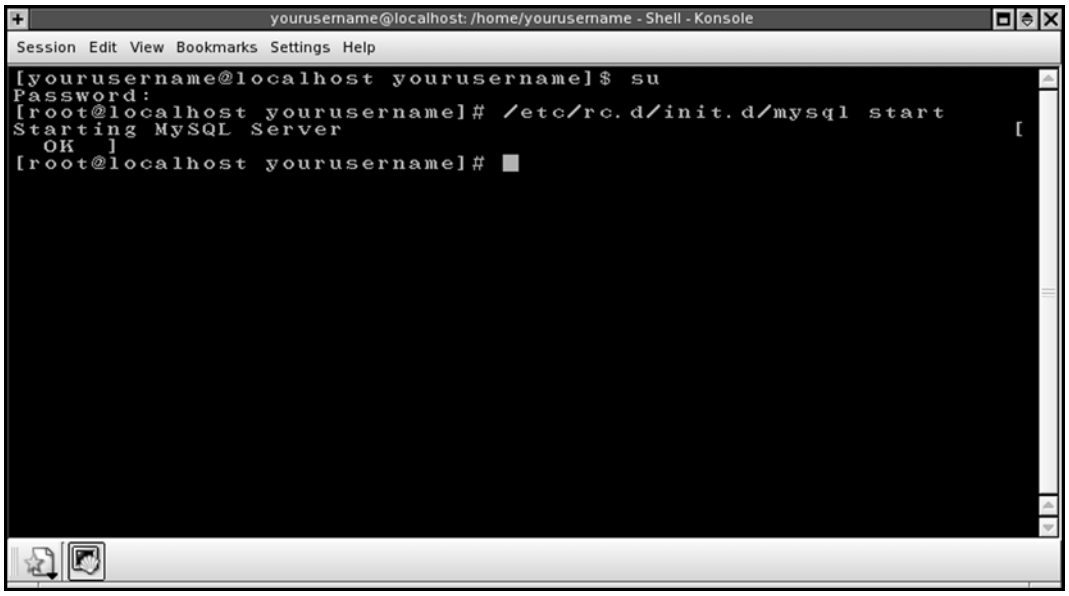
5. Next, type:

```
/etc/rc.d/init.d/mysql start
```



then press **ENTER**.

The window should look like this:

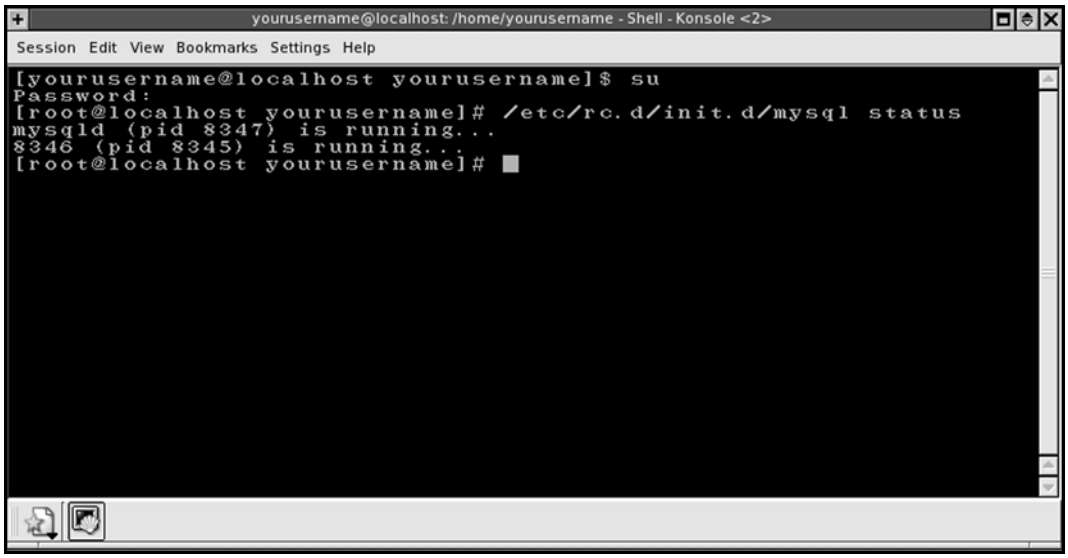


This starts the MySQL server—the program `mysql` in the `/etc/rc.d/init.d/` directory.

Tip: *If you are not sure whether or not the MySQL Server is running, type:*

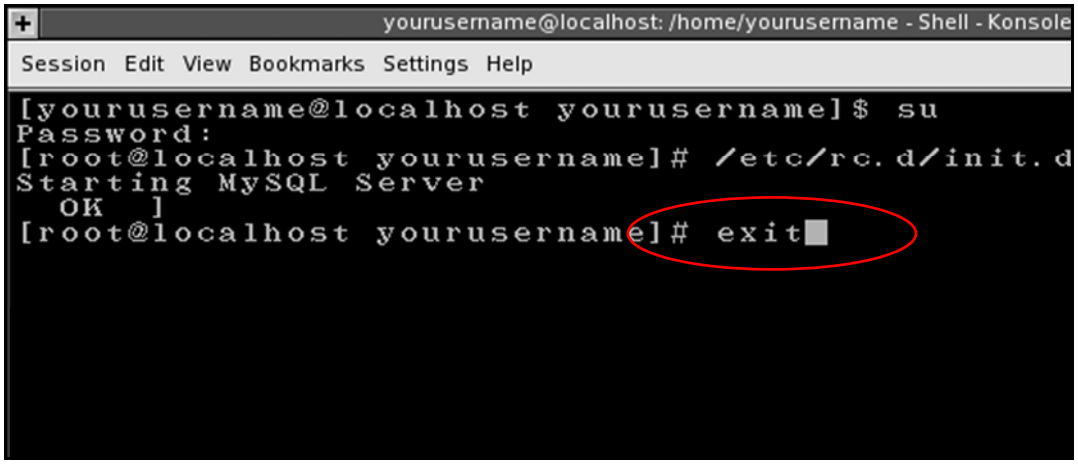
```
/etc/rc.d/init.d/mysql status
```

If it's running, the window will look like this:



6. Type:

```
exit
```



then press **ENTER**.

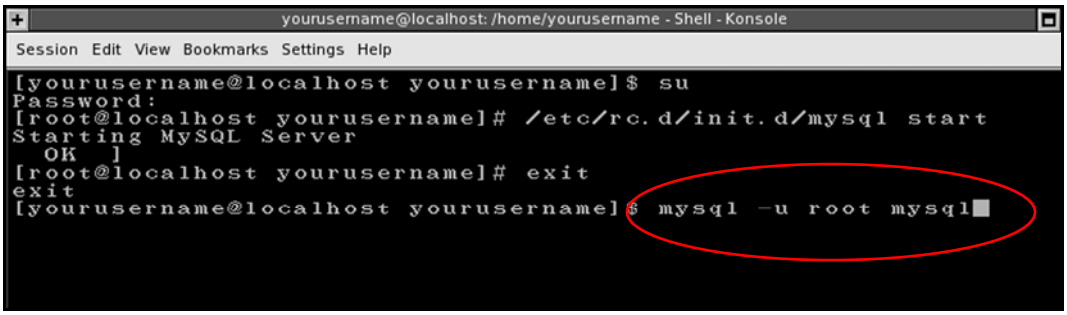
The prompt has now changed to:

```
[yourusername@localhost yourusername]$
```

Linux Root privileges were only needed to start MySQL, so you've logged out as the Linux computer's Super (Root) User.

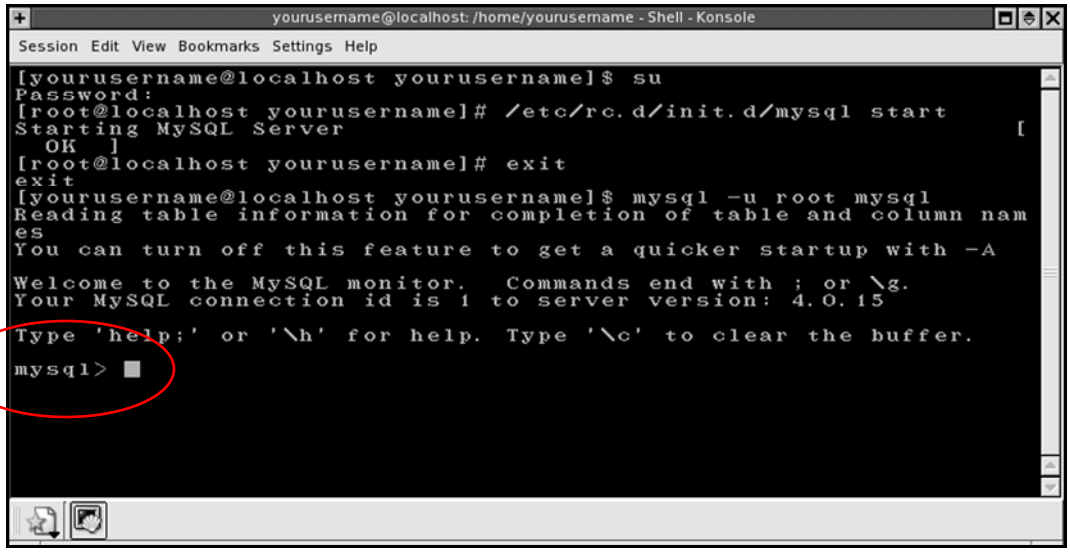
7. At the `[yourusername@localhost yourusername]$` prompt, type:

```
mysql -u root mysql
```



then press **ENTER**.

The window should look like this, with a `mysql>` prompt:



Here's what this string of commands means:

- **`mysql`**

  `mysql -u root mysql`

  This first `mysql` starts the MySQL client.

  MySQL is made up of two parts: the MySQL server program and a MySQL client program.

  The MySQL server program handles the storage of the data.

  The MySQL client program allows you to give commands to the MySQL server.

  You need both parts to make MySQL work.

- **`-u root`**

  `mysql` **`-u root`** `mysql`

  The `-u` command tells the MySQL client that you want to log into the MySQL server as a particular user. `root` denotes the root user of the MySQL server.

  You're not logging into the Linux computer as the Root user; you're logging into the *MySQL* server as *it*s root user. This gives you total control over the MySQL server.

- **mysql**

  ```
  mysql -u root mysql
  ```

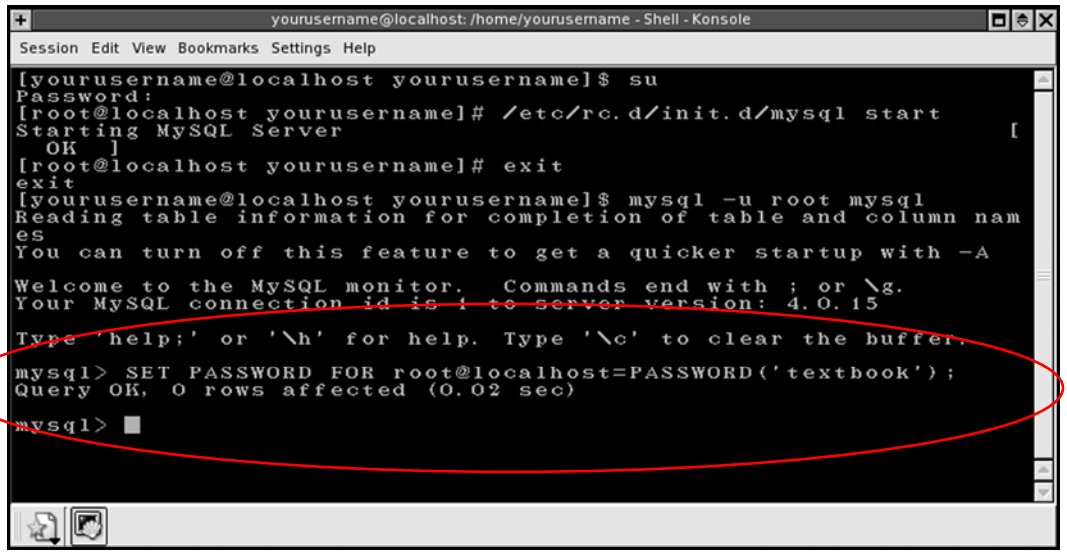
  This last `mysql` refers to a database called `mysql` that you'll use initially. This database is included by default in the MySQL server.

  The database `mysql` has several tables, including one that describes who can use the MySQL server.

8. **Type:**

```
SET PASSWORD FOR
root@localhost=PASSWORD('textbook');
```

The window should look like this:



This string of commands sets the password for the root user on the MySQL server to `textbook`.

Tip: ***Both the MySQL server and the Linux computer itself can have root users who can add/delete/modify anything. The passwords for each are independent, however.***

`textbook` ***is not the Root account password of your Linux computer. It's the root password for the MySQL server.***

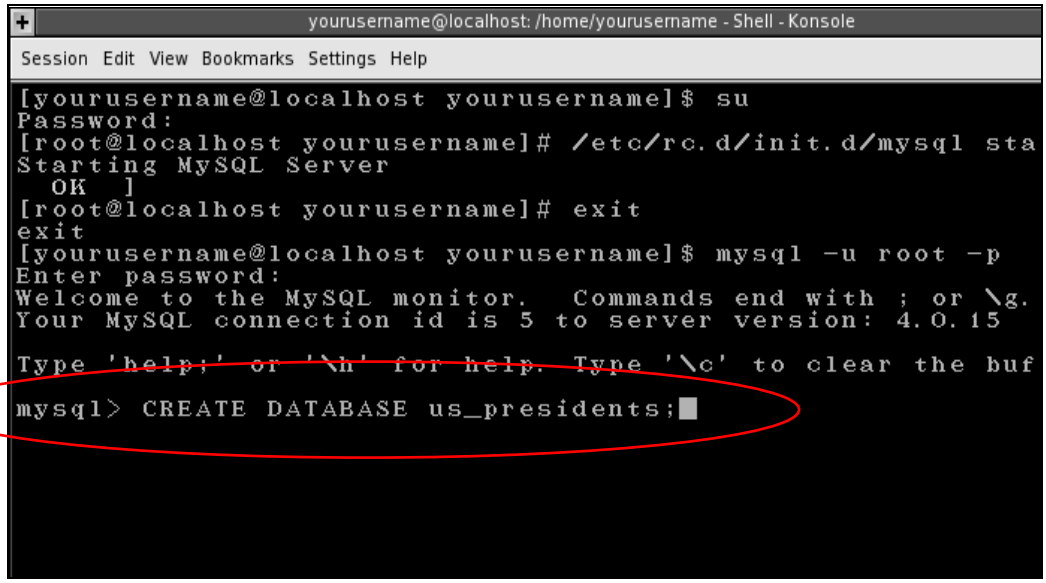
In the previous string of commands, you logged into the MySQL server as its root user, so the password `textbook` ***applies to the MySQL server.***

You can now give commands to add/delete/modify anything in the MySQL server, but not the Linux computer it runs on.

Create a new database

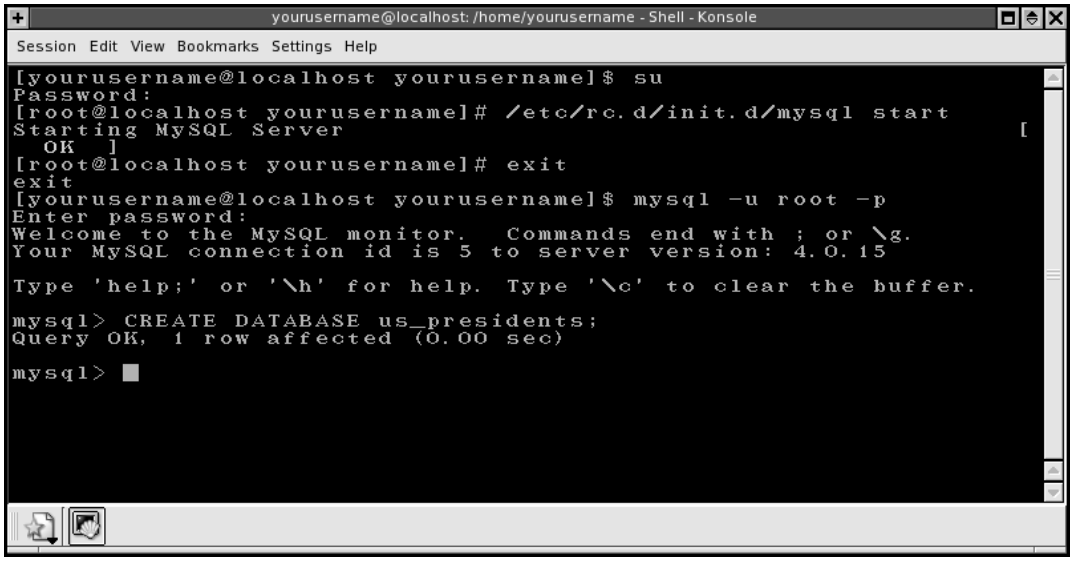
1. At the `mysql>` prompt, type:

```
CREATE DATABASE us_presidents;
```



then press **ENTER.**

The window should look like this:



Tip: *Now that you're logged into the MySQL server, you're giving MySQL commands.*

Unlike Linux commands, MySQL commands need a semicolon (`;`) on the end to execute.

The `CREATE DATABASE` *command created a database called* `us_presidents` *in the MySQL server.*

If ever you mistakenly end a command string with a character other than a semicolon…

```
CREATE DATABASE us_presidents
```

…then press **ENTER**, *there is no way to "fix" that command.*

Just add a semicolon to the new line you are on:

```
CREATE DATBASE us_presidents
;
```

If the command is valid, it will execute.

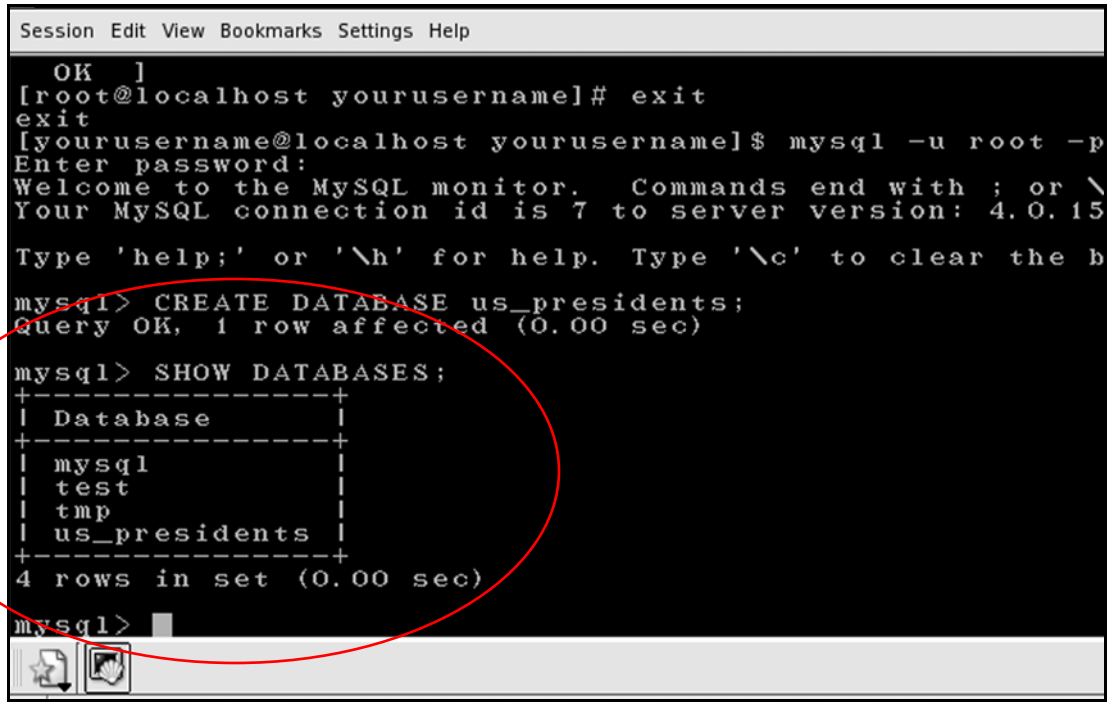
If there was an error in the command string and it's invalid, adding a semicolon here will execute it and MySQL will give an error.

2. Type:

```
SHOW DATABASES;
```

then press **ENTER**.

The window should look like this:



This shows the databases on your MySQL server: `mysql`, `test`, `tmp`, and `us_presidents`.

The other databases, mysql and tmp, are used by the MySQL server to store information about users, permissions, etc. The test database is often used as a workplace for MySQL users to test and try things – this is useful in a work environment where many people are working with critical information.

Tip: *MySQL commands don't have to be* `UPPER-CASE`.

In this book, commands are put in `UPPER-CASE` *to make them easier to distinguish.*

If you'd typed the command in `lower-case`:

```
show databases;
```

that would have been fine.

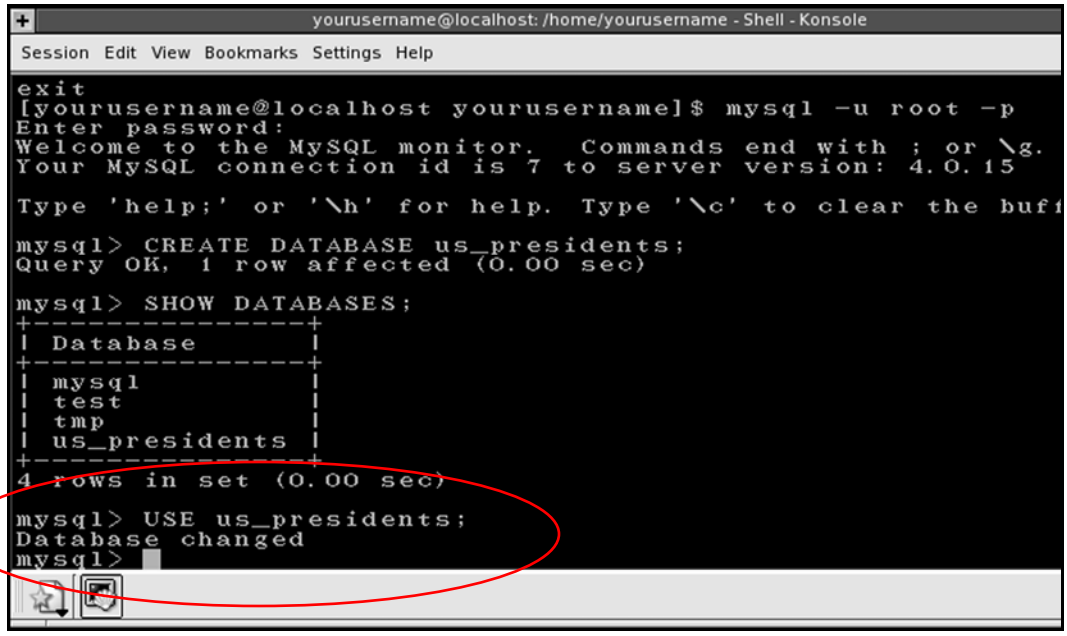
Create a table

1. Type:

   `USE us_presidents;`

   then press **ENTER**.

   The window should look like this:



   The `USE` command allows you to start using the database `us_presidents`.

Displaying text

Sometimes a string of commands is too wide to fit on the pages of this book. In those cases, an arrow is added that tells you to continue typing in the same line.

For instance, this command:

```
rpm -i MySQL-3.23.51-1.i386.rpm MySQL-client-
3.23.51-1.i386.rpm
```

could be displayed this way:

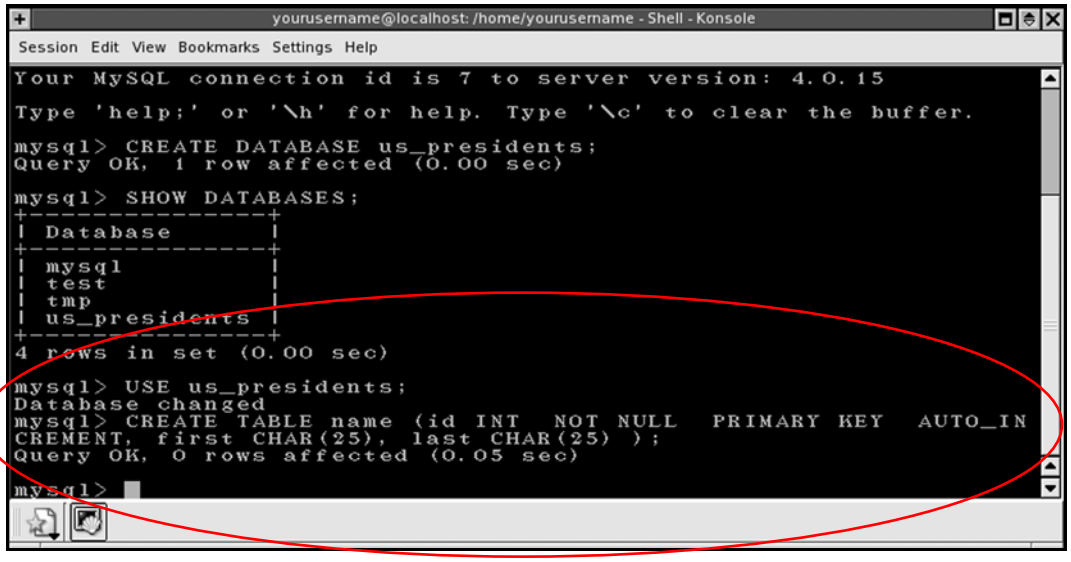
```
rpm -i MySQL-3.23.51-1.i386.rpm ➤➤
MySQL-client-3.23.51-1.i386.rpm
```

2. Type:

```
CREATE TABLE name ▶▶
(id INT  NOT NULL  PRIMARY KEY ▶▶
AUTO_INCREMENT, ▶▶
first CHAR(25), last CHAR(25) );
```

then press **ENTER**.

The window should look like this:



This string of commands is used to `CREATE` a `TABLE` called name with three fields: `id`, `first`, and `last`.

Here are the datatypes and properties for these fields:

- `INT`

  ```
  CREATE TABLE name
  (id INT NOT NULL PRIMARY KEY
  AUTO_INCREMENT,
  first CHAR(25), last CHAR(25) );
  ```

  The `INT` datatype for the `id` field ensures it will contain only integers—numbers, not text.

- **NOT NULL**

```
CREATE TABLE name
(id INT NOT NULL PRIMARY KEY
AUTO_INCREMENT,
first CHAR(25), last CHAR(25) );
```

The `NOT NULL` property ensures the `id` field cannot be left blank.

- **PRIMARY KEY**

```
CREATE TABLE name
(id INT NOT NULL PRIMARY KEY
AUTO_INCREMENT,
first CHAR(25), last CHAR(25) );
```

The `PRIMARY KEY` property makes `id` the key field in the table.

In any database table, one field should be the key field—a field that can contain no duplicates. In this table, `name`, the `id` field is the key field because it contains the `PRIMARY KEY` property.

This means the `name` table can't have two records with an `id` of 35.

- **AUTO_INCREMENT**

```
CREATE TABLE name
(id INT NOT NULL PRIMARY KEY
AUTO_INCREMENT,
first CHAR(25), last CHAR(25) );
```

The `AUTO_INCREMENT` property automatically assigns a value to the `id` field, increasing the previous id number by one for each new field.

This ensures that the `NOT NULL` (can't be blank) and the `PRIMARY KEY` (can't have duplicates) properties of the `id` field are both satisfied.

- **`CHAR`**

```
CREATE TABLE name
(id INT  NOT NULL  PRIMARY KEY
AUTO_INCREMENT,
first CHAR(25), last CHAR(25) );
```

The `CHAR` datatype for the first and last fields limits the length of entries to `25` characters each.

In the `us_presidents` database, you've created a table called `name` that's organized like this:

Field	Datatype	Properties
Id	**INT**	**primary key, not null, auto increment**
first	**CHAR(25)**	
last	**CHAR(25)**	

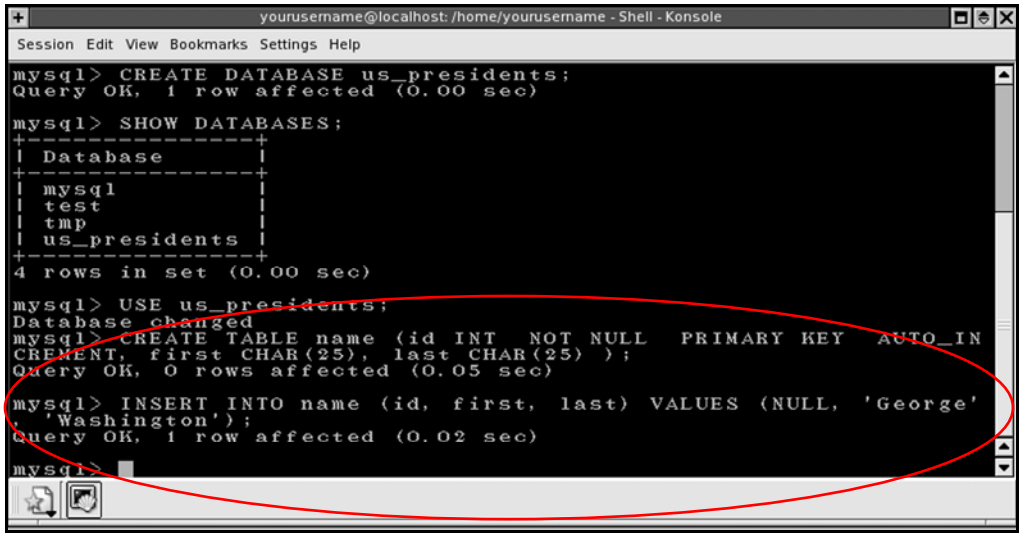
Create a record

1. Type:

```
INSERT INTO name (id, first, last) ▶▶
VALUES (NULL, 'George', 'Washington');
```

then press **ENTER**.

The window should look like this:



This command string creates the first record in the table `name`. It reads much like a sentence:

`INSERT INTO` the table `name` (which has the fields `id`, `first`, and `last`)
the corresponding `VALUES NULL, George,` and `Washington`.

Since the `id` field can't be blank (it has a `NOT NULL` property), putting a NULL value in it forces MySQL to automatically number the record (because the id field also has the property `AUTO_INCREMENT`).

The data in the table `name` is now organized like this:

Fields:	id	First	last
Record:	1	George	Washington

Tip: ***Text is enclosed within single quotes to let MySQL know that it's just text, not a command.***

If the phrase

```
'What is the first name of the president
named Washington whose values kept him from
cutting down the cherry tree?'
```

was not enclosed in single quotes, MySQL might interpret the words `name` ***and*** `values` ***as commands, and get confused.***

In these examples, single-quotes are used. Double-quotes perform the same function.

2. **Type:**

```
INSERT INTO name (id, first, last) ▶▶
VALUES ▶▶
(NULL, 'John', 'Adams'), ▶▶
(NULL, 'Thomas', 'Jefferson'), ▶▶
(NULL, 'James', 'Madison');
```

then press ENTER.

This adds three records to the table name: one record each for presidents John Adams, Thomas Jefferson, and James Madison.

The data in the table `name` are now organized like this:

Fields:	id	first	last
Records:	1	George	Washington
	2	John	Adams
	3	Thomas	Jefferson
	4	James	Madison

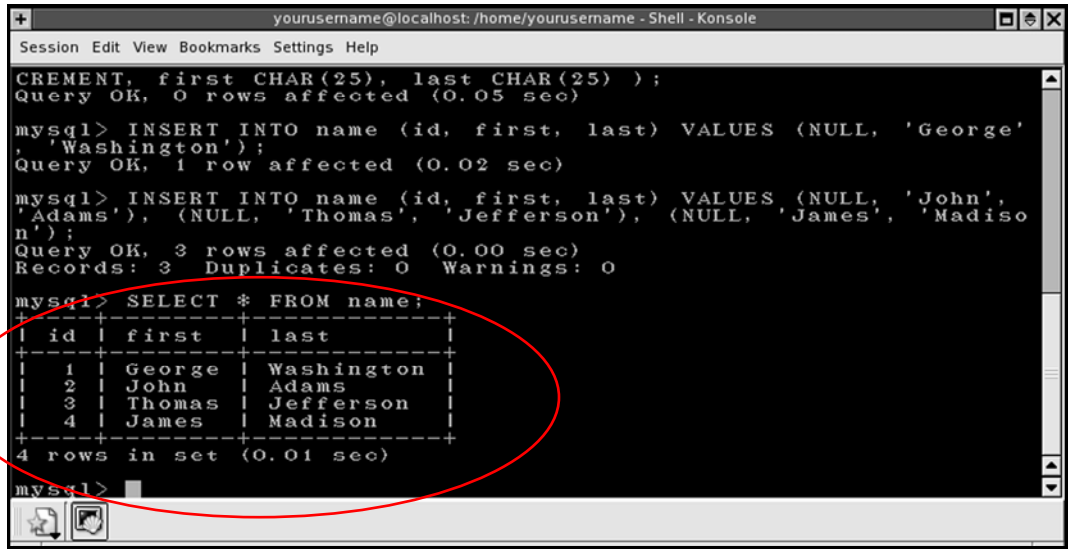
Run a query

1. Type:

   ```
   SELECT * FROM name;
   ```

   then press **ENTER**.

   The window should look like this:



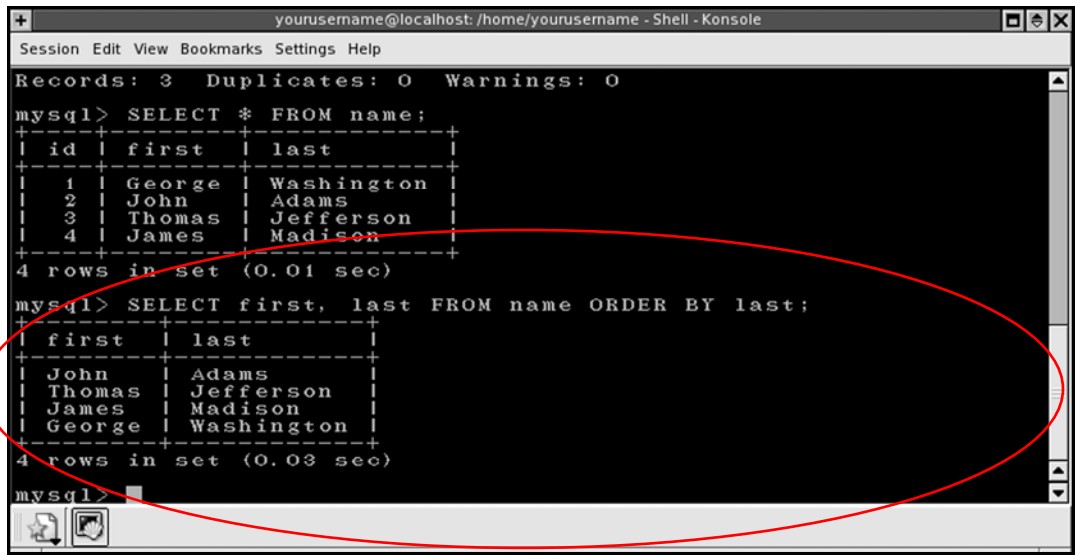
   The `SELECT` command tells MySQL to perform a query.

   The asterisk (`*`) command tells MySQL to return everything (the asterisk means "everything" or "all") that's in the table `name`.

2. Type:

```
SELECT first, last FROM name ▶▶
ORDER BY last;
```

The window should look like this:



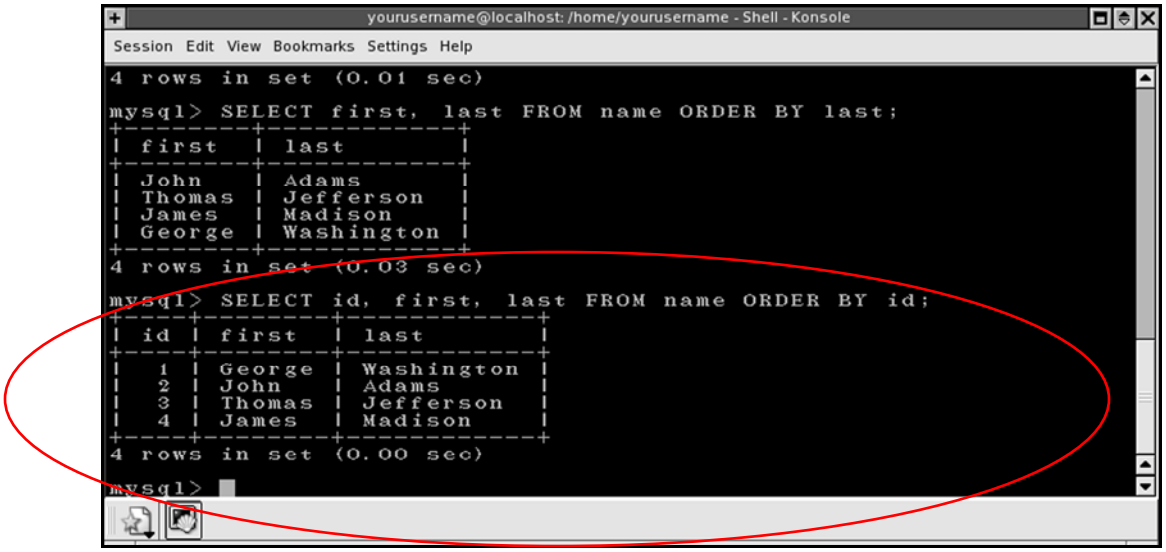
This query is more precise than the previous one: it selects the fields `first` and `last` from the table `name`.

`ORDER BY` puts the records in alphabetical order, based on the field `last`. In other words, it puts the presidents' last names in alphabetical order.

3. Type:

```
SELECT id, first, last FROM name ▶▶
ORDER BY id;
```

The window should look like this:



In this `query`, `ORDER BY id` places the records in numeric order, based on their id numbers.

Tip: *To arrange records in reverse numeric or reverse alphabetical order, add* `DESC` *on the end. For instance, type:*

```
SELECT first, last FROM name ORDER BY last
DESC;
```

The `DESC` *option refers to the word "descending." It tells MySQL to order things descending from high to low instead of the default: low to high.*

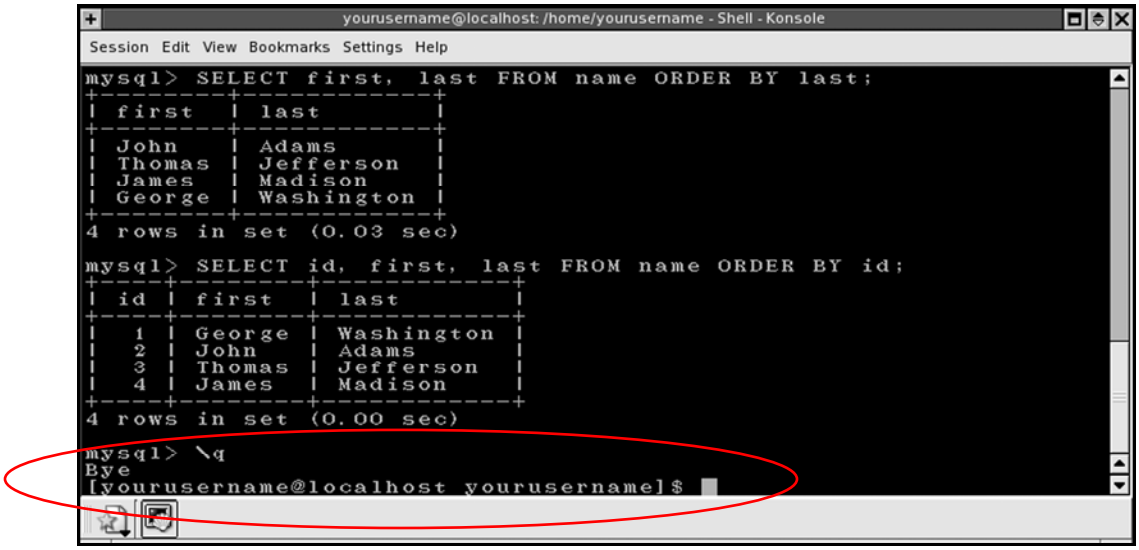
4. Type:

```
\q;
```

then press **ENTER**.

This closes your MySQL database connection.

You are now logged out of the MySQL server: the `mysql>` prompt is gone.



5. **Type:**

```
exit
```

then press ENTER.

The Konsole window should close.

Giving MySQL commands to a Web server

MySQL's client/server arrangement makes it well-suited to Web applications. With MySQL server running on a Web server, you can use a MySQL client to update/add/delete data remotely.

This book assumes that you've installed MySQL on your desktop Linux computer. Both the MySQL client and server programs are on this computer, called `localhost`.

To give commands to a MySQL server program running on a Linux Web server, just replace `localhost` with the IP address of the Web server, such as

```
10.0.1.10
```

or the domain name of the Web server, such as

```
mysql.domain.com
```

Provided you have an Internet connection with the Web server, and the proper username/password to access it, your commands will work.