

THE VISIBOOKS GUIDE TO MySQL Basics



See. Do. Learn.

Table of Contents

Getting Started	1
Install MySQL on a Linux computer.....	3
Start MySQL.....	13
Create a new database.....	24
Create a table	29
Create a record	35
Run a query.....	38
Administering Databases	49
Restart MySQL.....	50
Back up a database	53
Delete a table	61
Delete a database	63
Restore a database.....	64
Working with Tables	71
Alter tables.....	72
Update records	75
Delete records	79

Running Queries	85
Sort query results	86
Add query criteria	96
Securing a database	105
Add a local user	106
Add a remote user.....	109
Remove a user	111
Restrict a user	112
Web-enabling Databases.....	115
Perform a query using PERL	116
Join two tables in PERL	130
Create a CGI script.....	134
Write a query in a CGI script	143

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

1. Obtain a copy of Linux.


Tip: *A good version of Linux to use with this book is Linspire. It's very user-friendly.*

You can buy or download a copy at:

www.linspire.com



2. Install Linspire.

3. On the Launcher Bar, click the  icon.



4. When the **Sign In** screen appears, type your email address and password in the appropriate fields.

Make sure **Yes, I have an account password** is checked, then click the  button.



Tip: If you need to create an account, type your email address in the **E-mail Address** box.

Make sure **No, I need to create a new account** is checked, then click the **Go** button.

Sign In Or Create Account Here

- Updates for your operating system.
- Updates for programs you have installed.
- Enhancements, bug fixes and security patches.

What is your e-mail address?

E-mail Address:

Do you have an account password?

No, I need to create a new account. **It's free!**

Yes, I have an account password:

Remember me?

Forgot your password? [Click here](#)
Need help logging in? [Click here](#)

Fill in the email, password, and name fields in the **Account Information** screen. Then click the **Continue** button.

Account Information

My.Linspire Sign In
To ensure that your account is secure and private, please enter and confirm your email address and password, then enter your name and click "Continue"

E-mail Address:

Confirm E-mail Address:

Password:

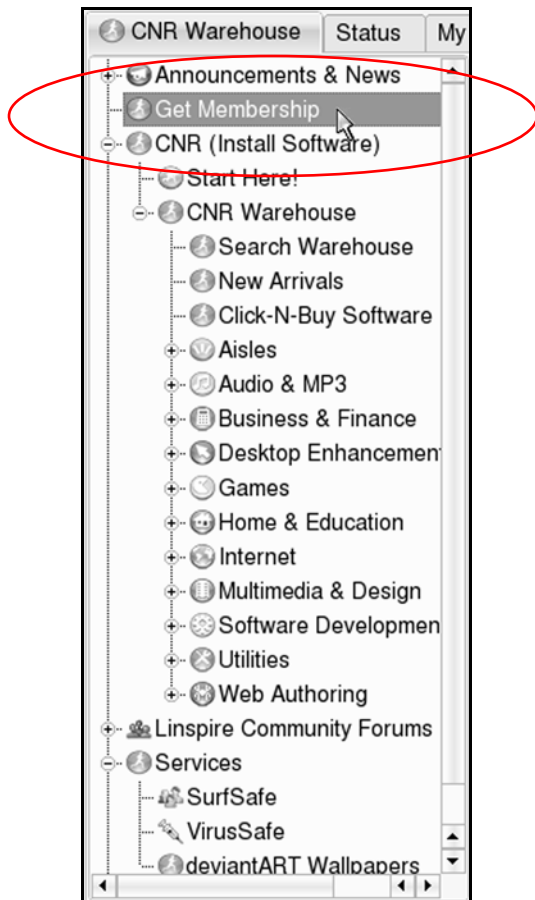
Confirm Password:

First Name:

Last Name:

As a valued Linspire customer, we would like to keep you informed of news and happenings, new products and services, and special offers we believe may interest you. Please uncheck this box if you don't want to receive these emails.

*After the account is created, you need to register for the CNR Service. In the left navigation pane, click **Get Membership**.*



When the Linspire Shopping Navigator screen appears, click the **Buy Now!** button under the CNR Service of your choice.

The screenshot shows the Linspire Shopping Navigator website. At the top, there is a search bar with "Warehouse" selected, a "Go" button, and links for "SHOPPING CART", "YOUR ACCOUNT", and "SIGN ON". The email address "meg@circaweb.com" is visible. The main content area features two service options: "CNR Service" for \$19.95/yr. and "CNR Gold Service" for \$49.95/yr. Both have "Buy Now!" buttons. A red circle highlights the "Buy Now!" button for the CNR Service. To the right, there is a circular icon of a runner and the text "The Easy Way to Get Linux Software". Below this, a paragraph explains that CNR is one of the main reasons why Linspire is the World's Easiest Desktop Linux. A navigation bar at the bottom contains links for "What is CNR", "Why CNR?", "Screenshots", "CNR Warehouse", "Free Aisle", "Aisles", and "Five-0 CNR Edition".

CNR Service
\$19.95/yr.
Buy Now!

CNR Gold Service
\$49.95/yr.
Buy Now!

The Easy Way to Get Linux Software

CNR is one of the main reasons why Linspire is the World's Easiest Desktop Linux. It is the "magic" that makes Linspire such a great value.

[What is CNR](#) [Why CNR?](#) [Screenshots](#) [CNR Warehouse](#) [Free Aisle](#) [Aisles](#) [Five-0 CNR Edition](#)

What is CNR?

CNR stands for "click and run". It is a software delivery service designed for Linspire users that makes it easy to install Linux software.

With the CNR Service you can install more than 2,000 FREE Linux software titles direct from the CNR Warehouse - all with just a single click.

In fact, the CNR Service is the easiest way to install Linux software. Simply click the software you want and it installs on your computer and is ready to run.

And the CNR Service gives you more than just one-click access to tons of free software.

You also get a powerful, easy way to manage your entire software library. Like customizable aisles where you can install entire groups of software with a single

Benefit	CNR Service	CNR Gold Service
One-Click Software Installs	✓	✓
One-Click Software Updates	✓	✓
"No Problem" Software Installs	✓	✓
Personal CNR Warehouse Aisle	✓	Unlimited
Priority Learning	✓	✓
Security Updates *	✓	✓
Software Update Notifications	✓	✓
High Discounts on Commercial Linux Software	✓	✓
Priority Support	✓	✓
Free Operating System Updates *	✓	✓
Free Limited Personal Productivity Suite	✓	✓
Free Download! Desktop Wallpaper	✓	✓
Discounts on Linspire Merchandise	✓	✓

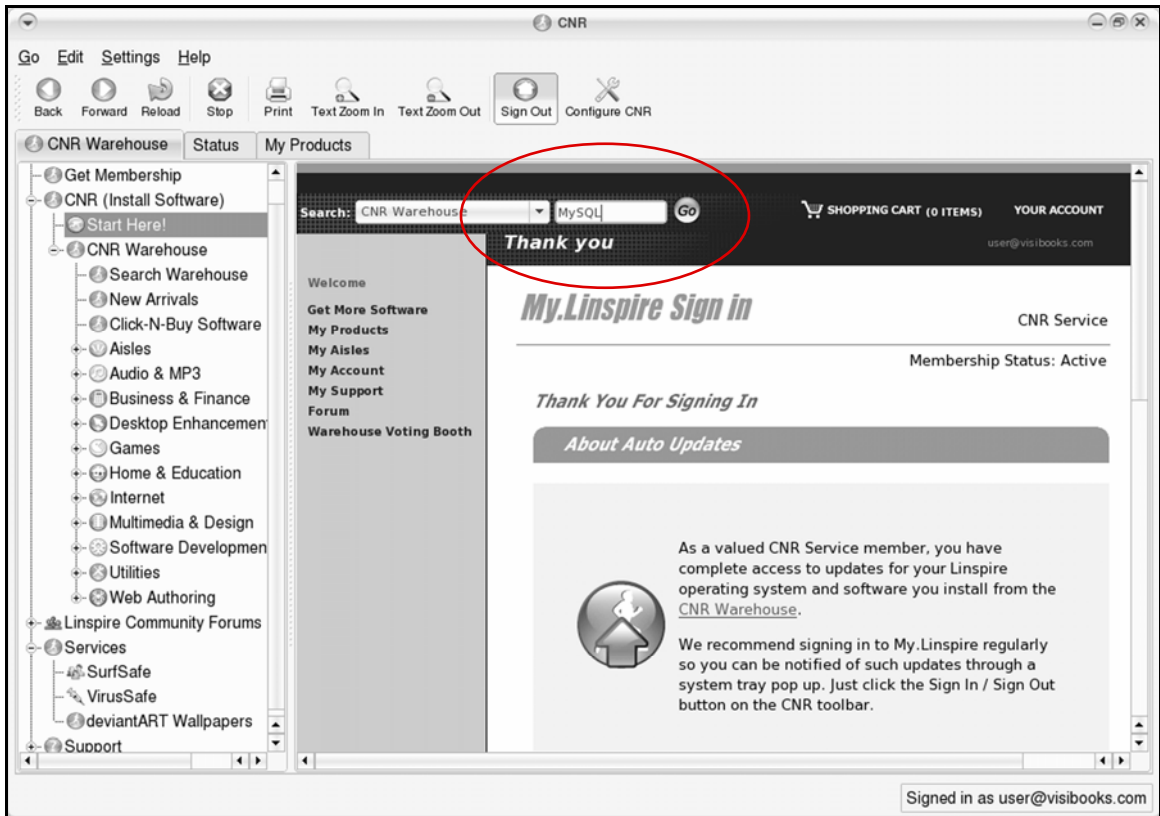
Compare the Benefits of CNR and CNR Gold


Complete the check out process. You are now logged in as a CNR member.

5. In the **CNR** window, type:

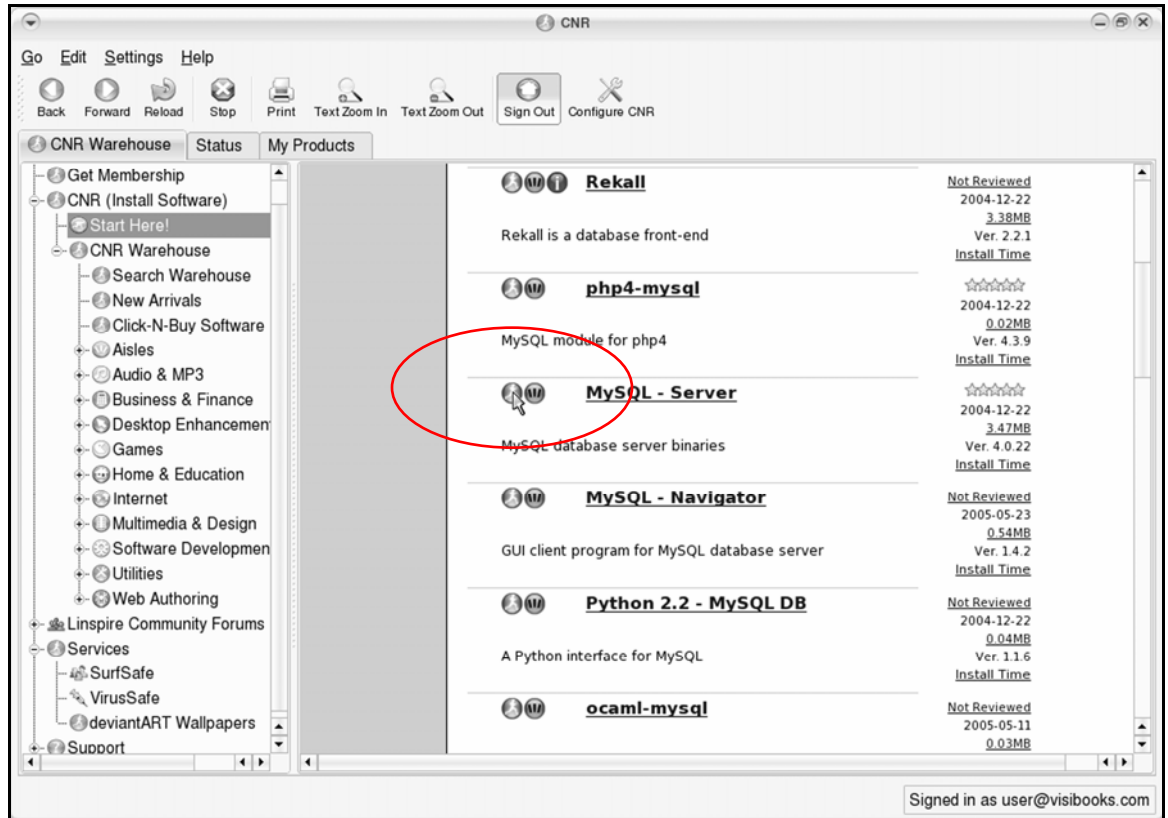
MySQL

in the **Search** box.



6. Click the  button.

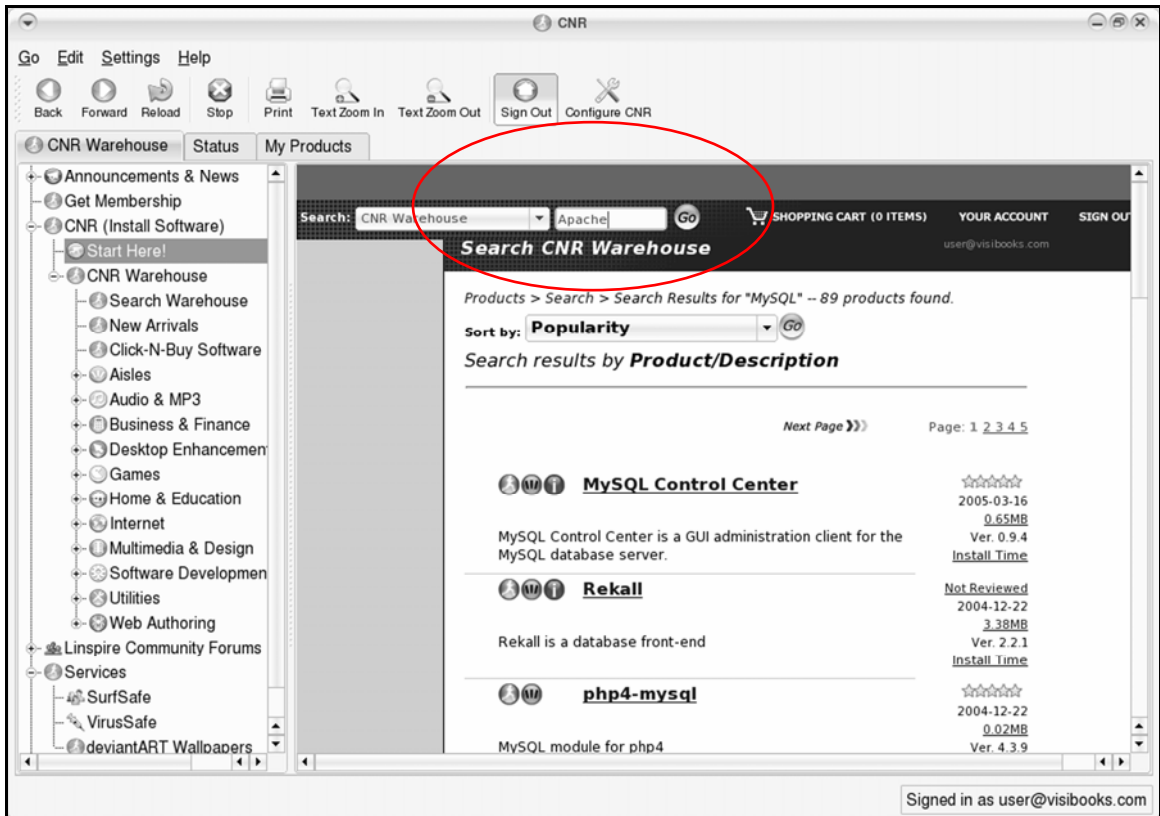
7. When the search results appear, scroll down and click the  icon next to **MySQL-Server**.





MySQL Server is installed.

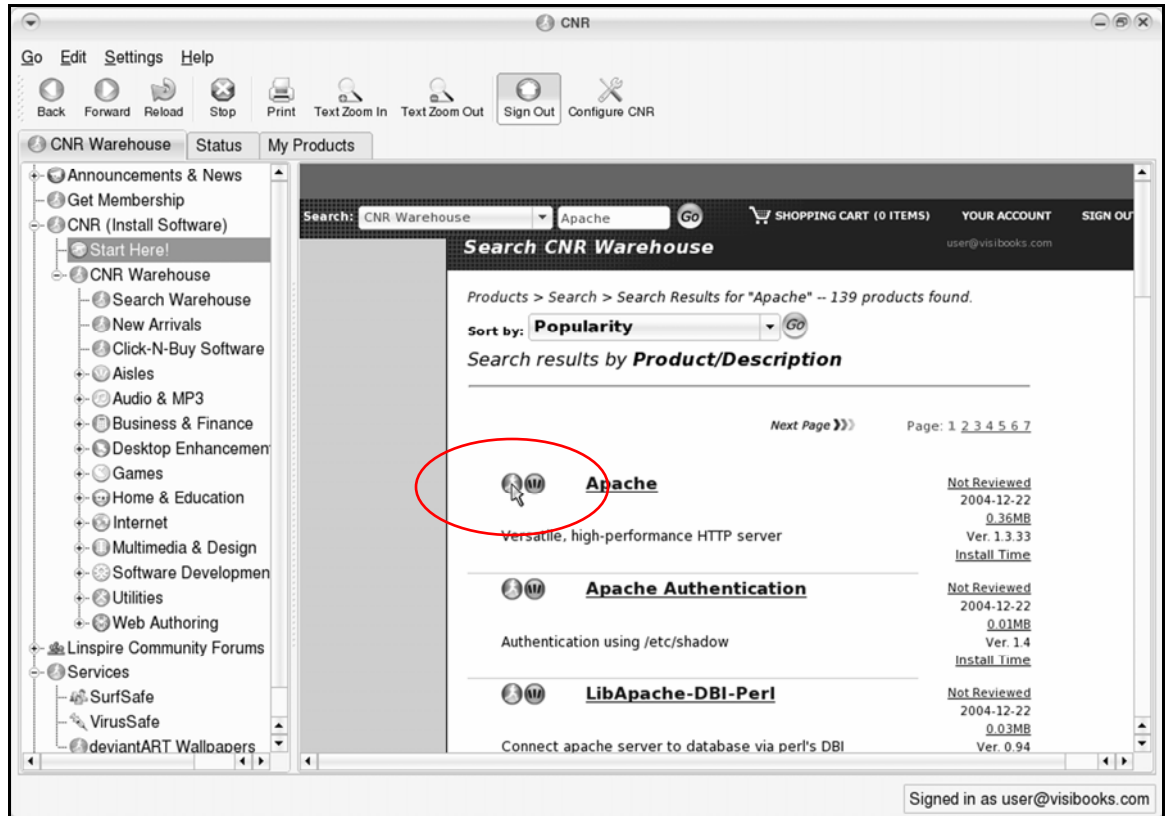


- 8.** Scroll back to the top of the **CNR** window. Type: **Apache** in the **Search** box.

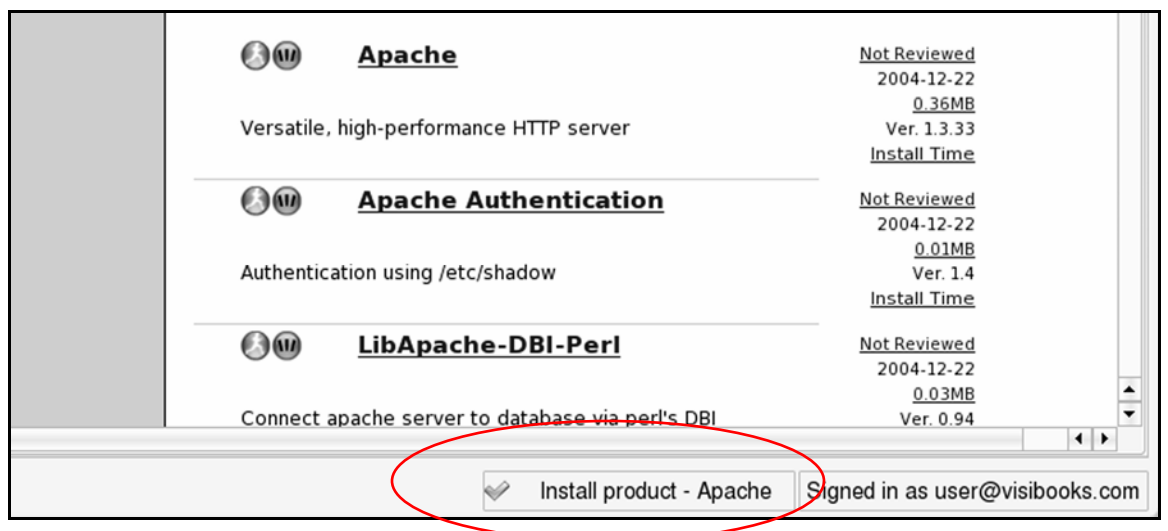


- 9.** Click the  button.

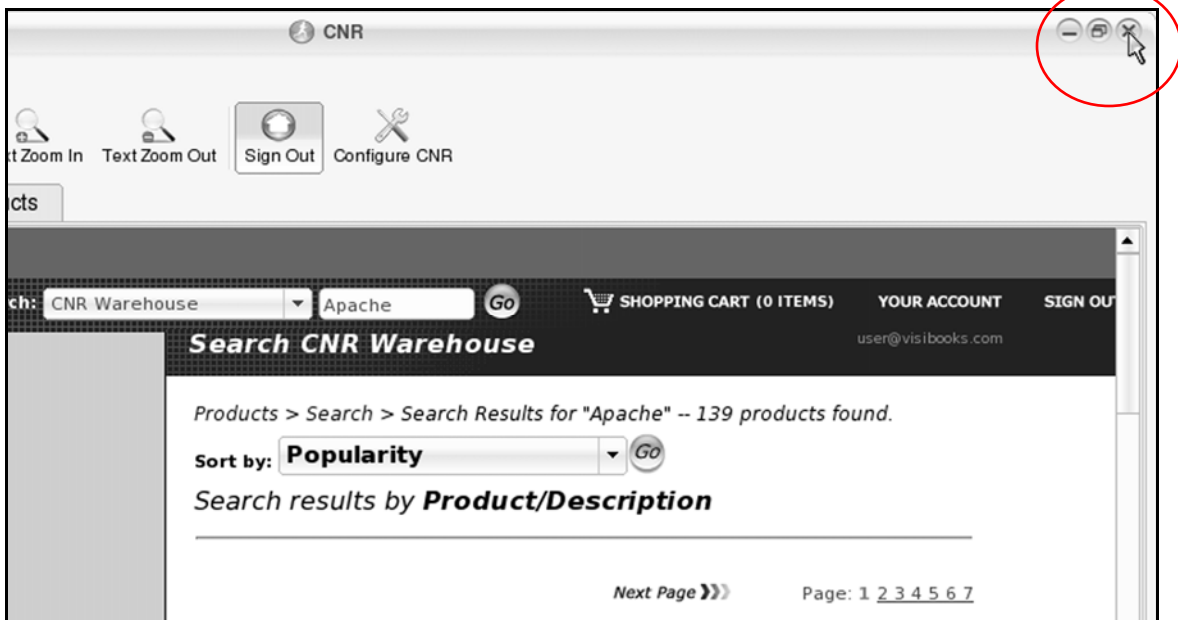
10. When the search results appear, click the  icon next to **Apache**.



The Apache Web Server is installed.




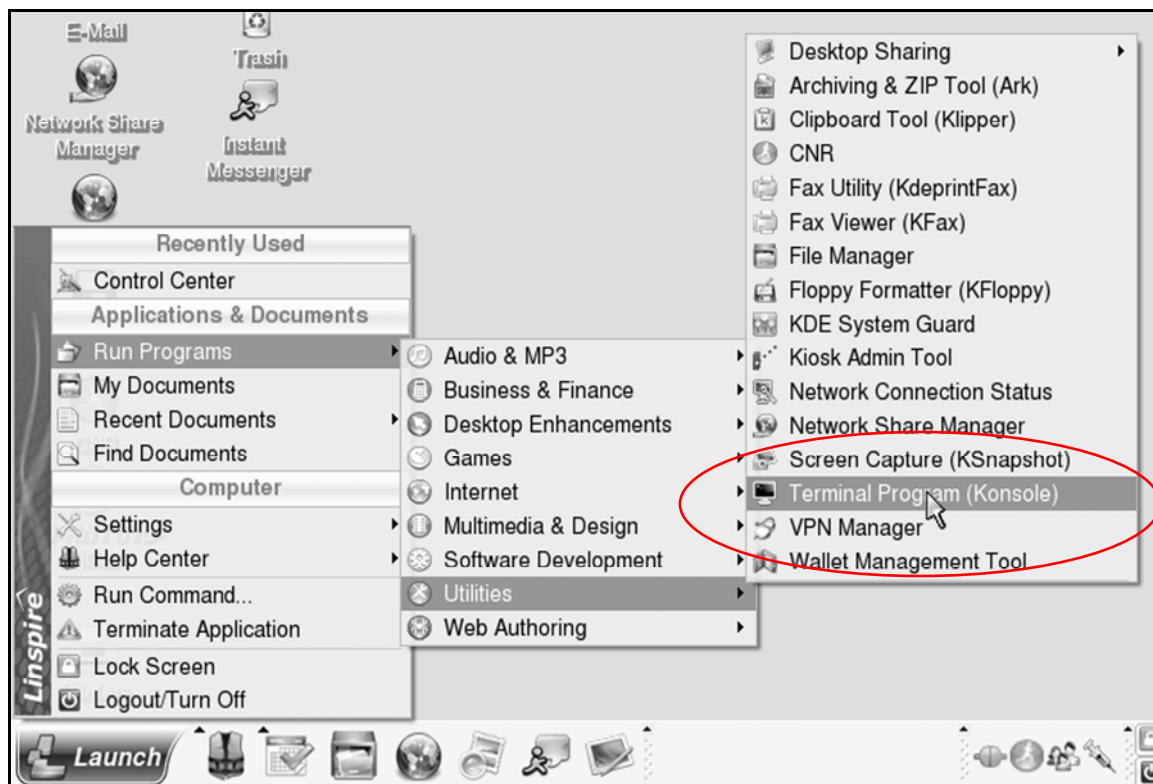
11. Close the CNR window.



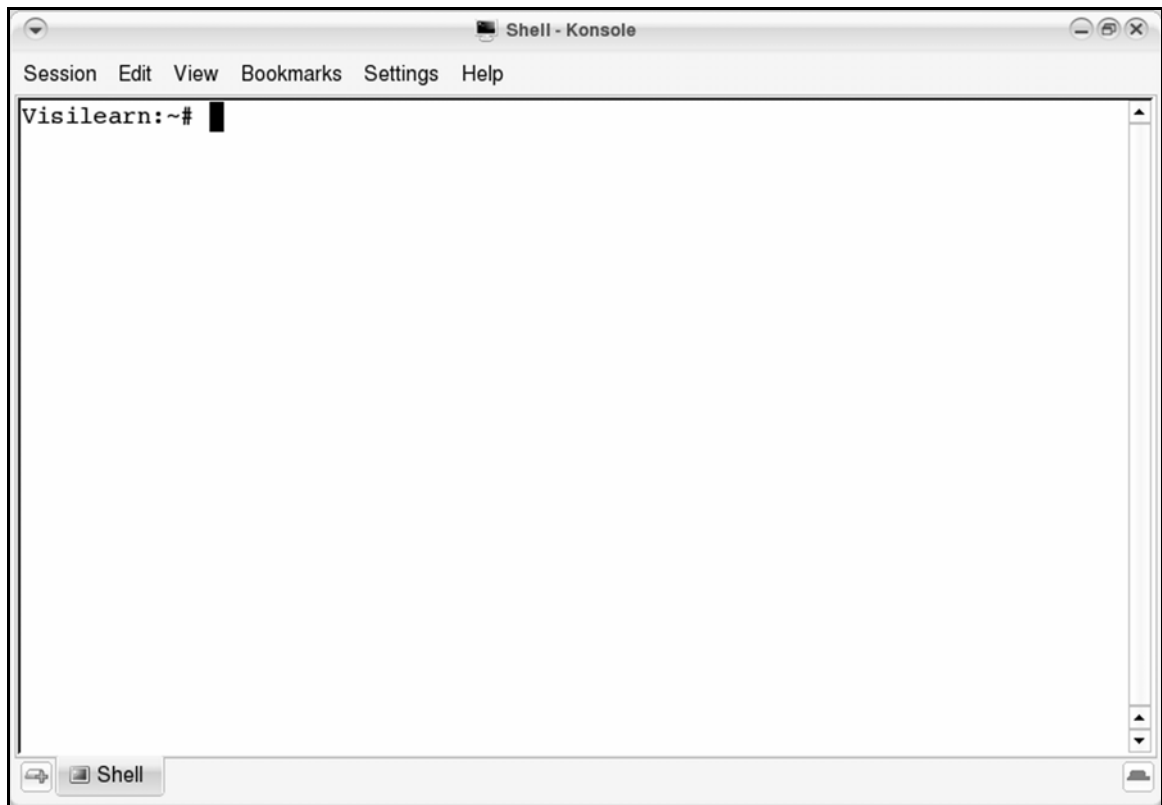
12. Restart your computer.

Start MySQL

1. Click the  button, then **Run Programs**, then **Utilities**, then **Terminal Program (Konsole)**.



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



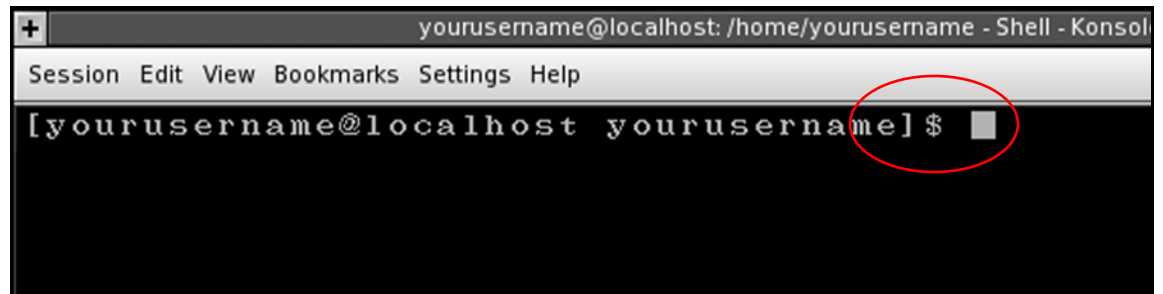
Tip: *In Linspire, the prompt is followed by a #*

Visilearn:~#

as you see above.

means you're giving commands as the Root user. The default user in Linspire is the Root user.

On other Linux distributions the terminal prompt is followed by a \$.



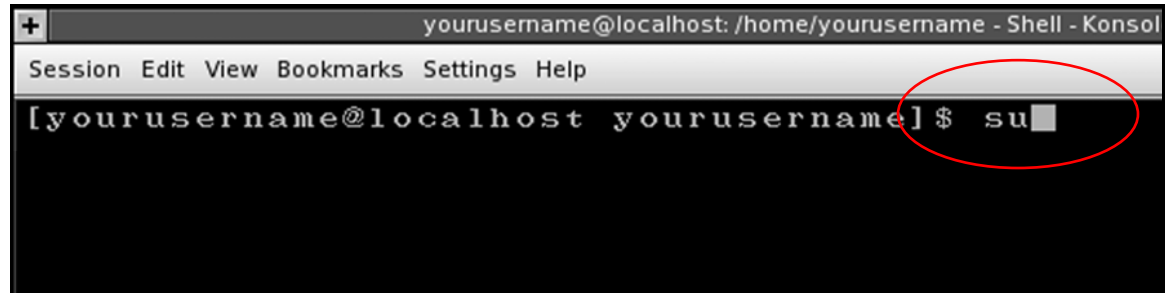
```
yourusername@localhost: /home/yourusername - Shell - Konsol
Session Edit View Bookmarks Settings Help
[yourusername@localhost yourusername]$
```

*\$ means you're giving Linux commands as a regular user. Giving the **su** command allows you to give commands as the "Super User," or Root user, of the computer.*

If your terminal prompt is followed by a \$, type

su

at the prompt.

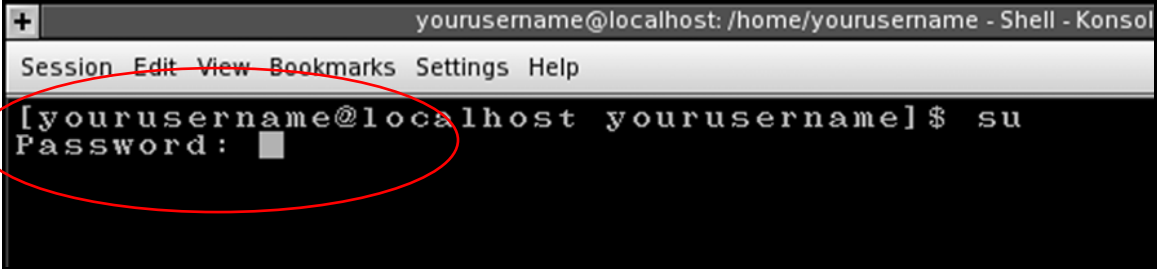


```
yourusername@localhost: /home/yourusername - Shell - Konsol
Session Edit View Bookmarks Settings Help
[yourusername@localhost yourusername]$ su
```

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

At the Password prompt, type:

Your Root user password



```
yourusername@localhost: /home/yourusername - Shell - Konsole
Session Edit View Bookmarks Settings Help
[yourusername@localhost yourusername]$ su
Password: █
```

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

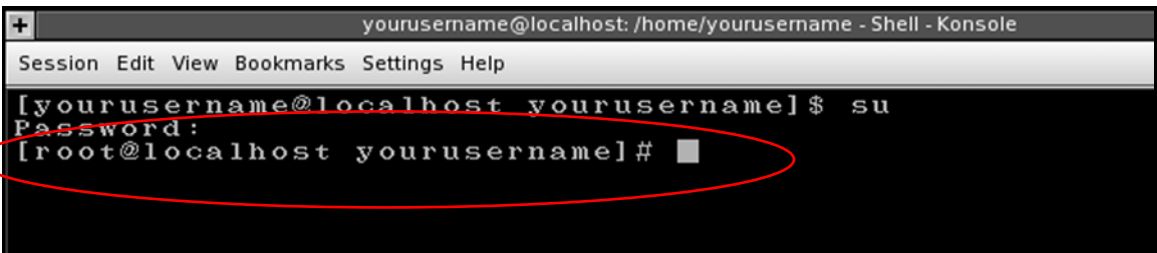
*Then press the **ENTER** key.*

Notice the prompt has changed from

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

to

```
[root@localhost yourusername]#
```

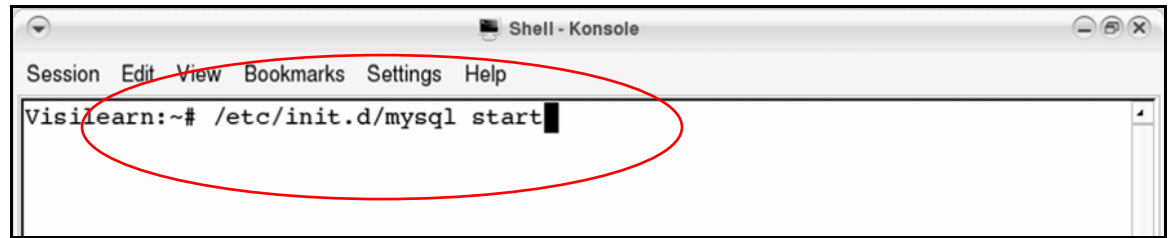


```
yourusername@localhost: /home/yourusername - Shell - Konsole
Session Edit View Bookmarks Settings Help
[yourusername@localhost yourusername]$ su
Password:
[root@localhost yourusername]# █
```

There's now a # at the end of the prompt. This means you are now giving commands as the Root user. As the Root user, you can add/delete/modify any file on the computer.

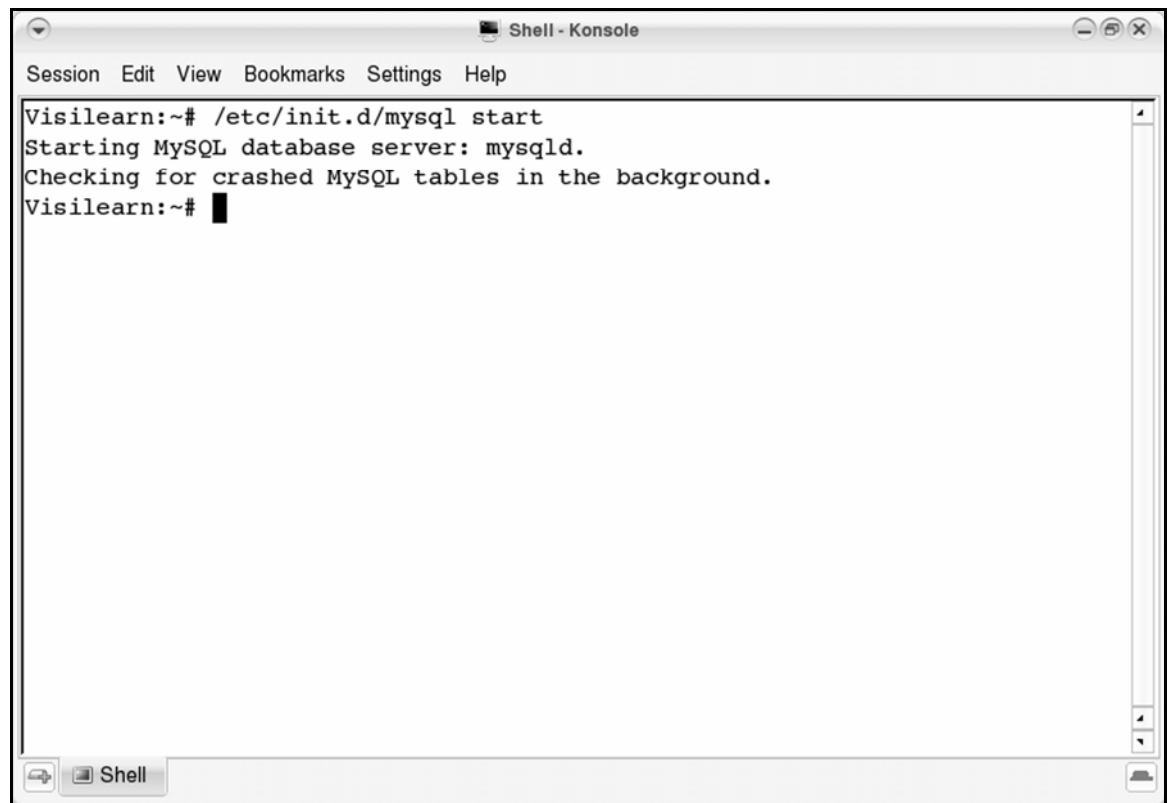
3. Type:

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



Then press **ENTER**.

The window should look like this:

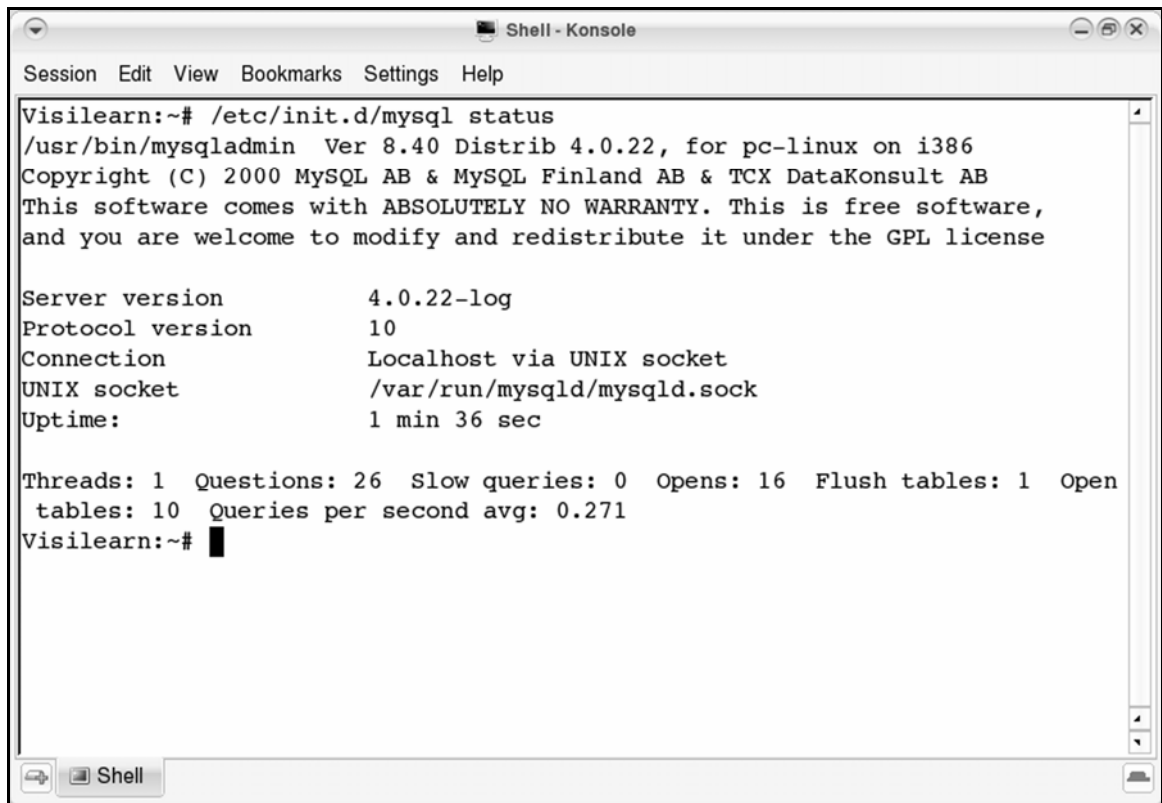


This starts the MySQL server—the program `mysql` in the `/etc/init.d/` directory.

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

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

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



The image shows a terminal window titled "Shell - Konsole" with a menu bar containing "Session", "Edit", "View", "Bookmarks", "Settings", and "Help". The terminal output is as follows:

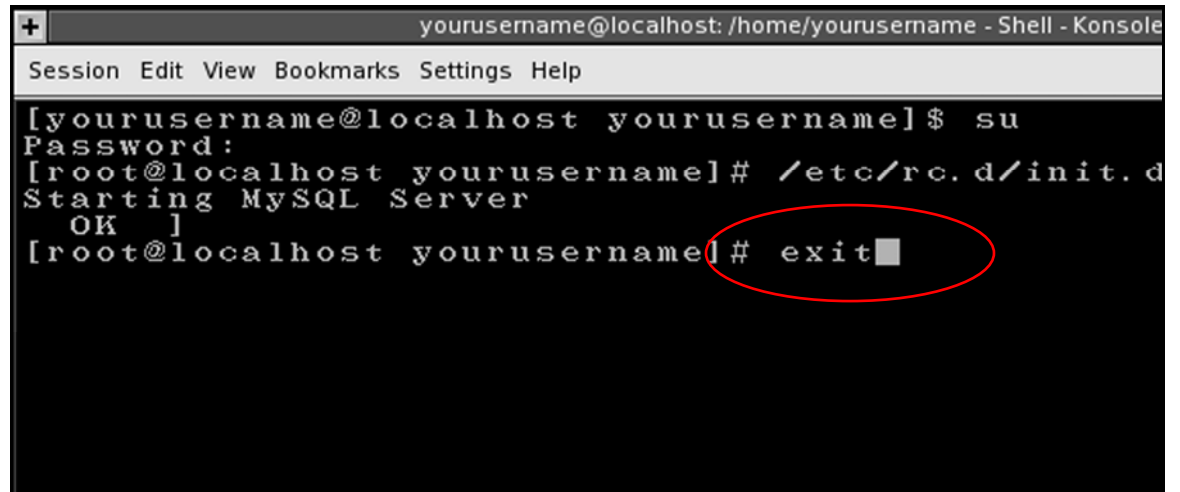
```
Visilearn:~# /etc/init.d/mysql status
/usr/bin/mysqladmin Ver 8.40 Distrib 4.0.22, for pc-linux on i386
Copyright (C) 2000 MySQL AB & MySQL Finland AB & TCX DataKonsult AB
This software comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to modify and redistribute it under the GPL license

Server version          4.0.22-log
Protocol version        10
Connection              Localhost via UNIX socket
UNIX socket             /var/run/mysqld/mysqld.sock
Uptime:                 1 min 36 sec

Threads: 1  Questions: 26  Slow queries: 0  Opens: 16  Flush tables: 1  Open
tables: 10  Queries per second avg: 0.271
Visilearn:~# █
```

Tip: *If you had to log in as the Super User earlier, type:*

`exit`

A terminal window titled "yourusername@localhost: /home/yourusername - Shell - Konsole". The terminal shows the following sequence of commands and output:

```
[yourusername@localhost yourusername]$ su
Password:
[root@localhost yourusername]# /etc/rc.d/init.d
Starting MySQL Server
  OK ]
[root@localhost yourusername]# exit
```

The "exit" command and its cursor are circled in red in the original image.

*Then press **ENTER**.*

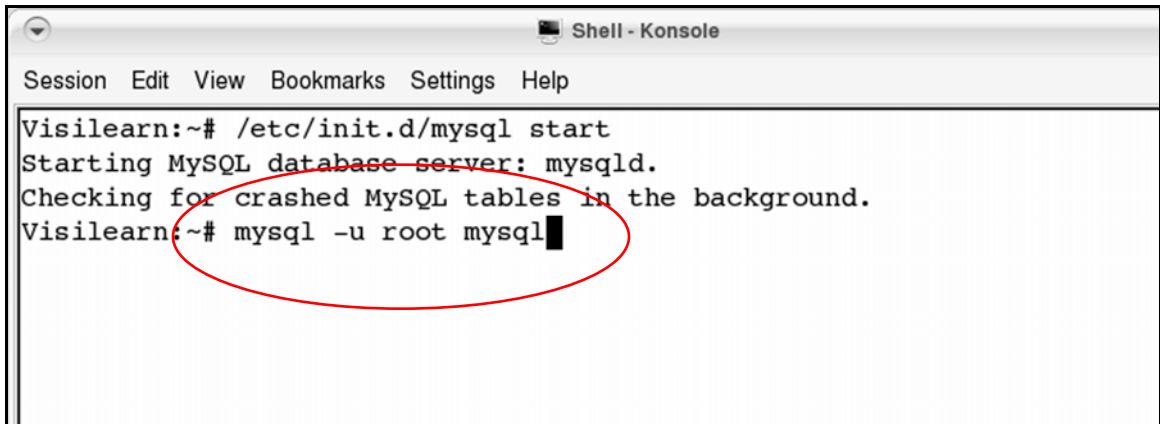
The prompt has now changed to:

```
[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.

4. At the prompt, type:

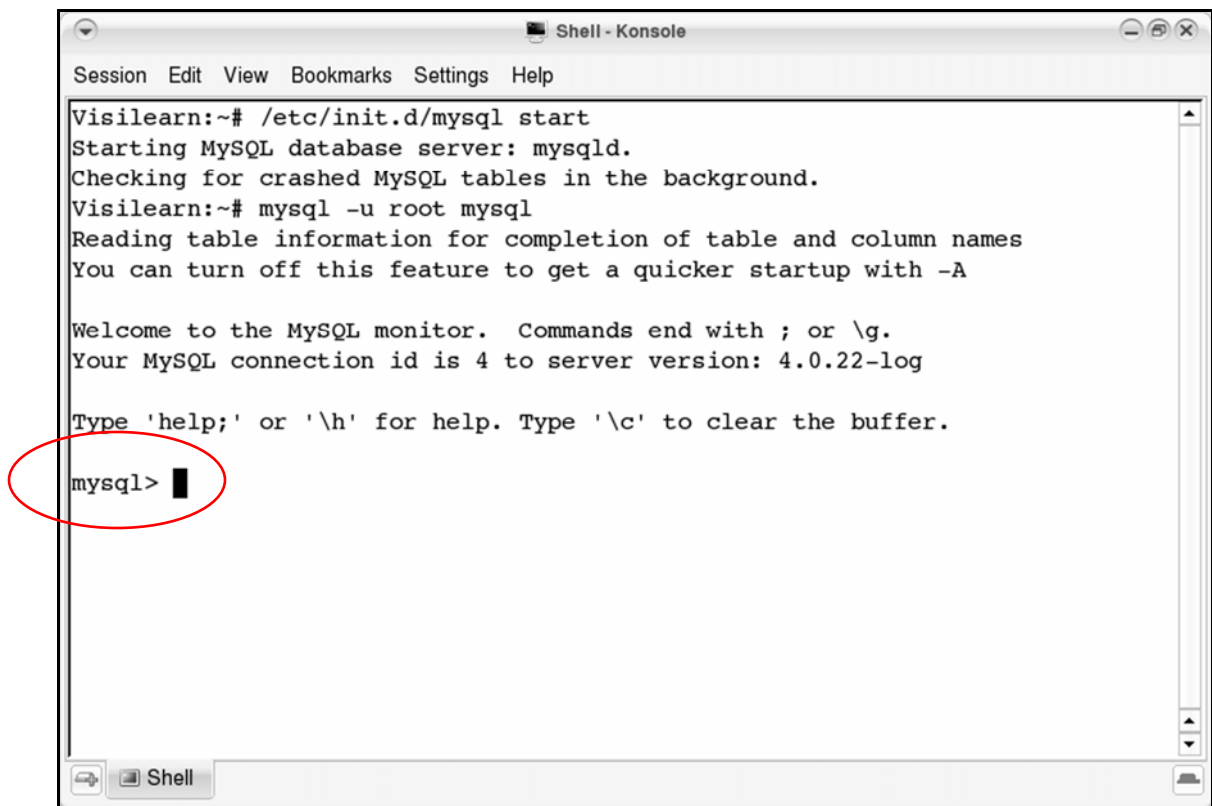
```
mysql -u root mysql
```



A terminal window titled "Shell - Konsole" with a menu bar (Session, Edit, View, Bookmarks, Settings, Help). The output shows the command `/etc/init.d/mysql start` being executed, followed by "Starting MySQL database server: mysqld." and "Checking for crashed MySQL tables in the background." The prompt `Visilearn:~# mysql -u root mysql` is shown with a cursor at the end, circled in red.

Then press **ENTER**.

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



The same terminal window now shows the MySQL prompt `mysql>` circled in red. The output includes: "Reading table information for completion of table and column names", "You can turn off this feature to get a quicker startup with -A", "Welcome to the MySQL monitor. Commands end with ; or \g.", "Your MySQL connection id is 4 to server version: 4.0.22-log", and "Type 'help;' or '\h' for help. Type '\c' to clear the buffer."

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 *its* 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.

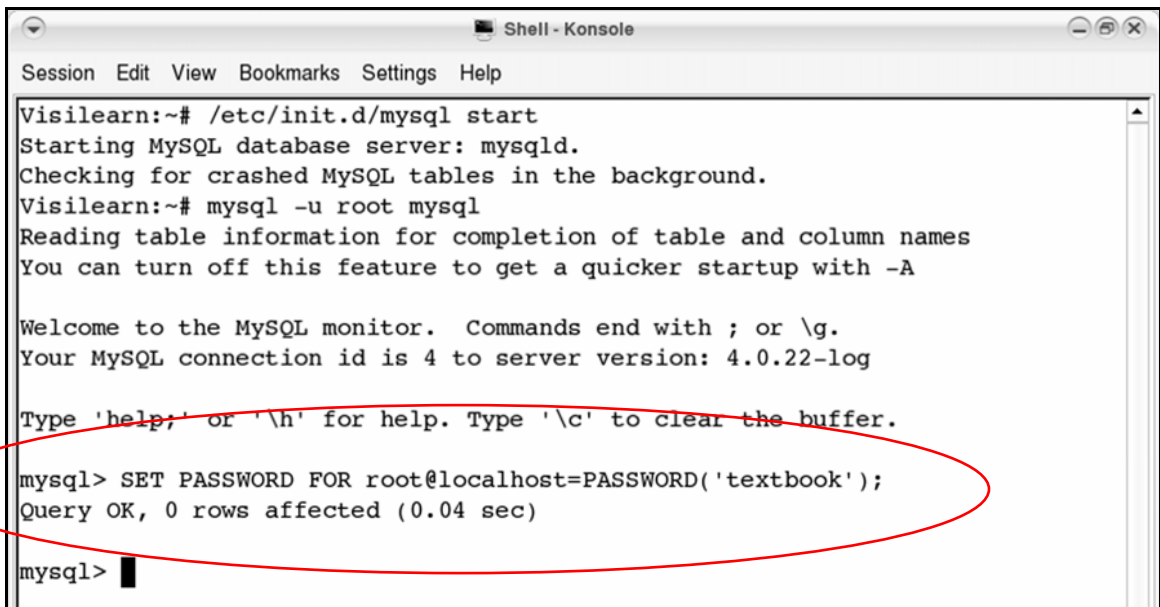
The database `mysql` has several tables, including one that describes who can use the MySQL server.

5. Type:

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

Then press **ENTER**.

The window should look like this:

A screenshot of a terminal window titled "Shell - Konsole". The window shows the following text: "Visilearn:~# /etc/init.d/mysql start", "Starting MySQL database server: mysqld.", "Checking for crashed MySQL tables in the background.", "Visilearn:~# mysql -u root mysql", "Reading table information for completion of table and column names", "You can turn off this feature to get a quicker startup with -A", "Welcome to the MySQL monitor. Commands end with ; or \g.", "Your MySQL connection id is 4 to server version: 4.0.22-log", "Type 'help;' or '\h' for help. Type '\c' to clear the buffer.", "mysql> SET PASSWORD FOR root@localhost=PASSWORD('textbook');", "Query OK, 0 rows affected (0.04 sec)", "mysql>". A red oval highlights the command "mysql> SET PASSWORD FOR root@localhost=PASSWORD('textbook');" and its output "Query OK, 0 rows affected (0.04 sec)".

```
Shell - Konsole  
Session Edit View Bookmarks Settings Help  
Visilearn:~# /etc/init.d/mysql start  
Starting MySQL database server: mysqld.  
Checking for crashed MySQL tables in the background.  
Visilearn:~# mysql -u root mysql  
Reading table information for completion of table and column names  
You can turn off this feature to get a quicker startup with -A  
  
Welcome to the MySQL monitor. Commands end with ; or \g.  
Your MySQL connection id is 4 to server version: 4.0.22-log  
Type 'help;' or '\h' for help. Type '\c' to clear the buffer.  
mysql> SET PASSWORD FOR root@localhost=PASSWORD('textbook');  
Query OK, 0 rows affected (0.04 sec)  
mysql>
```

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.*

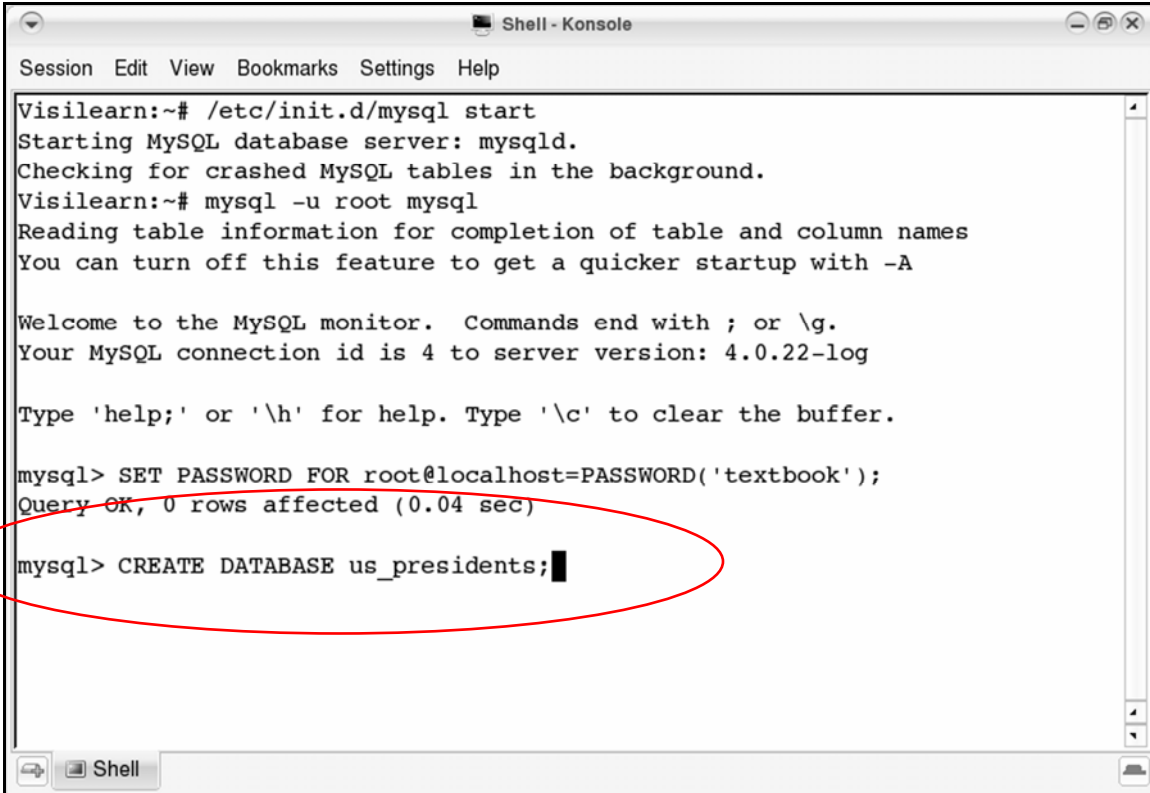
In the previous string of commands, you logged into the MySQL server as its root user, so the password `textbook` applies to the MySQL server.

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;
```

A screenshot of a terminal window titled "Shell - Konsole". The window shows the process of starting the MySQL database server. The user runs `/etc/init.d/mysql start`, which starts the MySQL server. Then, the user runs `mysql -u root mysql` to open the MySQL command-line interface. The prompt is `mysql>`. The user enters `SET PASSWORD FOR root@localhost=PASSWORD('textbook');` and receives the response "Query OK, 0 rows affected (0.04 sec)". The user then enters `CREATE DATABASE us_presidents;` at the prompt. This command is circled in red in the original image. The terminal window also shows a menu bar with "Session", "Edit", "View", "Bookmarks", "Settings", and "Help".

```
Visilearn:~# /etc/init.d/mysql start
Starting MySQL database server: mysqld.
Checking for crashed MySQL tables in the background.
Visilearn:~# mysql -u root mysql
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 4 to server version: 4.0.22-log

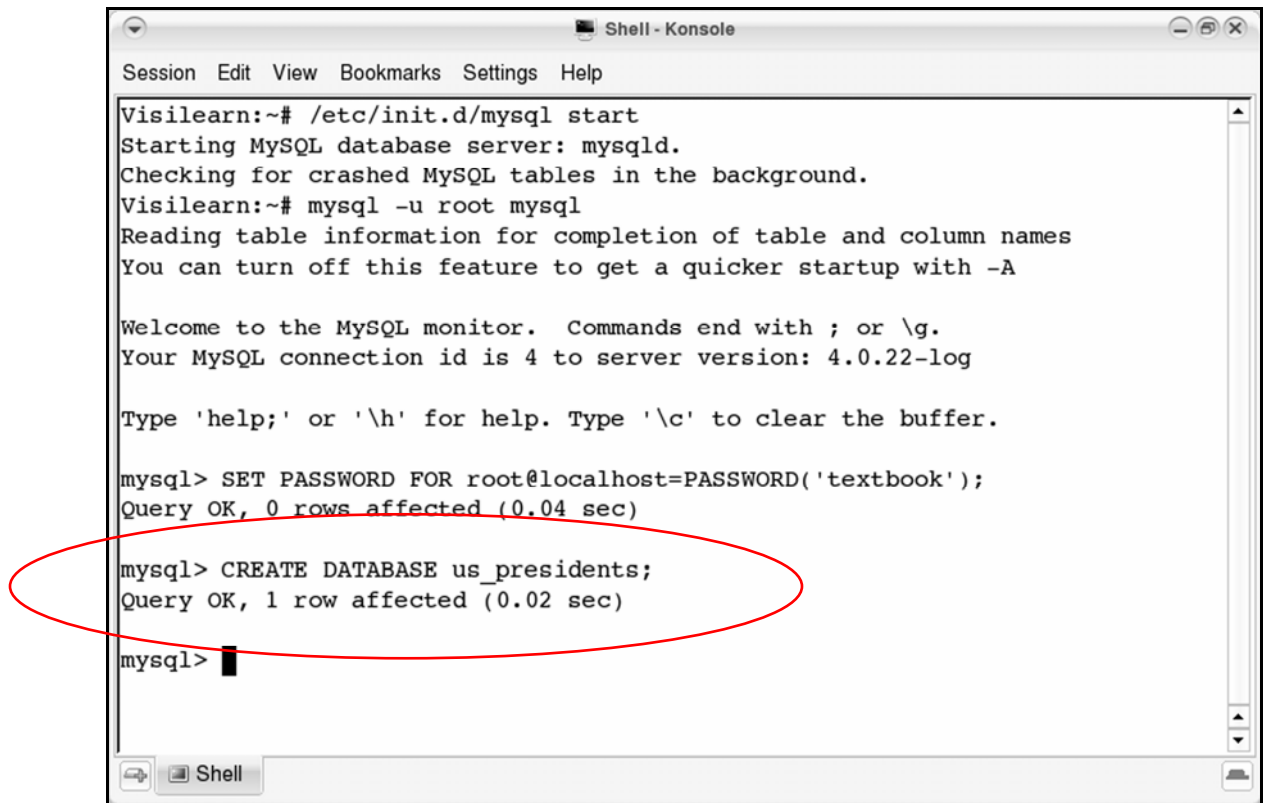
Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> SET PASSWORD FOR root@localhost=PASSWORD('textbook');
Query OK, 0 rows affected (0.04 sec)

mysql> CREATE DATABASE us_presidents;
```

Then press **ENTER**.

The window should look like this:



```
Shell - Konsole
Session Edit View Bookmarks Settings Help
Visilearn:~# /etc/init.d/mysql start
Starting MySQL database server: mysqld.
Checking for crashed MySQL tables in the background.
Visilearn:~# mysql -u root mysql
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 4 to server version: 4.0.22-log

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

mysql> SET PASSWORD FOR root@localhost=PASSWORD('textbook');
Query OK, 0 rows affected (0.04 sec)

mysql> CREATE DATABASE us_presidents;
Query OK, 1 row affected (0.02 sec)

mysql>
```

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 DATABASE us_presidents  
;
```

If the command is valid, it will execute.

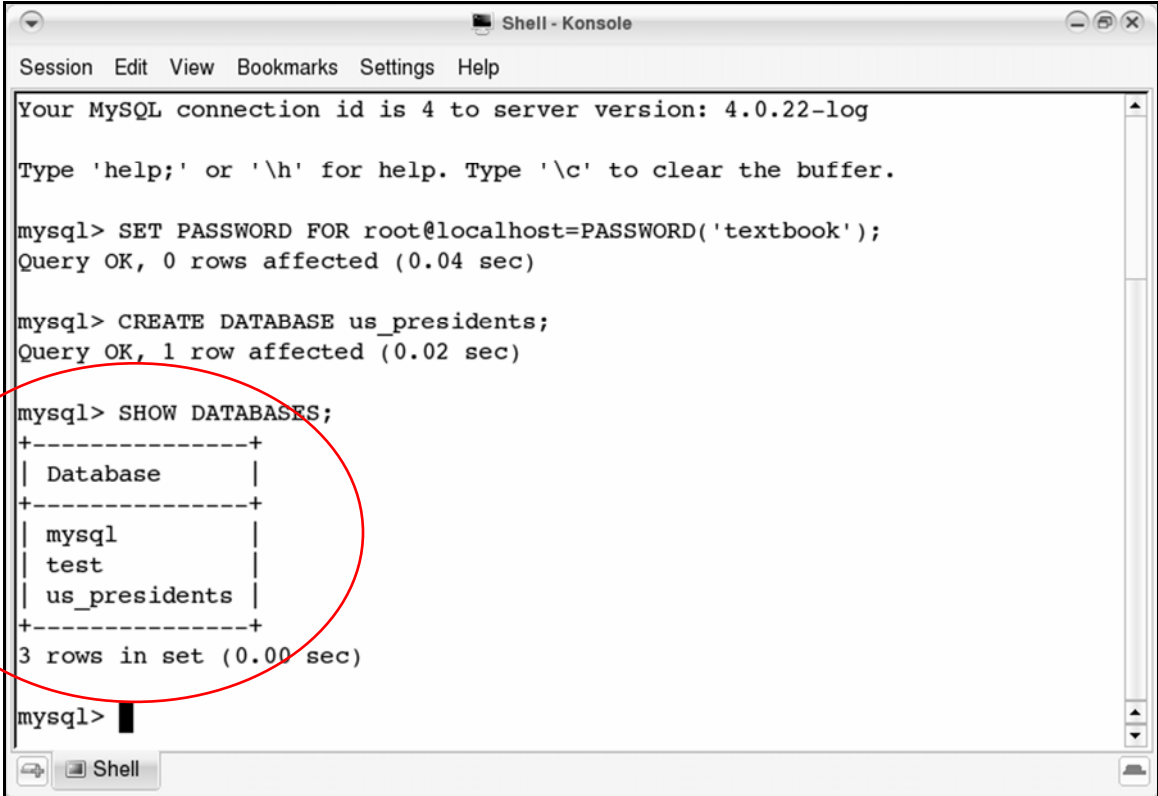
If there was an error in the command string and it's invalid, adding a semicolon here will execute it and MySQL will give an error.

2. Type:

```
SHOW DATABASES;
```

then press **ENTER**.

The window should look like this:



```
Shell - Konsole
Session Edit View Bookmarks Settings Help
Your MySQL connection id is 4 to server version: 4.0.22-log
Type 'help;' or '\h' for help. Type '\c' to clear the buffer.
mysql> SET PASSWORD FOR root@localhost=PASSWORD('textbook');
Query OK, 0 rows affected (0.04 sec)
mysql> CREATE DATABASE us_presidents;
Query OK, 1 row affected (0.02 sec)
mysql> SHOW DATABASES;
+-----+
| Database |
+-----+
| mysql    |
| test     |
| us_presidents |
+-----+
3 rows in set (0.00 sec)
mysql>
```

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

The `mysql` database is 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.

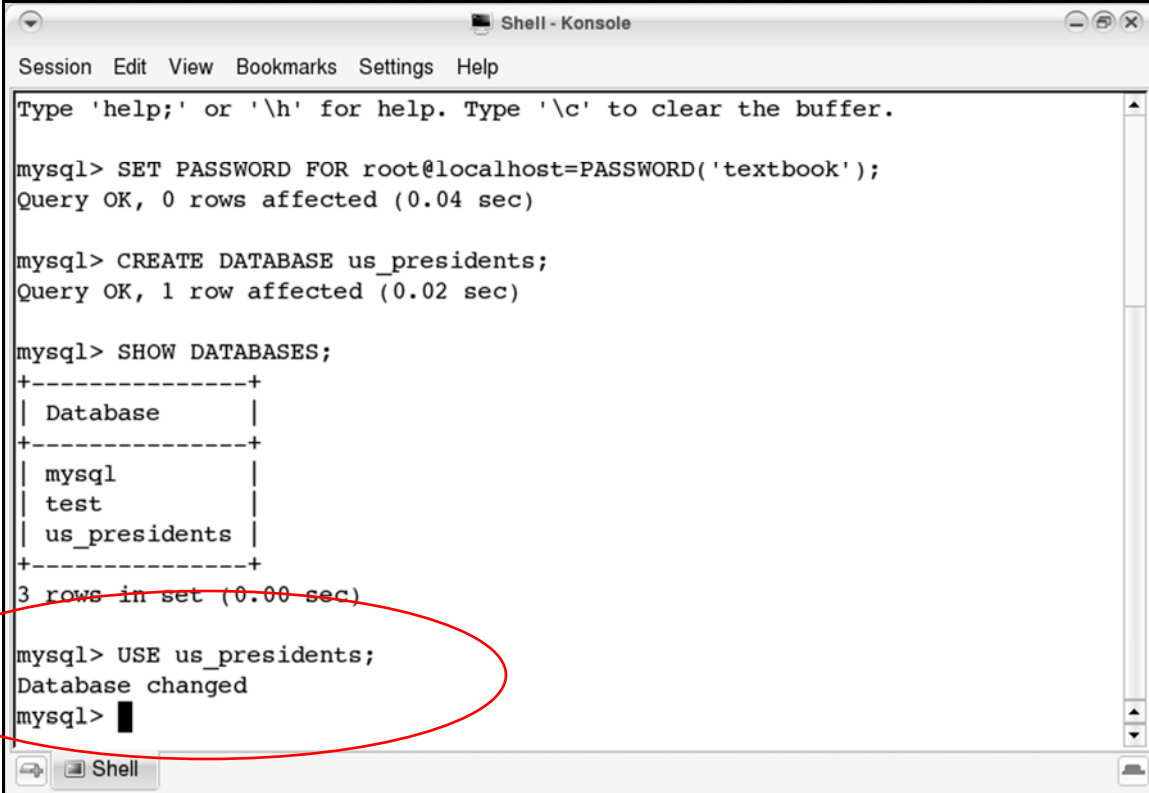
Create a table

1. Type:

```
USE us_presidents;
```

then press **ENTER**.

The window should look like this:



```
Shell - Konsole
Session Edit View Bookmarks Settings Help
Type 'help;' or '\h' for help. Type '\c' to clear the buffer.
mysql> SET PASSWORD FOR root@localhost=PASSWORD('textbook');
Query OK, 0 rows affected (0.04 sec)
mysql> CREATE DATABASE us_presidents;
Query OK, 1 row affected (0.02 sec)
mysql> SHOW DATABASES;
+-----+
| Database |
+-----+
| mysql    |
| test     |
| us_presidents |
+-----+
3 rows in set (0.00 sec)
mysql> USE us_presidents;
Database changed
mysql>
```

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:

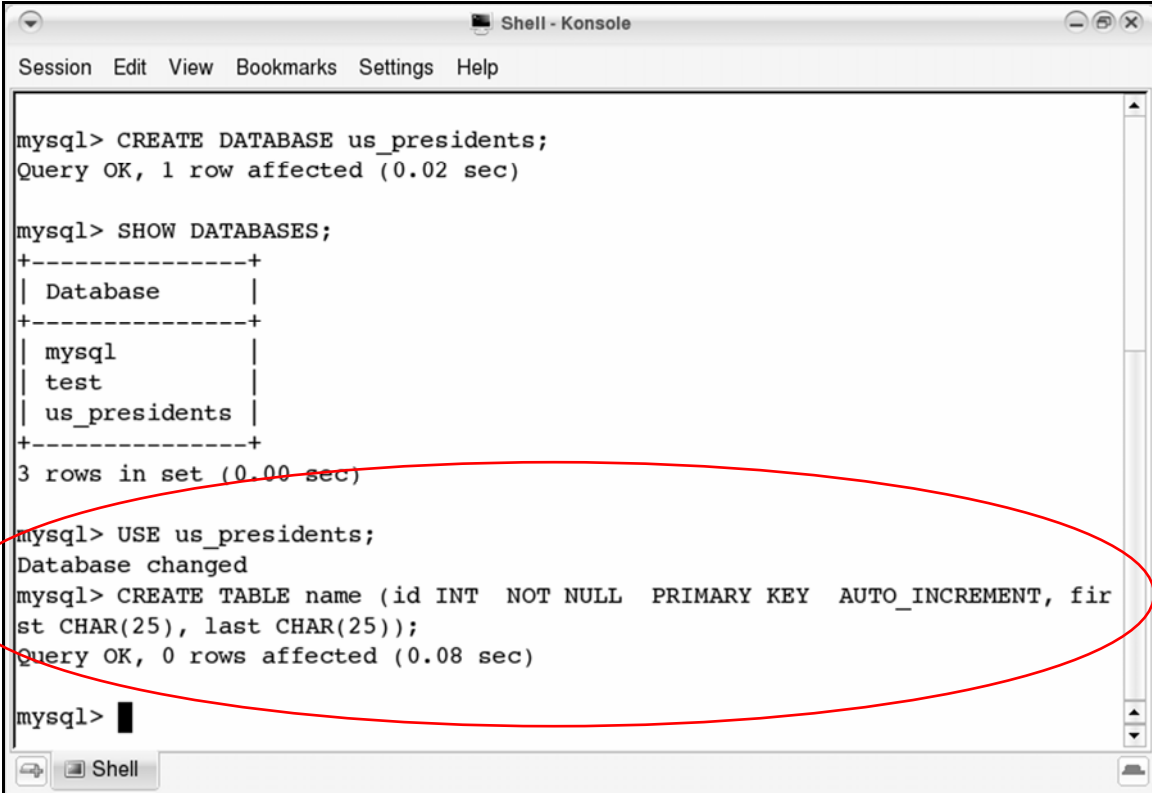
```
rpm -i MySQL-3.23.51-1.i386.rpm ►►  
MySQL-client-3.23.51-1.i386.rpm
```

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:



```
mysql> CREATE DATABASE us_presidents;
Query OK, 1 row affected (0.02 sec)

mysql> SHOW DATABASES;
+-----+
| Database |
+-----+
| mysql    |
| test     |
| us_presidents |
+-----+
3 rows in set (0.00 sec)

mysql> USE us_presidents;
Database changed
mysql> CREATE TABLE name (id INT NOT NULL PRIMARY KEY AUTO_INCREMENT, first CHAR(25), last CHAR(25));
Query OK, 0 rows affected (0.08 sec)

mysql>
```

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)	

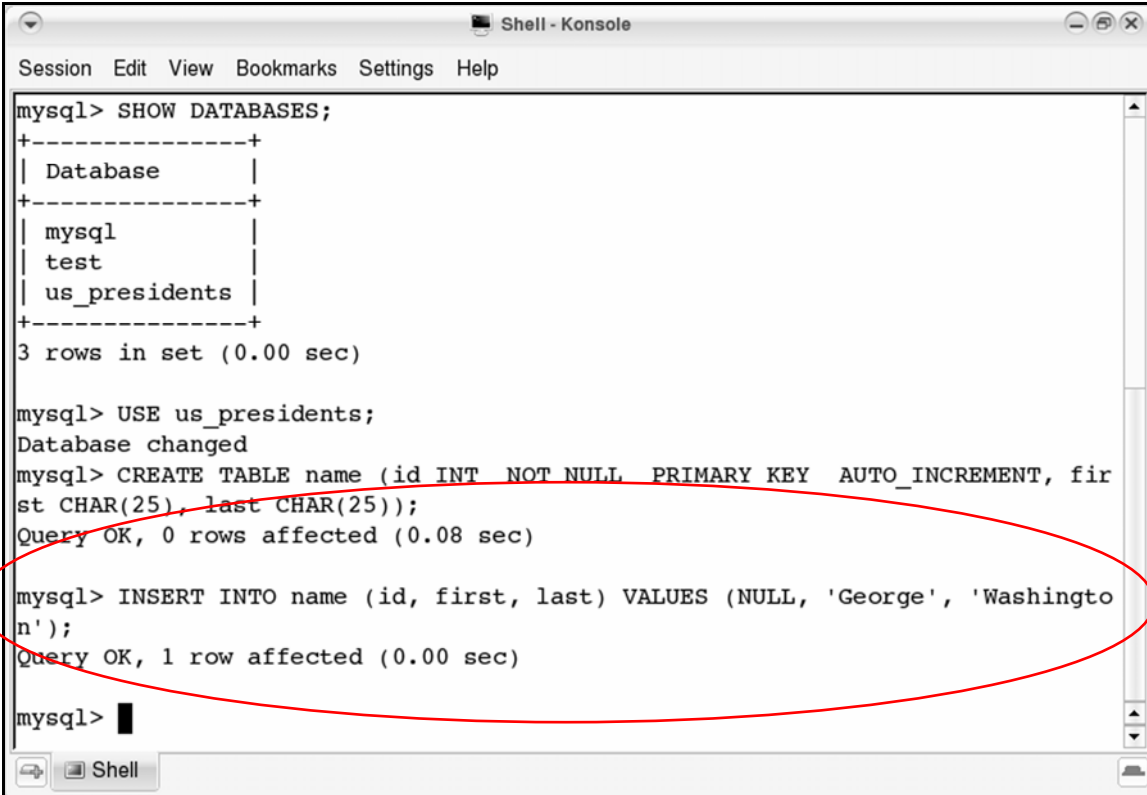
Create a record

1. Type:

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

then press **ENTER**.

The window should look like this:



```
mysql> SHOW DATABASES;  
+-----+  
| Database |  
+-----+  
| mysql   |  
| test    |  
| us_presidents |  
+-----+  
3 rows in set (0.00 sec)  
  
mysql> USE us_presidents;  
Database changed  
mysql> CREATE TABLE name (id INT NOT NULL PRIMARY KEY AUTO_INCREMENT, first  
CHAR(25), last CHAR(25));  
Query OK, 0 rows affected (0.08 sec)  
  
mysql> INSERT INTO name (id, first, last) VALUES (NULL, 'George', 'Washington');  
Query OK, 1 row affected (0.00 sec)  
  
mysql> █
```

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

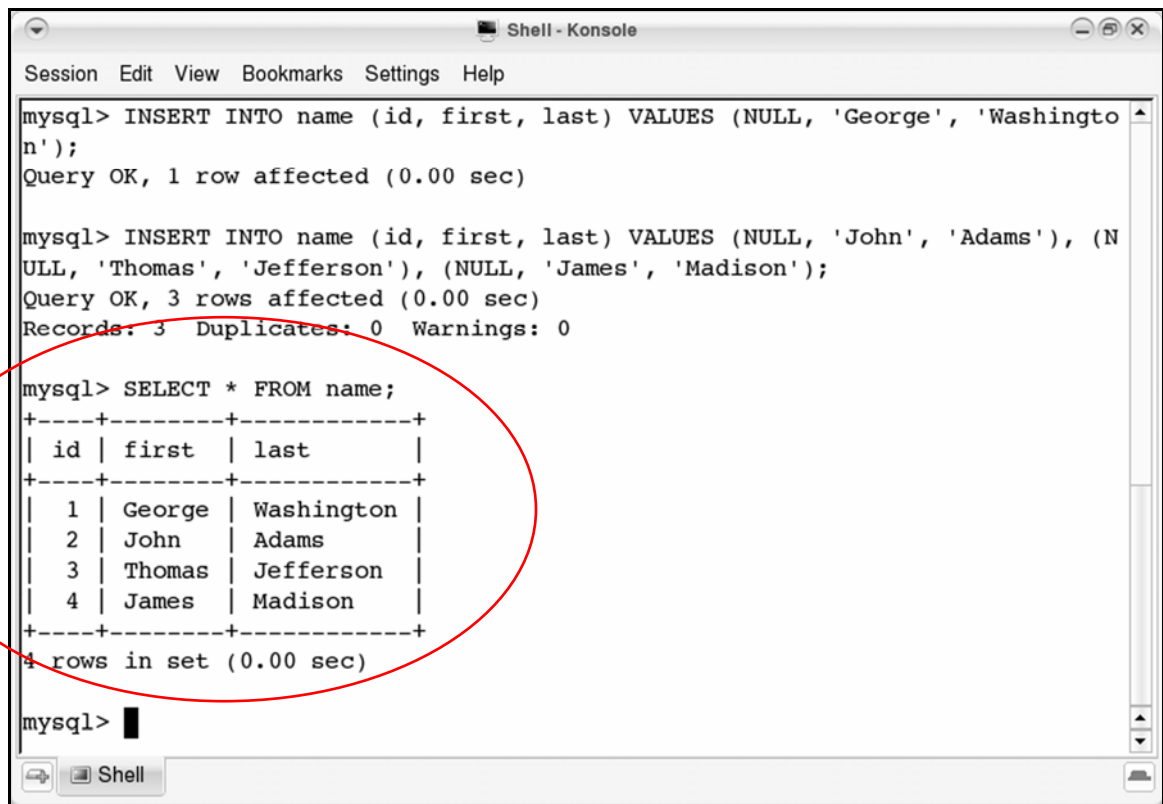
Run a query

1. Type:

```
SELECT * FROM name;
```

then press **ENTER**.

The window should look like this:



```
mysql> INSERT INTO name (id, first, last) VALUES (NULL, 'George', 'Washington');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO name (id, first, last) VALUES (NULL, 'John', 'Adams'), (NULL, 'Thomas', 'Jefferson'), (NULL, 'James', 'Madison');
Query OK, 3 rows affected (0.00 sec)
Records: 3 Duplicates: 0 Warnings: 0

mysql> SELECT * FROM name;
+----+-----+-----+
| id | first | last  |
+----+-----+-----+
| 1  | George | Washington |
| 2  | John  | Adams  |
| 3  | Thomas | Jefferson |
| 4  | James | Madison |
+----+-----+-----+
4 rows in set (0.00 sec)

mysql>
```

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

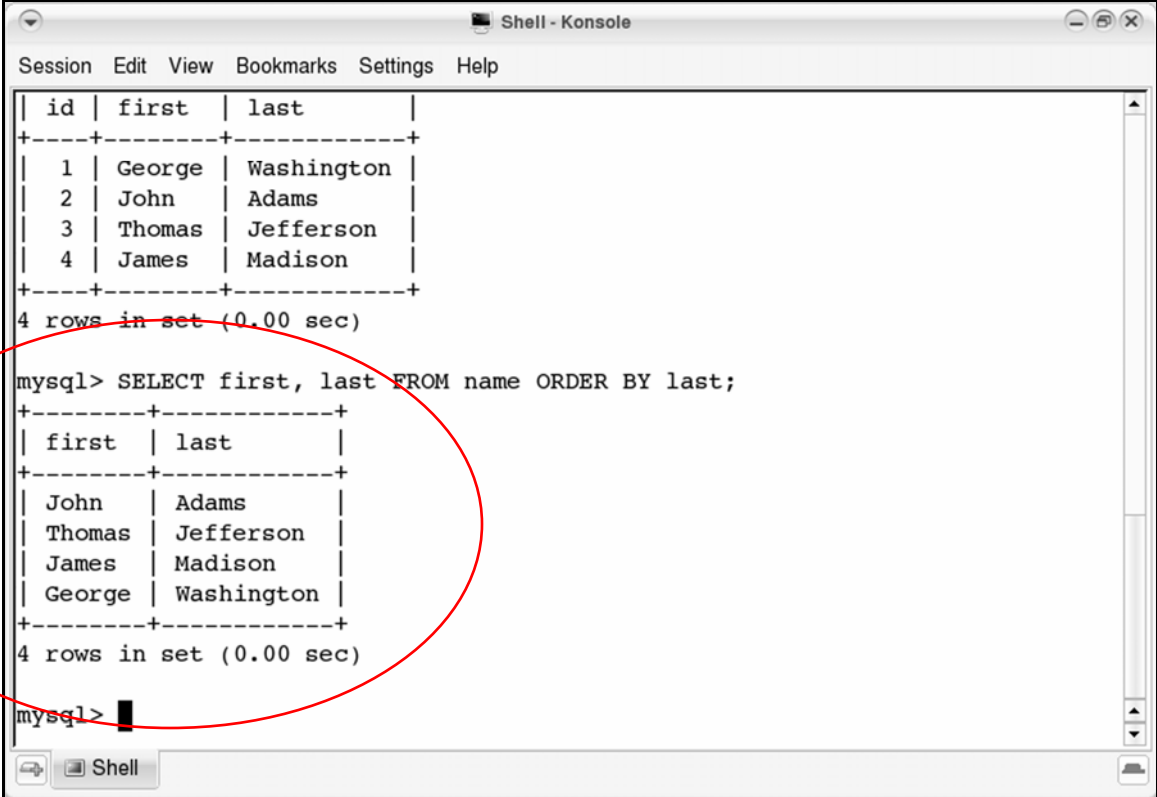
The asterisk (*) command tells MySQL to return everything (the asterisk means “everything” or “all”) that’s in the table **name**.

2. Type:

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

then press **ENTER**.

The window should look like this:



```
Session Edit View Bookmarks Settings Help  
+----+-----+-----+  
| id | first | last |  
+----+-----+-----+  
| 1 | George | Washington |  
| 2 | John | Adams |  
| 3 | Thomas | Jefferson |  
| 4 | James | Madison |  
+----+-----+-----+  
4 rows in set (0.00 sec)  
  
mysql> SELECT first, last FROM name ORDER BY last;  
+----+-----+  
| first | last |  
+----+-----+  
| John | Adams |  
| Thomas | Jefferson |  
| James | Madison |  
| George | Washington |  
+----+-----+  
4 rows in set (0.00 sec)  
  
mysql> █
```

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

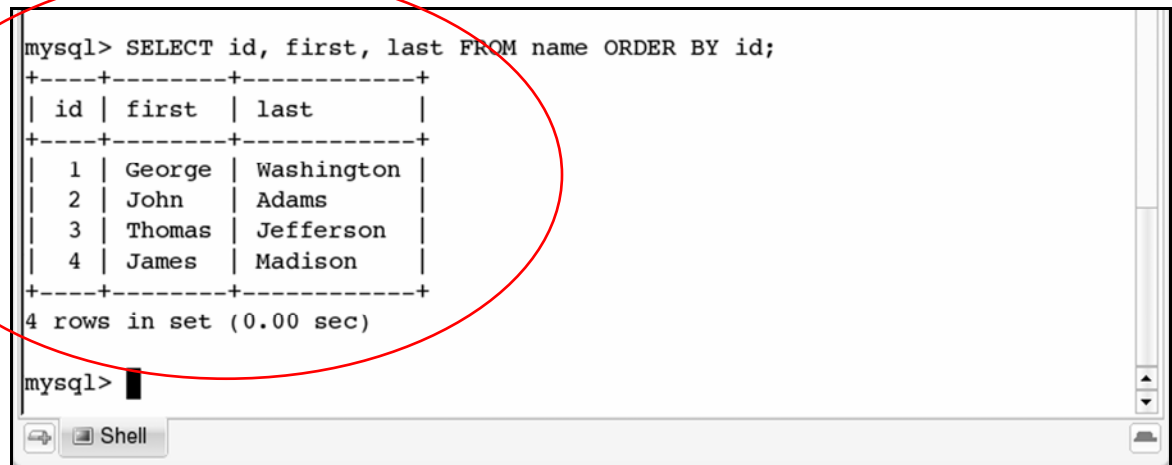
`ORDER BY` puts the records in alphabetical order, based on the field `last`. In other words, it puts the presidents' last names in alphabetical order.

3. Type:

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

then press **ENTER**.

The window should look like this:



```
mysql> SELECT id, first, last FROM name ORDER BY id;  
+----+-----+-----+  
| id | first | last  |  
+----+-----+-----+  
|  1 | George | Washington |  
|  2 | John  | Adams   |  
|  3 | Thomas | Jefferson |  
|  4 | James | Madison |  
+----+-----+-----+  
4 rows in set (0.00 sec)  
  
mysql> |
```

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.*

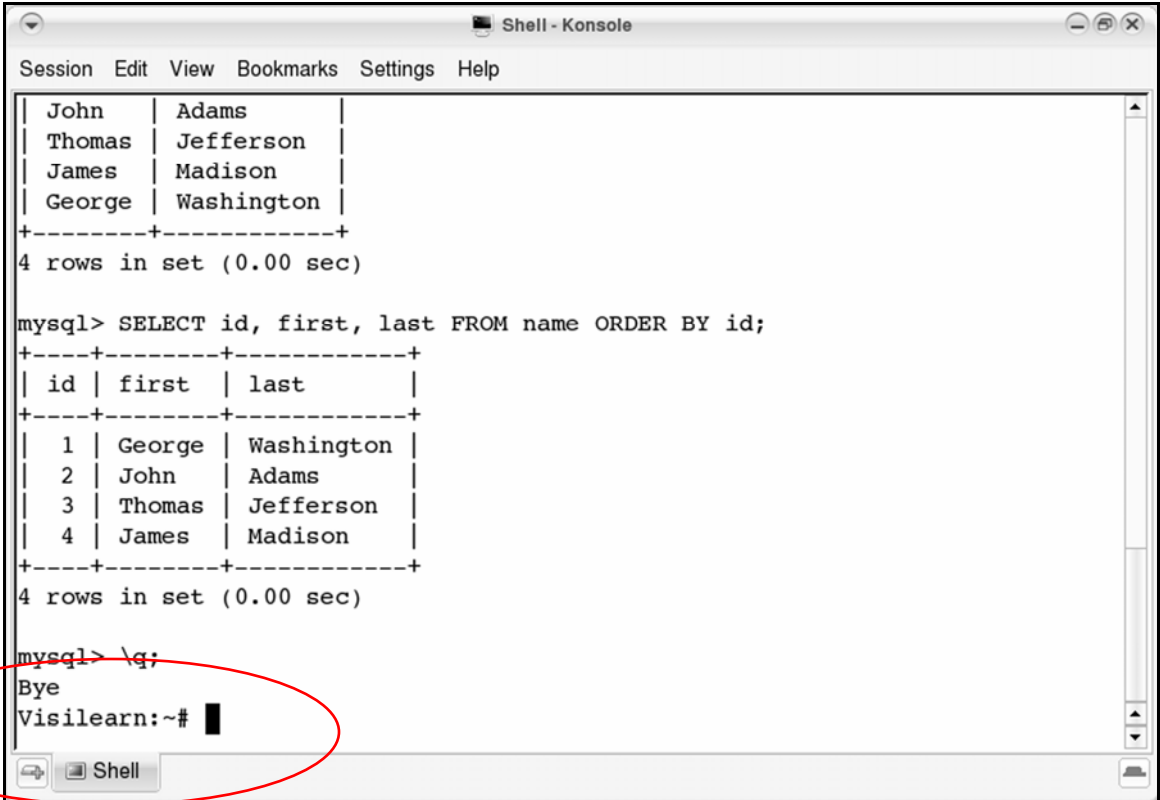
4. Type:

```
\q;
```

then press **ENTER**.

This closes your MySQL database connection.

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



```
Session Edit View Bookmarks Settings Help

| John | Adams |
| Thomas | Jefferson |
| James | Madison |
| George | Washington |
+-----+-----+
4 rows in set (0.00 sec)

mysql> SELECT id, first, last FROM name ORDER BY id;
+-----+-----+
| id | first | last |
+-----+-----+
| 1 | George | Washington |
| 2 | John | Adams |
| 3 | Thomas | Jefferson |
| 4 | James | Madison |
+-----+-----+
4 rows in set (0.00 sec)

mysql> \q;
Bye
Visilearn:~#
```

5. Type:

```
exit
```

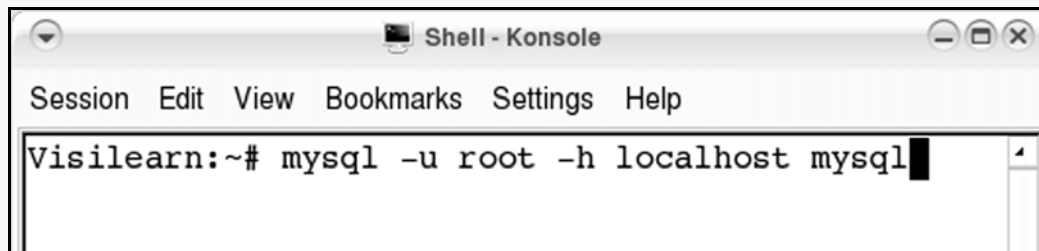
then press **ENTER**.

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`.

A screenshot of a terminal window titled "Shell - Konsole". The window has a menu bar with "Session", "Edit", "View", "Bookmarks", "Settings", and "Help". The terminal prompt is "Visilearn:~#" and the command entered is "mysql -u root -h localhost mysql".

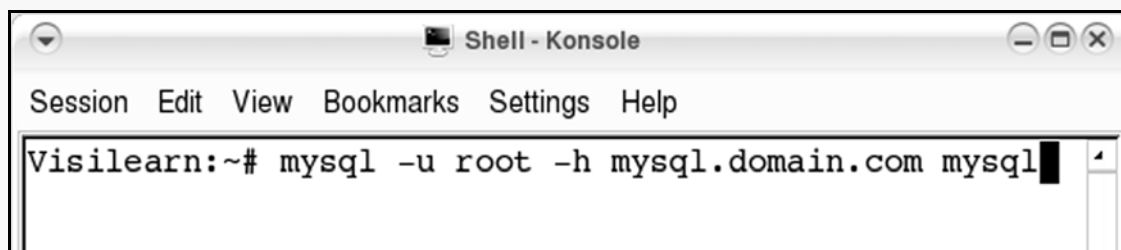
```
Shell - Konsole
Session Edit View Bookmarks Settings Help
Visilearn:~# mysql -u root -h localhost mysql
```

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`

A screenshot of a terminal window titled "Shell - Konsole". The window has a menu bar with "Session", "Edit", "View", "Bookmarks", "Settings", and "Help". The terminal prompt is "Visilearn:~#" and the command entered is "mysql -u root -h mysql.domain.com mysql".

```
Shell - Konsole
Session Edit View Bookmarks Settings Help
Visilearn:~# mysql -u root -h mysql.domain.com mysql
```

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

Practice: Getting Started

Task: A University has been giving computers to students without keeping track of who has what. Create a database for recording the computers given to students.

1. Open the **Konsole** window.

2. At the prompt, type:

```
mysql -u root -p
```

then press **ENTER**.

Tip: *The `-p` command tells MySQL to prompt the user for a password.*

3. Type the password used to gain root access to the MySQL server:

```
textbook
```

then press **ENTER**.

You are now logged in to the MySQL server.

4. Create a new database called `hardware`.

- 5.** Create three new tables in the `hardware` database, organized in the format shown below:

Table: `student`

Field	Datatype	Properties
<code>id</code>	<code>INT</code>	primary key, not null, auto increment
<code>first_name</code>	<code>CHAR(15)</code>	
<code>last_name</code>	<code>CHAR(25)</code>	

Table: `computer`

Field	Datatype	Properties
<code>id</code>	<code>INT</code>	primary key, not null, auto increment
<code>description</code>	<code>CHAR(35)</code>	

Table: `history`

Field	Datatype	Properties
<code>id</code>	<code>INT</code>	primary key, not null, auto increment
<code>date_added</code>	<code>DATE</code>	
<code>student_id</code>	<code>INT</code>	
<code>computer_id</code>	<code>INT</code>	
<code>comments</code>	<code>CHAR(50)</code>	

6. Insert these data into the table `student`:

id	first_name	last_name
1	Jack	Hanson
2	Lon	James
3	Ken	Jones

7. Insert these data into the table `computer`:

id	description
1	Apple iBook
2	Apple PowerBook
3	Apple iMac

8. Insert these data into the table `history`:

id	date_added	student_id	computer_id	comments
1	2002-01-08	1	2	Cool new laptop
2	2002-01-09	1	3	Workstation
3	2002-01-14	2	1	Wireless web

Tip: *When inserting dates, use the YYYY-MM-DD format, where Y's are the year, M's the month, and D's the day. If you don't, MySQL will not properly store the information.*

Also, treat the date like text by surrounding it with quote marks. Otherwise, MySQL may think that 2002-01-08 is 2002 minus 1 minus 8, or 1993.

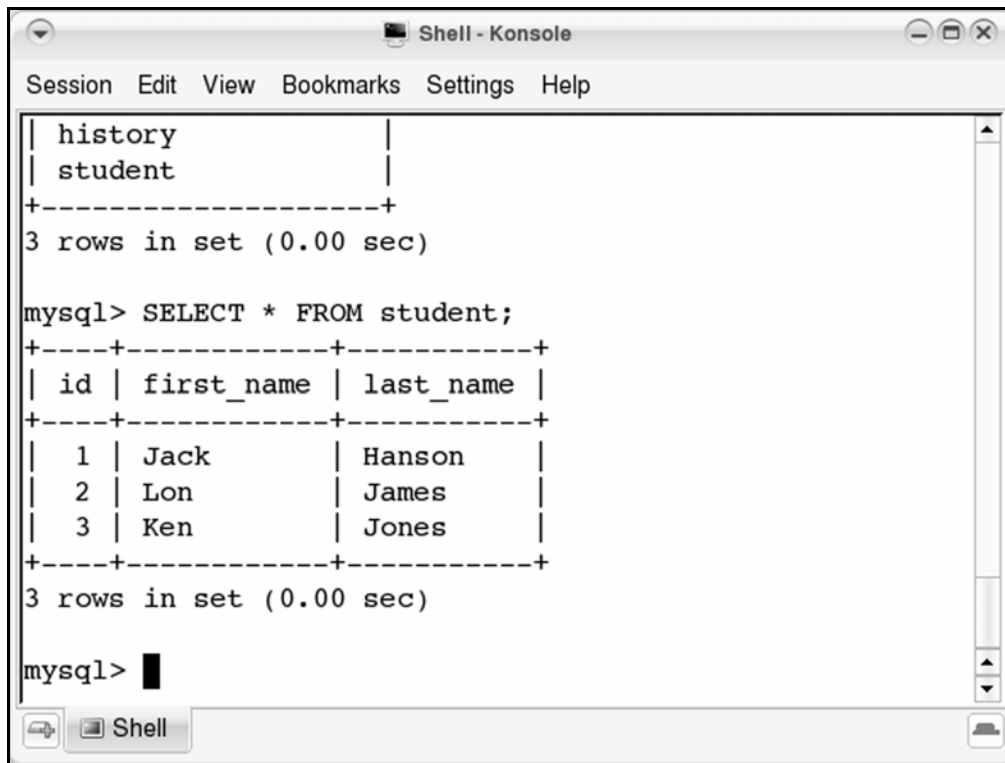
A command to insert a date would look like this:

```
INSERT INTO birthdays ►►  
(name, birthday) ►►  
VALUES ('Kevin', '1975-11-18');
```

- 9.** Check your work by displaying the contents of the `student` table.

Tip: *Use a query to show everything in the table.*

When you're done, the window should look like this:



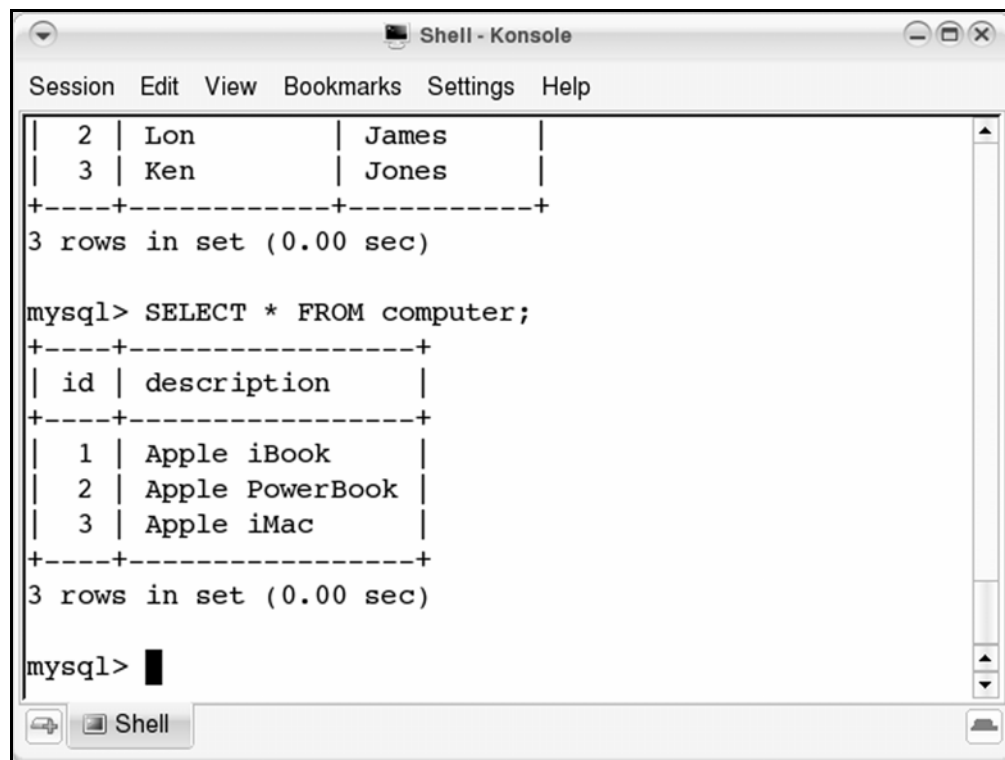
The screenshot shows a terminal window titled "Shell - Konsole" with a menu bar containing "Session", "Edit", "View", "Bookmarks", "Settings", and "Help". The terminal displays the following text:

```
| history |  
| student |  
+-----+  
3 rows in set (0.00 sec)  
  
mysql> SELECT * FROM student;  
+----+-----+-----+  
| id | first_name | last_name |  
+----+-----+-----+  
| 1 | Jack      | Hanson   |  
| 2 | Lon       | James    |  
| 3 | Ken       | Jones    |  
+----+-----+-----+  
3 rows in set (0.00 sec)  
  
mysql> █
```

The terminal window also shows a "Shell" tab at the bottom left and standard window controls (minimize, maximize, close) at the top right.

10. Display the contents of the `computer` table.

The window should look like this:



```
Session Edit View Bookmarks Settings Help
| 2 | Lon      | James  |
| 3 | Ken      | Jones  |
+---+-----+-----+
3 rows in set (0.00 sec)

mysql> SELECT * FROM computer;
+---+-----+
| id | description |
+---+-----+
| 1  | Apple iBook |
| 2  | Apple PowerBook |
| 3  | Apple iMac  |
+---+-----+
3 rows in set (0.00 sec)

mysql> █
```

11. Close the MySQL database connection.

12. Exit the **Konsole** window.

Administering Databases

In this section, you'll learn how to:

- **Restart MySQL**
- **Back up a database**
- **Delete a table**
- **Delete a database**
- **Restore a database**

Restart MySQL

If you've shut off your computer since the last exercise, you might need to restart MySQL.

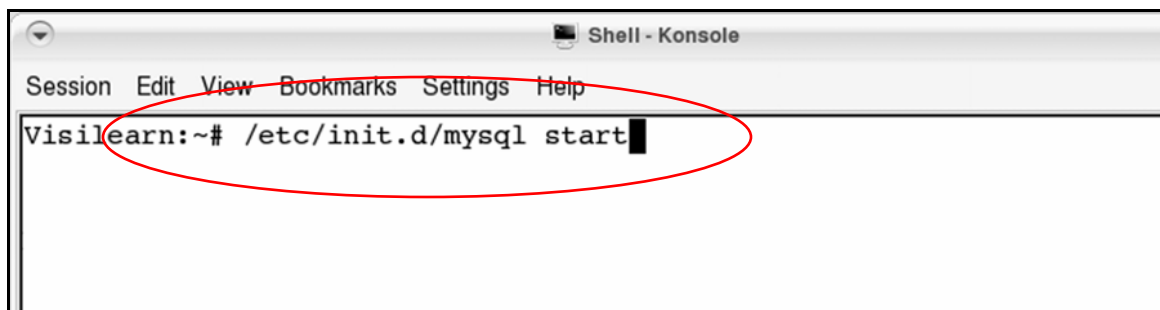
First, login to your Linux computer as the Root user. Then restart the MySQL server:

1. Open the **Konsole** window.

Tip: *If your terminal prompt is followed by a \$, login as the Root user. Type **su** and press **ENTER**. Type your Root password and press **ENTER** again.*

2. At the prompt, type:

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



then press **ENTER**.

Tip: *If you had to login as the Root user in step 1, type:*

```
exit
```

*then press **ENTER**.*

You're now logged out of the Root account.

```
+ yourusername@localhost: /home/yourusername - Shell - Konsole
Session Edit View Bookmarks Settings Help
[yourusername@localhost yourusername]$ su
Password:
[root@localhost yourusername]# /etc/rc.d/init.d/mys
Starting MySQL Server
  OK
[root@localhost yourusername]# exit
exit
[yourusername@localhost yourusername]$ █
```

Now you'll have to establish a MySQL client connection to the MySQL server:

3. At the prompt, type:

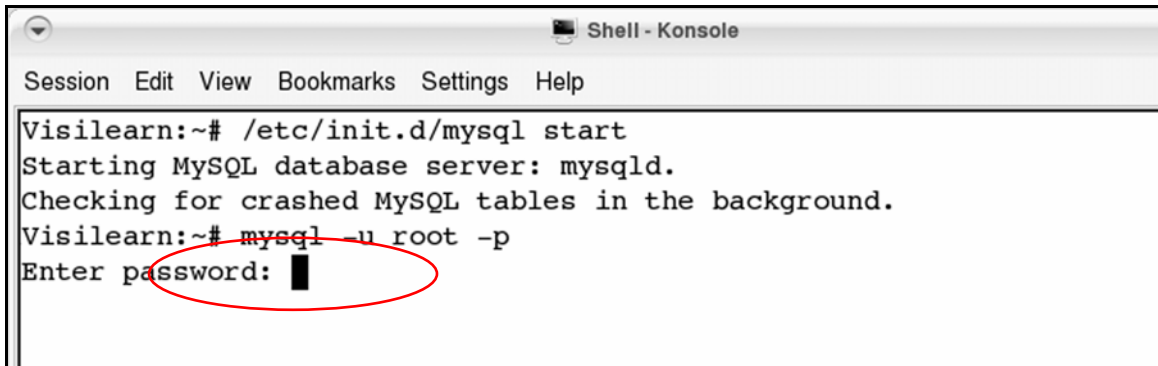
```
mysql -u root -p
```

```
Shell - Konsole
Session Edit View Bookmarks Settings Help
Visilearn:~# /etc/init.d/mysql start
Starting MySQL database server: mysqld.
Checking for crashed MySQL tables in the background.
Visilearn:~# mysql -u root -p█
```

then press **ENTER**.

4. Type the password used to gain root access to the MySQL server:

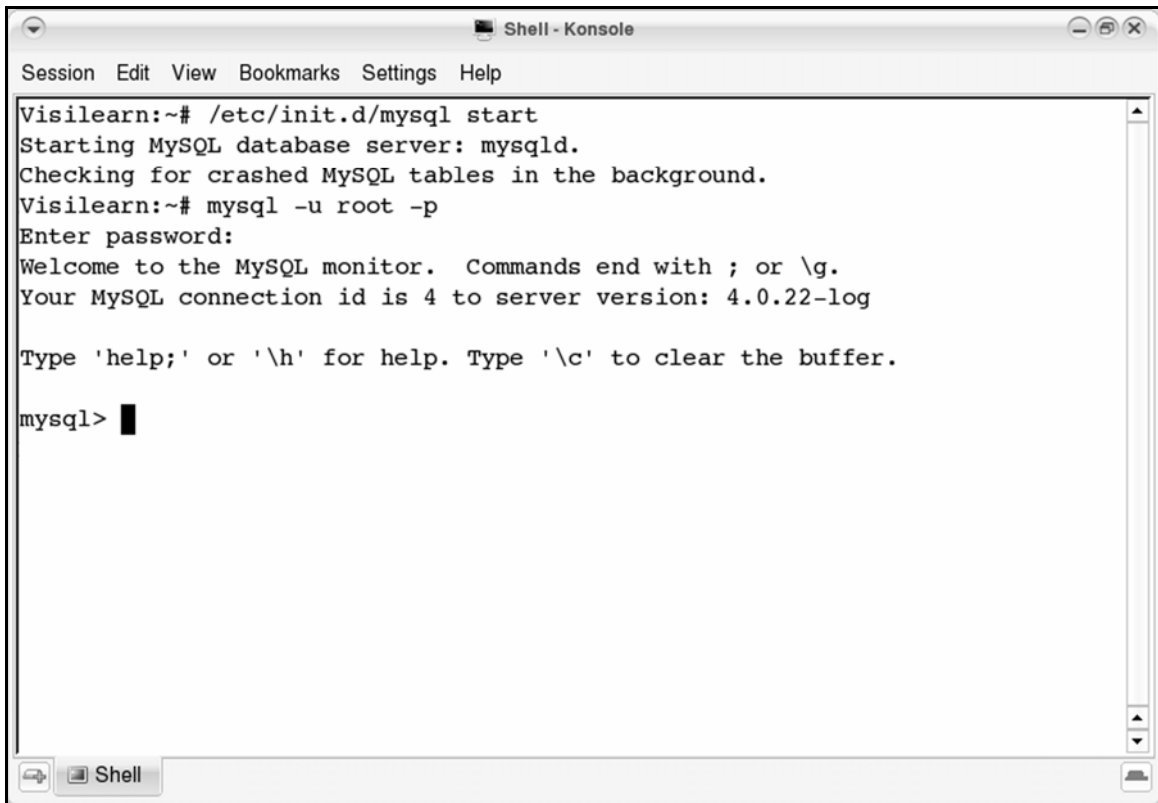
textbook



```
Shell - Konsole
Session Edit View Bookmarks Settings Help
Visilearn:~# /etc/init.d/mysql start
Starting MySQL database server: mysqld.
Checking for crashed MySQL tables in the background.
Visilearn:~# mysql -u root -p
Enter password: █
```

then press **ENTER**.

The window should look like this:



```
Shell - Konsole
Session Edit View Bookmarks Settings Help
Visilearn:~# /etc/init.d/mysql start
Starting MySQL database server: mysqld.
Checking for crashed MySQL tables in the background.
Visilearn:~# mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 4 to server version: 4.0.22-log

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

mysql> █
```

Back up a database

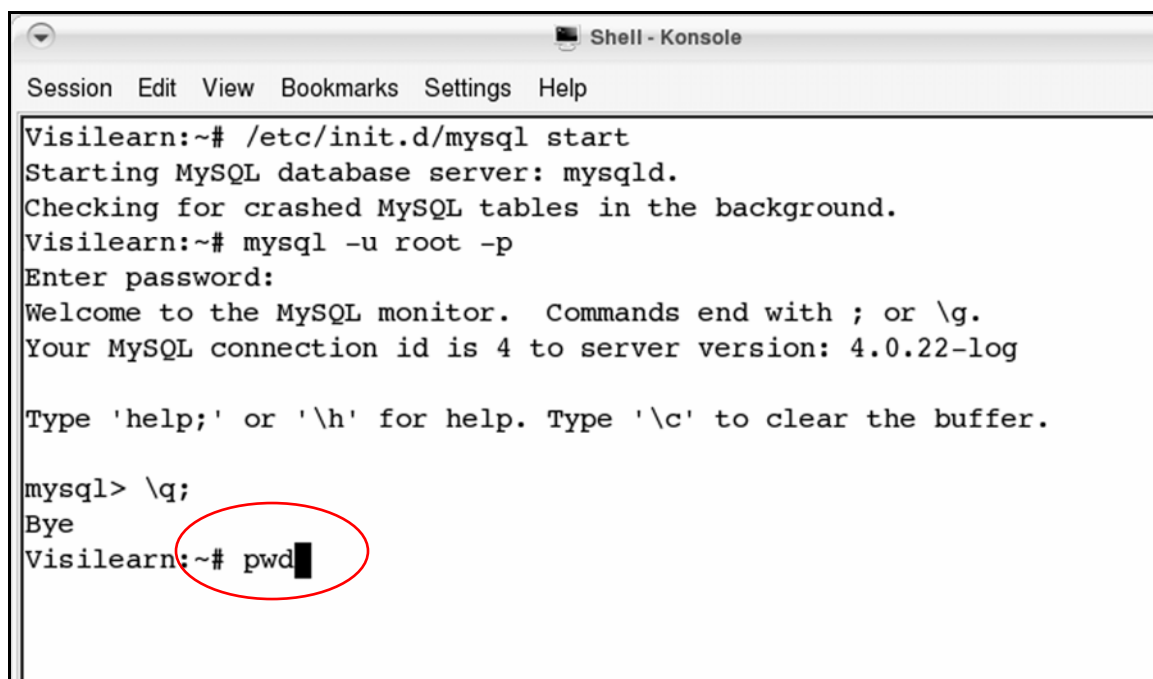
1. Make sure the **Konsole** window is open. If it's not, open it.

2. Make sure you're logged out of the MySQL server.

Tip: Give the `\q;` command.

3. At the prompt, type:

`pwd`



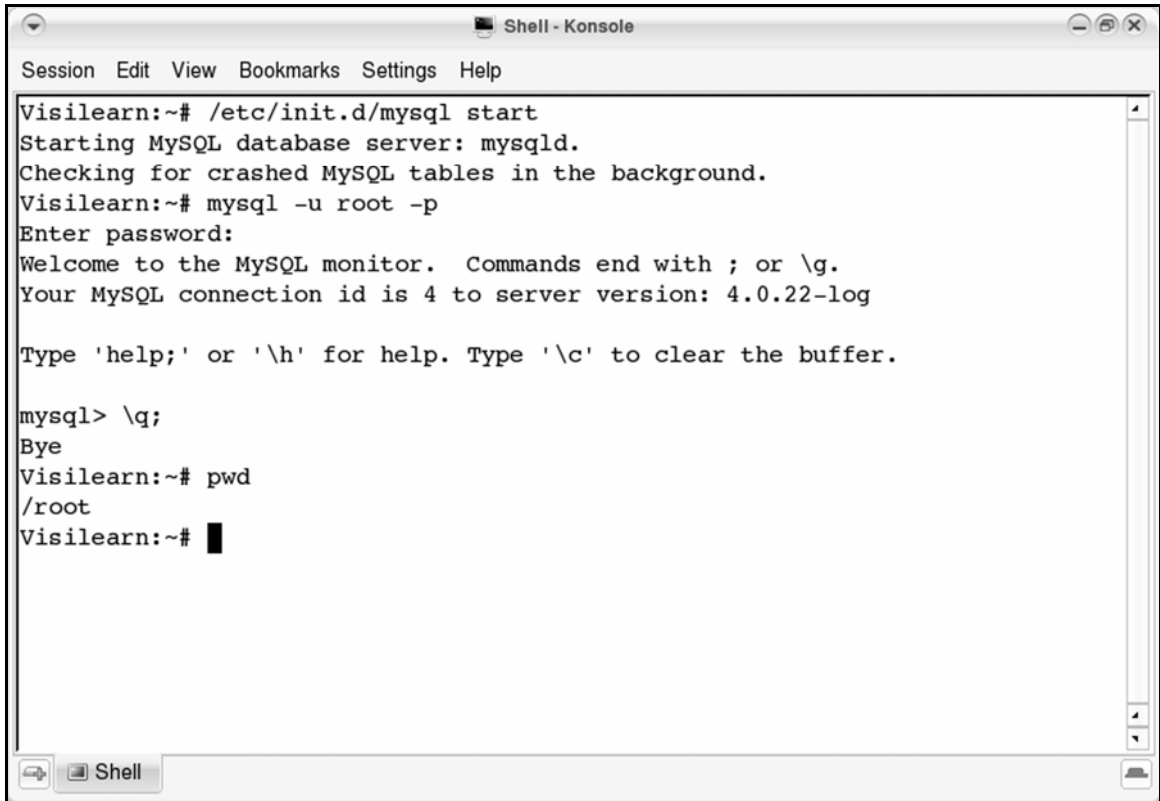
```
Shell - Konsole
Session Edit View Bookmarks Settings Help
Visilearn:~# /etc/init.d/mysql start
Starting MySQL database server: mysqld.
Checking for crashed MySQL tables in the background.
Visilearn:~# mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 4 to server version: 4.0.22-log

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

mysql> \q;
Bye
Visilearn:~# pwd
```

then press **ENTER**.

4. The window should look something like this:



```
Shell - Konsole
Session Edit View Bookmarks Settings Help
Visilearn:~# /etc/init.d/mysql start
Starting MySQL database server: mysqld.
Checking for crashed MySQL tables in the background.
Visilearn:~# mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 4 to server version: 4.0.22-log

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

mysql> \q;
Bye
Visilearn:~# pwd
/root
Visilearn:~#
```

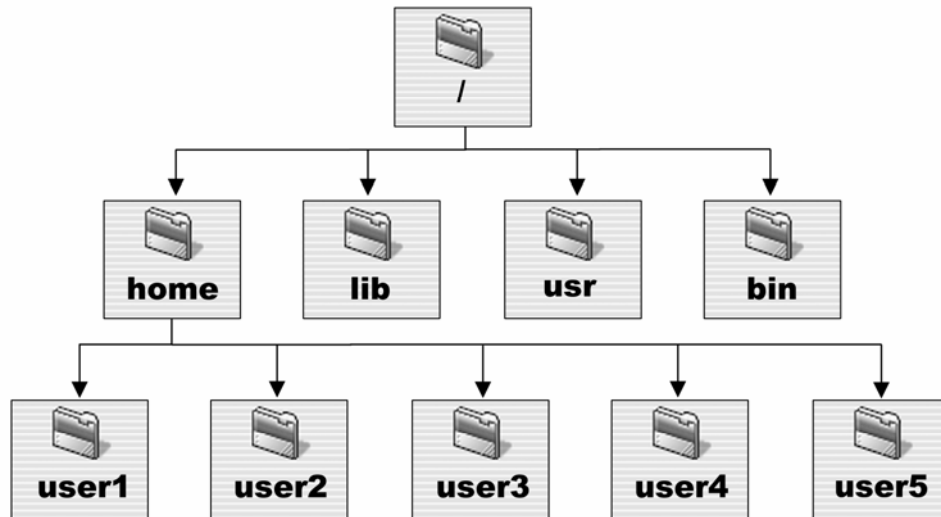
The Linux command `pwd` is an acronym for print working directory. In other words, “print the path to the directory I’m working in.”

This is the path to your current working directory on this computer: `root`.

When you first open the **Konsole** window, Linux automatically goes to your **home** directory. (Linspire’s default user is the Root user, so the current working directory, `root`, is actually the root user’s home directory.)

Each user on a Linux computer has his own **home** directory, which contains preferences and files unique to that user.

Tip: *The Linux file system is structured like a pyramid, with the Root directory at the top.*



Starting from the Root directory, you can dig down into all the other directories, or folders, on the computer.

5. Type:

```
mkdir backups
```

then press **ENTER**.

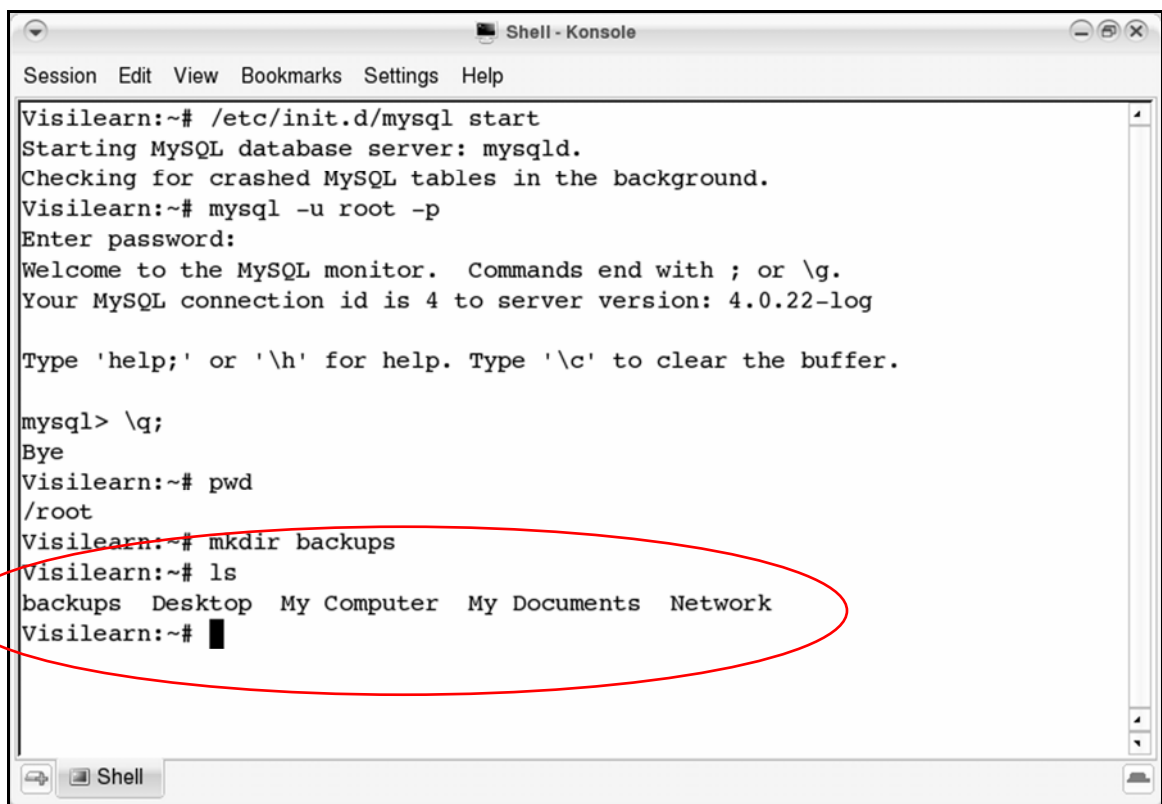
`mkdir` is a Linux command to create a new directory, in this case a new directory within your home directory called `backups`.

6. Type:

```
ls
```

then press **ENTER**.

The `ls` command lists all the items in the current directory: the `backups`, `Desktop`, `My Computer`, `My Documents`, and `Network` directories.

A screenshot of a terminal window titled "Shell - Konsole". The window contains the following text:

```
Session Edit View Bookmarks Settings Help
Visilearn:~# /etc/init.d/mysql start
Starting MySQL database server: mysqld.
Checking for crashed MySQL tables in the background.
Visilearn:~# mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 4 to server version: 4.0.22-log

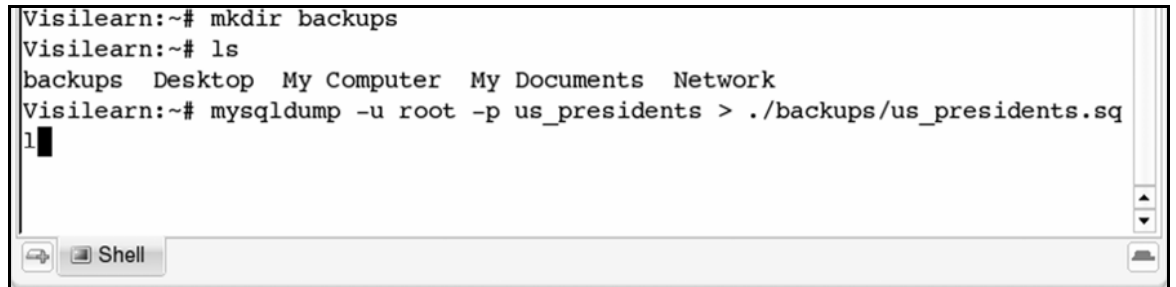
Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> \q;
Bye
Visilearn:~# pwd
/root
Visilearn:~# mkdir backups
Visilearn:~# ls
backups Desktop My Computer My Documents Network
Visilearn:~# █
```

The last two lines of the terminal output, "Visilearn:~# ls" and "backups Desktop My Computer My Documents Network", are circled in red.

7. Type:

```
mysqldump -u root -p us_presidents > >>  
./backups/us_presidents.sql
```

A screenshot of a terminal window titled "Shell". The terminal shows the following commands and output:

```
Visilearn:~# mkdir backups  
Visilearn:~# ls  
backups Desktop My Computer My Documents Network  
Visilearn:~# mysqldump -u root -p us_presidents > ./backups/us_presidents.sq  
1
```

then press **ENTER**.

Here's an explanation of this command string:

- **mysqldump**

```
mysqldump -u root -p us_presidents >  
./backups/us_presidents.sql
```

The **mysqldump** command does exactly what it says – it connects to the MySQL server, selects a database, then dumps all the information from it into a text file.

- **-u root -p**

```
mysqldump -u root -p us_presidents >  
./backups/us_presidents.sql
```

The **-u** command tells **mysqldump** to use the MySQL root user account to connect to the MySQL server.

The **-p** command tells MySQL to prompt the user for a password.

- **us_presidents**

```
mysqldump -u root -p us_presidents >  
./backups/us_presidents.sql
```

us_presidents is the name of the database you want to back up.

- **>**

```
mysqldump -u root -p us_presidents >  
./backups/us_presidents.sql
```

The **>** character is called a “pipe,” and is a Linux command. Pipe is an apt name for what **>** does: it pipes, or places, the information provided by **mysqldump** into a file.

- **./backups/**

```
mysqldump -u root -p us_presidents >  
./backups/us_presidents.sql
```

./backups/ is the directory path to **us_presidents.sql**.

Tip: *The period in front of the slash (./) represents the current directory you are working in.*

- **us_presidents.sql**

```
mysqldump -u root -p us_presidents >  
./backups/us_presidents.sql
```

us_presidents.sql is the name of the file you’re piping the backup to.

8. At the password prompt, type:

```
textbook
```

then press **ENTER**.

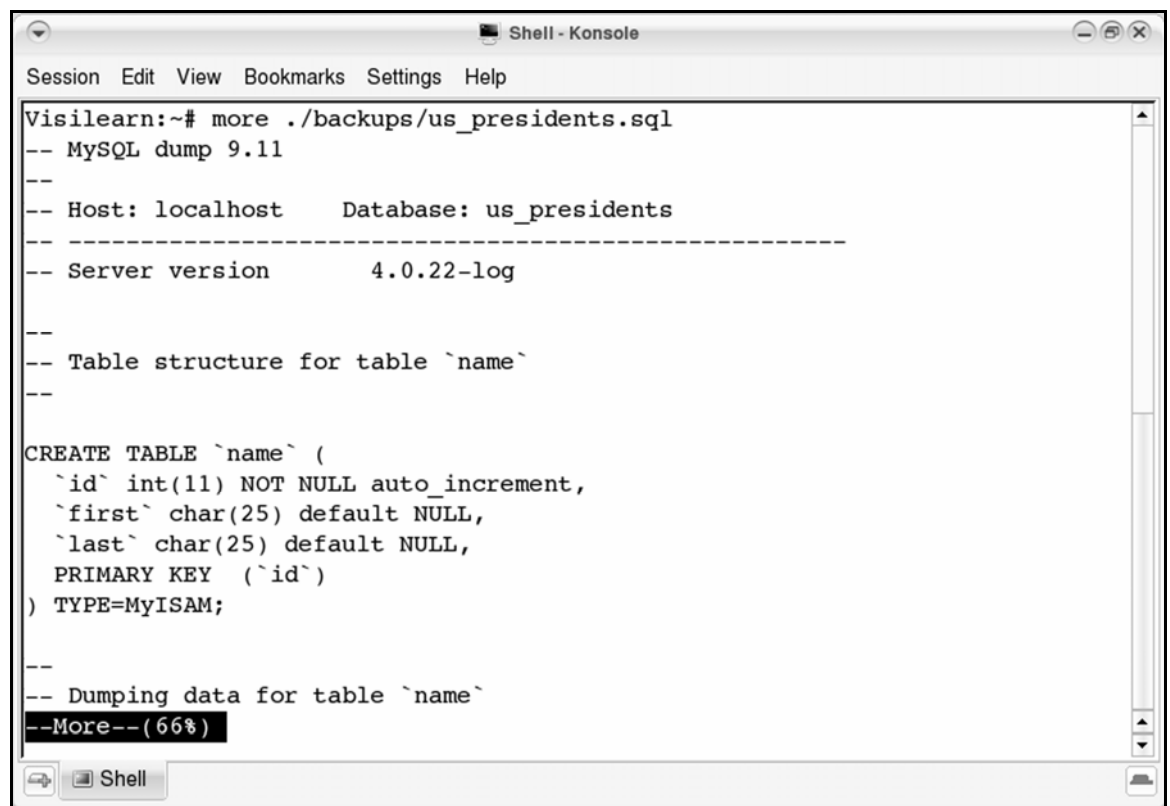
The file `us_presidents.sql` has now been created in the `backups` directory.

9. Type:

```
more ./backups/us_presidents.sql
```

then press **ENTER**.

This shows you the contents of `us_presidents.sql`:



```
Shell - Konsole
Session Edit View Bookmarks Settings Help
Visilearn:~# more ./backups/us_presidents.sql
-- MySQL dump 9.11
--
-- Host: localhost      Database: us_presidents
-----
-- Server version      4.0.22-log
--
-- Table structure for table `name`
--
CREATE TABLE `name` (
  `id` int(11) NOT NULL auto_increment,
  `first` char(25) default NULL,
  `last` char(25) default NULL,
  PRIMARY KEY (`id`)
) TYPE=MyISAM;
--
-- Dumping data for table `name`
--More--(66%)
```

Tip: *The `more` command shows you the contents of any text file.*

If the size of the file is larger than can fit in your window, you will be shown a percentage at the bottom of the page. Press the spacebar to continue scrolling down.

Delete a table

1. Type:

```
mysql -u root -p us_presidents
```

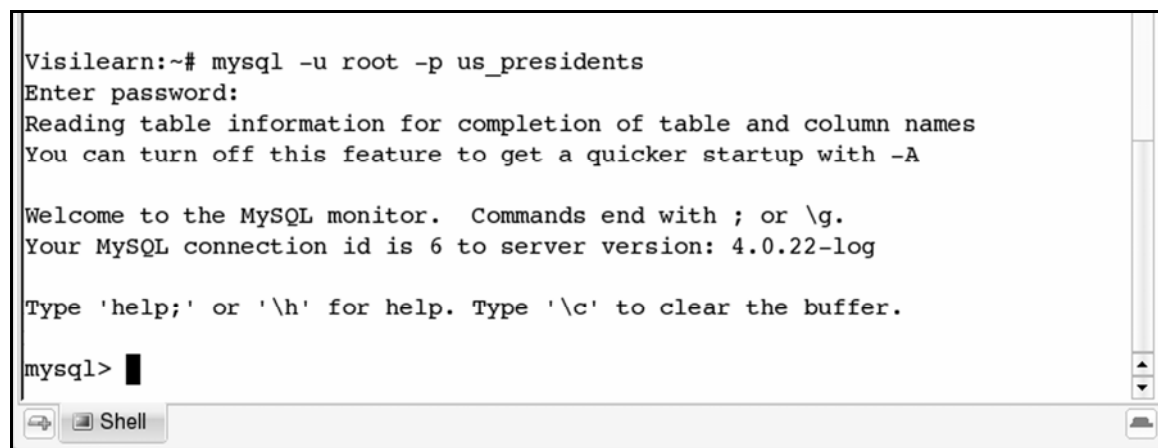
then press **ENTER**.

2. At the password prompt, type:

```
textbook
```

then press **ENTER**.

The window should look like this:

A screenshot of a terminal window titled "Shell". The terminal shows the command `mysql -u root -p us_presidents` being executed. The output includes the password prompt, a message about table information, a welcome message to the MySQL monitor, and the connection ID and version. The prompt `mysql>` is visible at the bottom of the terminal.

```
Visilearn:~# mysql -u root -p us_presidents
Enter password:
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 6 to server version: 4.0.22-log

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

mysql>
```

You're now logged into the MySQL server with the root user account and password.

You're using the `us_presidents` database.

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

```
DROP TABLE name;
```

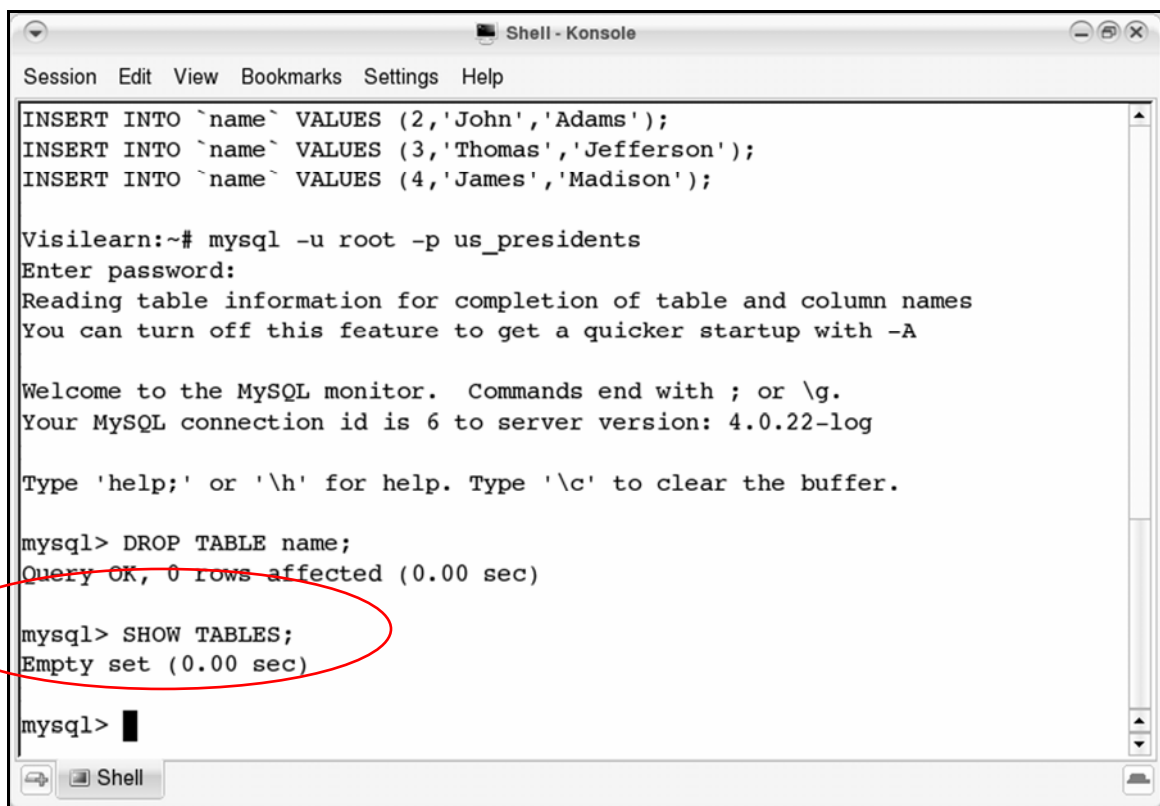
then press **ENTER**.

4. Type:

```
SHOW TABLES;
```

then press **ENTER**.

The table `name` has been dropped, or deleted, from the `us_presidents` database:

A screenshot of a terminal window titled "Shell - Konsole". The window shows a MySQL session. At the top, there are three lines of SQL commands: `INSERT INTO `name` VALUES (2, 'John', 'Adams');`, `INSERT INTO `name` VALUES (3, 'Thomas', 'Jefferson');`, and `INSERT INTO `name` VALUES (4, 'James', 'Madison');`. Below these, the user runs `mysql -u root -p us_presidents`. The terminal shows the MySQL prompt `mysql>` and the command `DROP TABLE name;`. The output is `Query OK, 0 rows affected (0.00 sec)`. The next command is `SHOW TABLES;`, which is circled in red. The output is `Empty set (0.00 sec)`. The terminal ends with `mysql>` and a cursor.

```
Session Edit View Bookmarks Settings Help
INSERT INTO `name` VALUES (2, 'John', 'Adams');
INSERT INTO `name` VALUES (3, 'Thomas', 'Jefferson');
INSERT INTO `name` VALUES (4, 'James', 'Madison');

Visilearn:~# mysql -u root -p us_presidents
Enter password:
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 6 to server version: 4.0.22-log

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

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

mysql> SHOW TABLES;
Empty set (0.00 sec)

mysql>
```

If you hadn't made a backup of the `us_presidents` database and put it in your backups directory, the table `name` would be gone forever.

Delete a database

1. Type:

```
DROP DATABASE us_presidents;
```

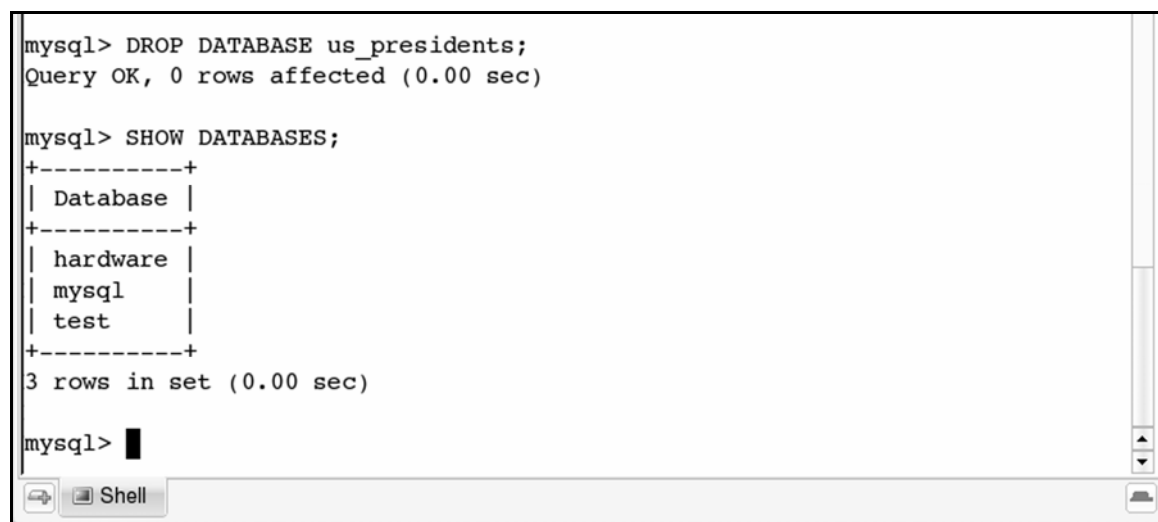
then press **ENTER**.

2. Type:

```
SHOW DATABASES;
```

then press **ENTER**.

The window should look like this:



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

mysql> SHOW DATABASES;
+-----+
| Database |
+-----+
| hardware |
| mysql    |
| test     |
+-----+
3 rows in set (0.00 sec)

mysql> █
```

The screenshot shows a terminal window titled "Shell" with a MySQL prompt. The first command executed is `DROP DATABASE us_presidents;`, which returns `Query OK, 0 rows affected (0.00 sec)`. The second command is `SHOW DATABASES;`, which returns a table with three rows: `hardware`, `mysql`, and `test`. The prompt `mysql>` is followed by a cursor.

The database `us_presidents` has been dropped, or deleted.

Restore a database

1. Type:

```
CREATE DATABASE us_presidents;
```

then press **ENTER**.

The database has been restored, but is empty. There are no tables or data in it.

2. Type:

```
\q;
```

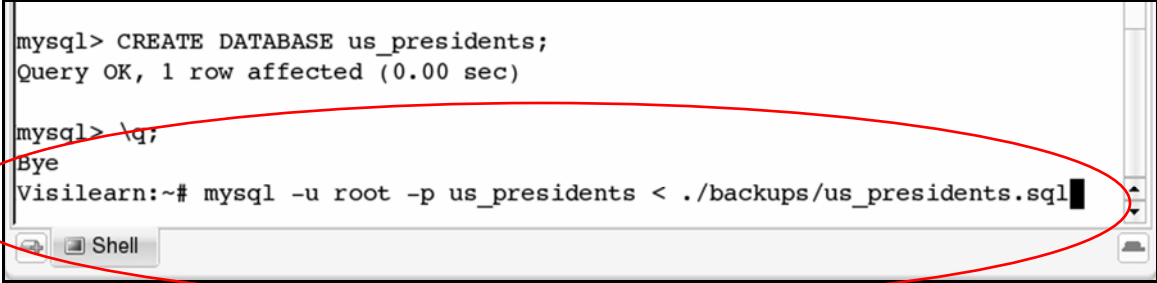
then press **ENTER**.

This closes the MySQL client connection.

You are closing the connection so you can use a Linux command line pipe (>) to restore the database.

3. Type:

```
mysql -u root -p us_presidents <
./backups/us_presidents.sql
```



The screenshot shows a terminal window with the following text: `mysql> CREATE DATABASE us_presidents;` followed by `Query OK, 1 row affected (0.00 sec)`. Below that is `mysql> \q;` followed by `Bye`. At the bottom, a shell prompt `Visilearn:~#` is shown with the command `mysql -u root -p us_presidents < ./backups/us_presidents.sql` entered. A red oval highlights the shell prompt and the command line.

then press **ENTER**.

This restores the data in the database `us_presidents` from the backup.

This command string should look familiar:

- `mysql -u root -p`

```
mysql -u root -p us_presidents <
./backups/us_presidents.sql
```

`mysql -u root -p` establishes a connection to the MySQL server using the MySQL client. The connection is made using the `root` user account and password.

- `us_presidents`

```
mysql -u root -p us_presidents <
./backups/us_presidents.sql
```

`us_presidents` is the database you want to pipe data into.

- <

```
mysql -u root -p us_presidents <
./backups/us_presidents.sql
```

Similar to the > pipe we used to backup the database, the < will read text from a file and pipe it into the MySQL server.

- **./backups/us_presidents.sql**

```
mysql -u root -p us_presidents <
./backups/us_presidents.sql
```

us_presidents.sql is the file in the backups directory that you backed up your **us_presidents** database to.

Now you're just reading it back into the **us_presidents** database on the MySQL server.

4. Type:

```
textbook
```

then press **ENTER**.

5. Type:

```
mysql -u root -p
```

then press **ENTER**.

6. At the `password` prompt, type:

```
textbook
```

then press **ENTER**.

You've reestablished a connection to MySQL Server.

7. Type:

```
USE us_presidents;
```

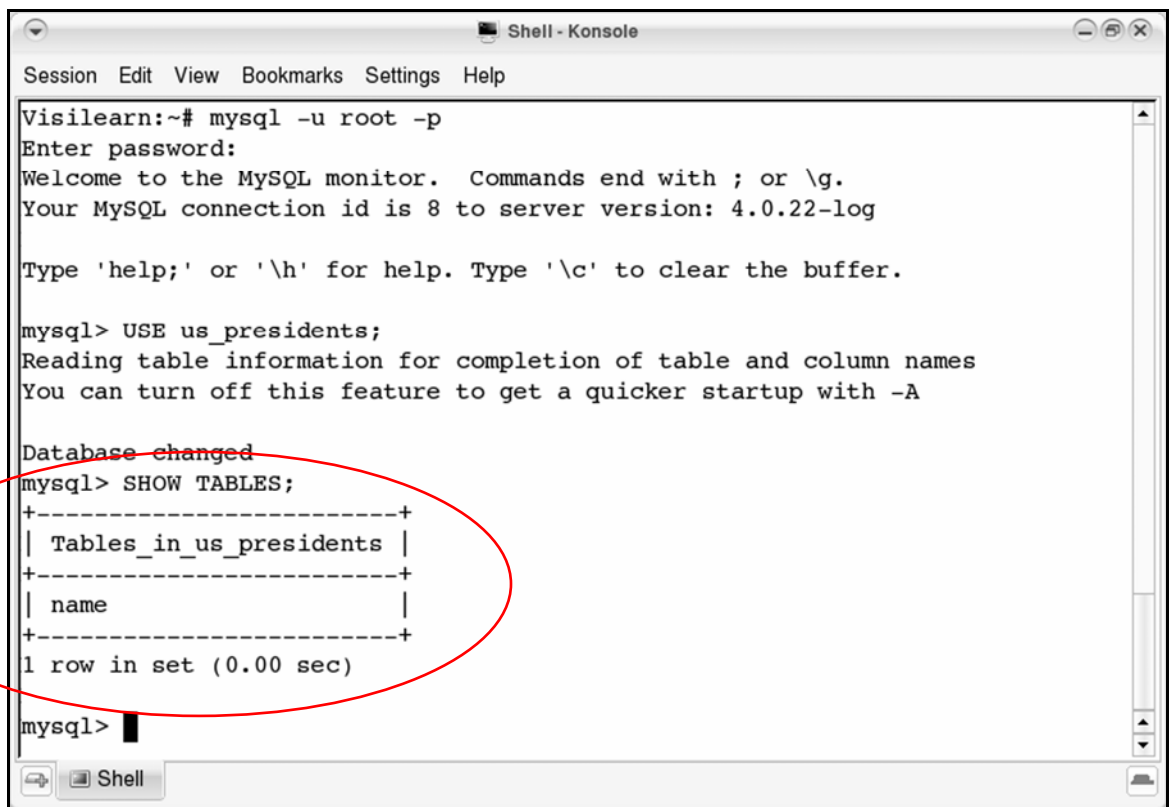
then press **ENTER**.

8. Type:

```
SHOW TABLES;
```

then press **ENTER**.

The window should look like this:



```
Shell - Konsole
Session Edit View Bookmarks Settings Help
Visilearn:~# mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8 to server version: 4.0.22-log

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

mysql> USE us_presidents;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> SHOW TABLES;
+-----+
| Tables_in_us_presidents |
+-----+
| name                     |
+-----+
1 row in set (0.00 sec)

mysql>
```

The table `name` within the database `us_presidents` has been restored.

9. Type:

```
exit
```

then press **ENTER**.

The MySQL server connection will close.

Practice:

Administering Databases

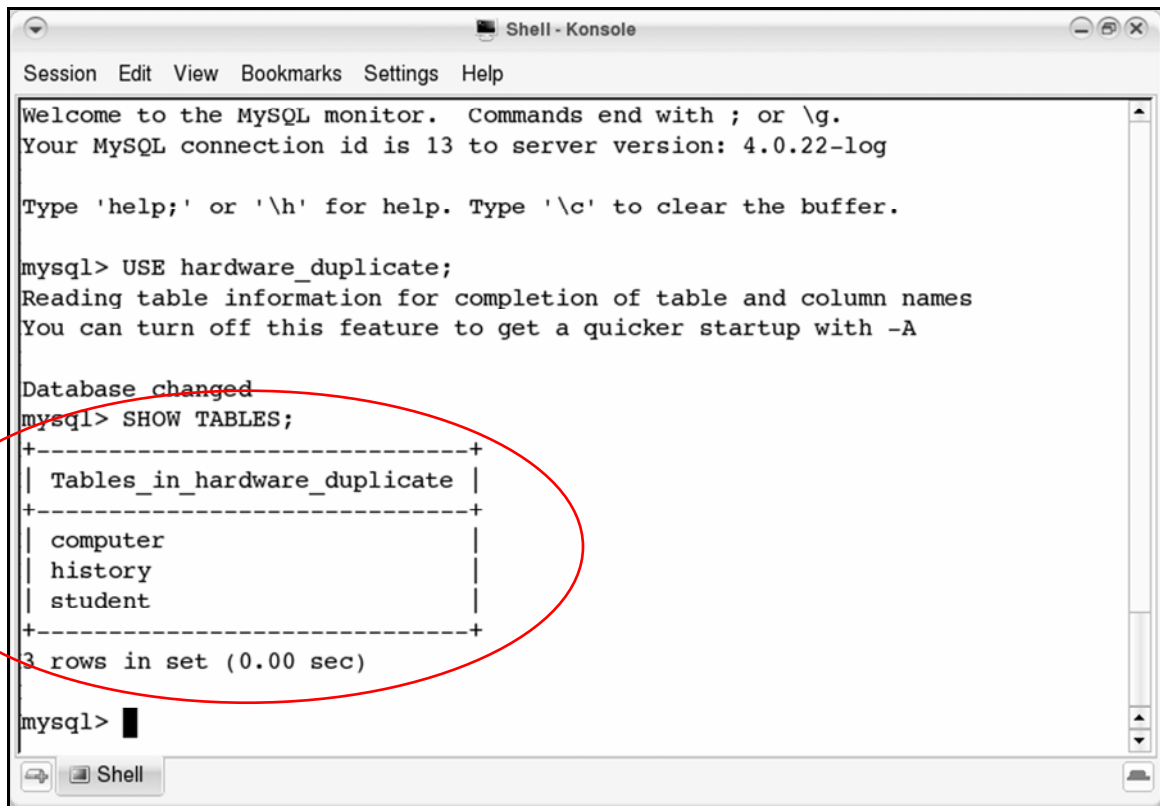
Task: The Dean of the college is concerned about losing the hardware database, because it would be difficult to reconstruct.

Put the Dean's mind at ease: Back up the `hardware` database and duplicate it to a new database named `hardware_duplicate`.

- 1.** Create a directory called `db_backup` in your home directory.
- 2.** Backup the database `hardware` to the text file `hardware.sql` at `./db_backup/hardware.sql`.
- 3.** Connect to the MySQL database server and enter the MySQL root password to gain access to it.
- 4.** Create a new database named `hardware_duplicate`.
- 5.** Using the `hardware.sql` backup file, restore the `hardware` database to the new database `hardware_duplicate`.

6. View the tables in the `hardware_duplicate` database to verify that the backup worked.

You should see the tables `student`, `computer`, and `history`:



```
Shell - Konsole
Session Edit View Bookmarks Settings Help
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 13 to server version: 4.0.22-log

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

mysql> USE hardware_duplicate;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> SHOW TABLES;
+-----+
| Tables_in_hardware_duplicate |
+-----+
| computer                      |
| history                       |
| student                      |
+-----+
3 rows in set (0.00 sec)

mysql> █
```

7. Close the MySQL database connection.
8. Exit the **Konsole** window.

Working with Tables

In this section, you'll learn how to:

- **Alter tables**
- **Update records**
- **Delete records**

Alter tables

1. Open the **Konsole** window.

2. Type:

```
mysql -u root -p us_presidents
```

then press **ENTER**.

This command string establishes a connection to the MySQL server, specifically the database `us_presidents`.

3. At the `password` prompt, type:

```
textbook
```

then press **ENTER**.

4. Type:

```
ALTER TABLE name ADD COLUMN party CHAR(25);
```

then press **ENTER**.

This command string will add a field, or column, to the table `name`. MySQL refers to table fields as columns.

These commands read pretty much like a sentence in English:

ALTER the **TABLE** `name` by **ADDING** a **COLUMN** called `party`. Then make `party` a column that contains a maximum of 25 characters.

Now the table `name` is organized like this, with a new field called `party`:

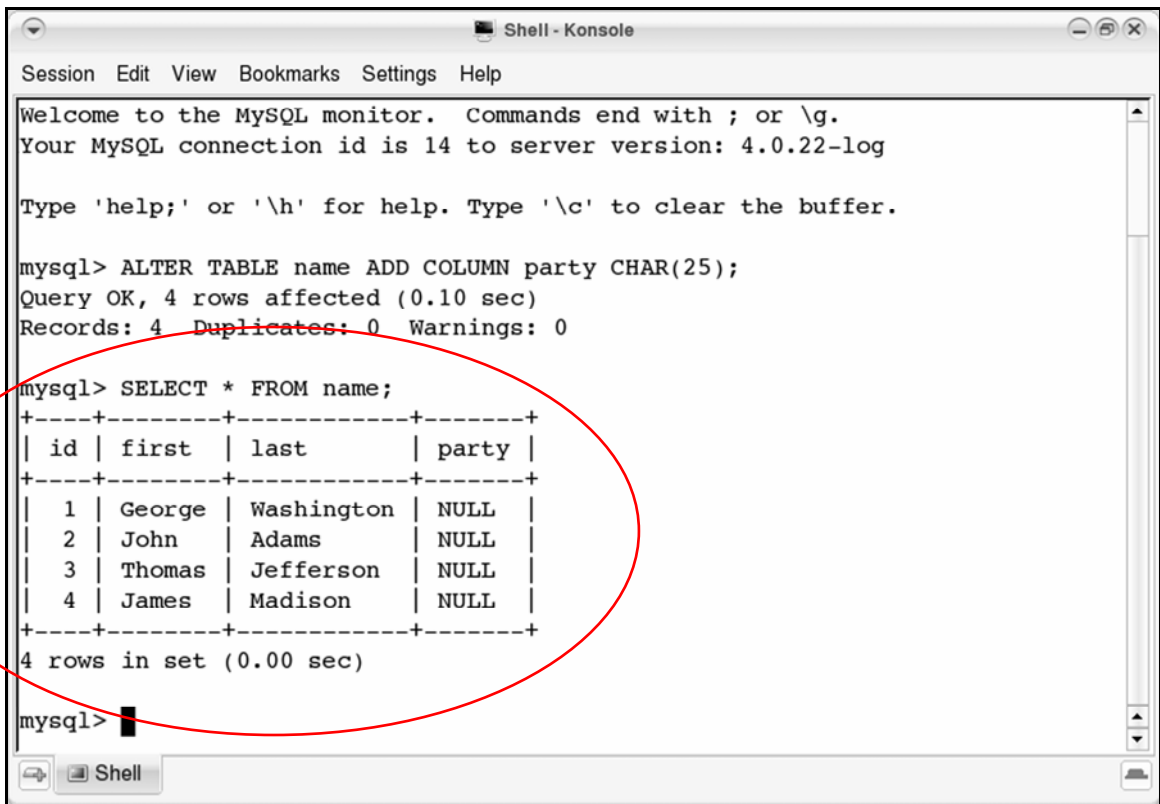
Column	Datatype	Properties
<code>id</code>	<code>INT</code>	primary key, not null, auto increment
<code>first</code>	<code>CHAR(25)</code>	
<code>last</code>	<code>CHAR(25)</code>	
<code>party</code>	<code>CHAR(25)</code>	

5. Type:

```
SELECT * FROM name;
```

then press **ENTER**.

The window should look like this:



```
Shell - Konsole
Session Edit View Bookmarks Settings Help

Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 14 to server version: 4.0.22-log

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

mysql> ALTER TABLE name ADD COLUMN party CHAR(25);
Query OK, 4 rows affected (0.10 sec)
Records: 4  Duplicates: 0  Warnings: 0

mysql> SELECT * FROM name;
+----+-----+-----+-----+
| id | first | last   | party |
+----+-----+-----+-----+
|  1 | George | Washington | NULL |
|  2 | John  | Adams    | NULL |
|  3 | Thomas | Jefferson | NULL |
|  4 | James | Madison  | NULL |
+----+-----+-----+-----+
4 rows in set (0.00 sec)

mysql>
```

id	first	last	party
1	George	Washington	null
2	John	Adams	null
3	Thomas	Jefferson	null
4	James	Madison	null

Update records

1. Type:

```
UPDATE name SET party='Federalist' ►►  
WHERE (last='Washington' OR last='Adams');
```

then press **ENTER**.

The **UPDATE** command fills in the blank entries in the **name** table that were created when you added the **party** field.

This string of commands reads like this:

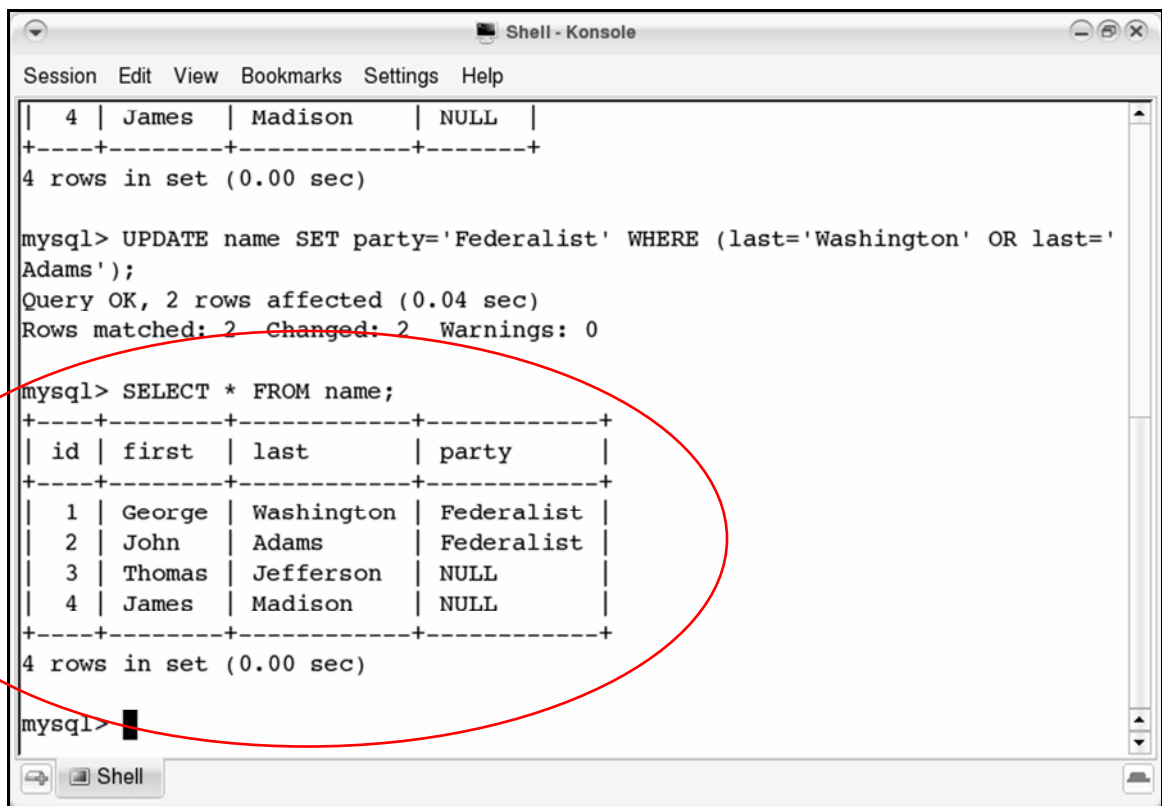
```
UPDATE the table name . SET the party field to "Federalist"  
WHERE the last name of the president is either "Washington" OR  
"Adams."
```

2. Type:

```
SELECT * FROM name;
```

then press **ENTER**.

The window should look like this:



```
Session Edit View Bookmarks Settings Help
| 4 | James | Madison | NULL |
+-----+-----+-----+
4 rows in set (0.00 sec)

mysql> UPDATE name SET party='Federalist' WHERE (last='Washington' OR last='
Adams');
Query OK, 2 rows affected (0.04 sec)
Rows matched: 2  Changed: 2  Warnings: 0

mysql> SELECT * FROM name;
+-----+-----+-----+
| id | first | last | party |
+-----+-----+-----+
| 1 | George | Washington | Federalist |
| 2 | John | Adams | Federalist |
| 3 | Thomas | Jefferson | NULL |
| 4 | James | Madison | NULL |
+-----+-----+-----+
4 rows in set (0.00 sec)

mysql>
```

id	first	last	party
1	George	Washington	Federalist
2	John	Adams	Federalist
3	Thomas	Jefferson	
4	James	Madison	

3. Type:

```
UPDATE name SET ►►  
party='Democratic Republican' ►►  
WHERE (last='Jefferson' OR ►►  
last='Madison');
```

then press **ENTER**.

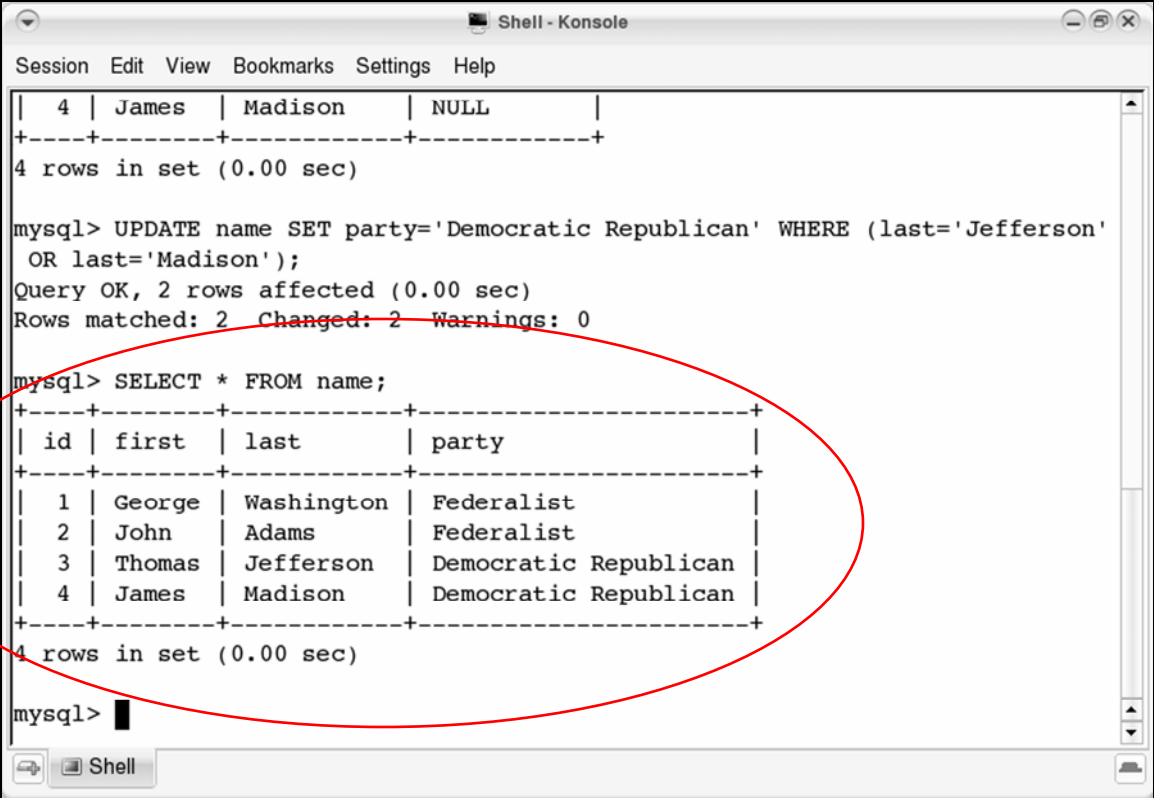
This updates the party affiliations for Jefferson and Madison.

4. Type:

```
SELECT * FROM name;
```

then press **ENTER**.

The window should look like this:



```
Shell - Konsole
Session Edit View Bookmarks Settings Help

| 4 | James | Madison | NULL |
+---+-----+-----+-----+
4 rows in set (0.00 sec)

mysql> UPDATE name SET party='Democratic Republican' WHERE (last='Jefferson'
OR last='Madison');
Query OK, 2 rows affected (0.00 sec)
Rows matched: 2  Changed: 2  Warnings: 0

mysql> SELECT * FROM name;
+---+-----+-----+-----+
| id | first | last   | party                |
+---+-----+-----+-----+
| 1  | George| Washington | Federalist           |
| 2  | John  | Adams    | Federalist           |
| 3  | Thomas| Jefferson | Democratic Republican |
| 4  | James | Madison  | Democratic Republican |
+---+-----+-----+-----+
4 rows in set (0.00 sec)

mysql>
```

id	first	last	party
1	George	Washington	Federalist
2	John	Adams	Federalist
3	Thomas	Jefferson	Democratic Republican
4	James	Madison	Democratic Republican

Delete records

1. Type:

```
DELETE FROM name WHERE id>2;
```

then press **ENTER**.

The **DELETE** command deletes records that match the criteria you set.

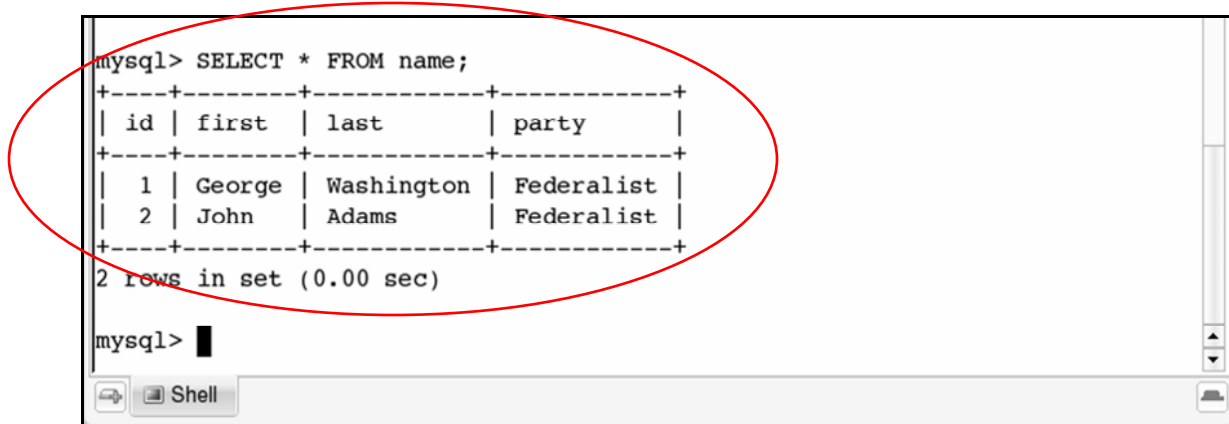
In this case, you told MySQL to **DELETE** from the table **name** any records **WHERE** the value for **id** is **greater than 2**.

2. Type:

```
SELECT * FROM name;
```

then press **ENTER**.

The table should now hold only these records:



```
mysql> SELECT * FROM name;
+----+-----+-----+-----+
| id | first | last  | party |
+----+-----+-----+-----+
| 1  | George | Washington | Federalist |
| 2  | John  | Adams   | Federalist |
+----+-----+-----+-----+
2 rows in set (0.00 sec)

mysql> █
```

id	first	last	party
1	George	Washington	Federalist
2	John	Adams	Federalist

3. Type:

```
\q;
```

then press **ENTER**

to close the MySQL database connection.

4. Type:

```
exit
```

then press **ENTER**

to exit the **Konsole** window.

Practice:

Working with Tables

Task: All of the students in the `hardware` database are in different departments. Add a column to the `students` table to keep track of which department a student is in.

1. Open the **Konsole** window.

2. Type:

```
mysql -u root -p hardware
```

then press **ENTER**

to connect to the MySQL database server, then the `hardware` database.

3. Type:

```
textbook
```

then press **ENTER**.

4. In the `student` table, add a column named `department` using the **ALTER** command.

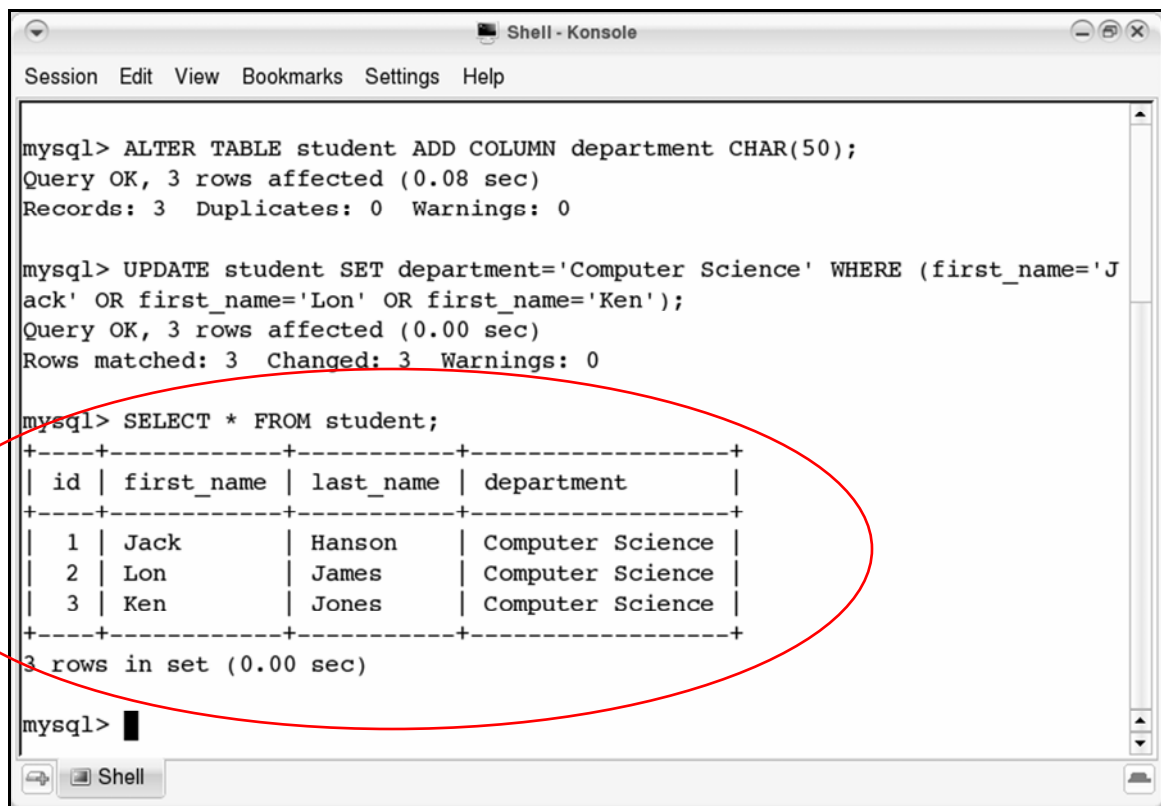
The column should hold up to 50 characters: `char (50)`.

5. Now **UPDATE** the values in the new column:

Specify that Jack, Lon and Ken be in the Computer Science Department.

6. Run a query that selects everything from the `student` table.

It should look like this:



```
mysql> ALTER TABLE student ADD COLUMN department CHAR(50);
Query OK, 3 rows affected (0.08 sec)
Records: 3 Duplicates: 0 Warnings: 0

mysql> UPDATE student SET department='Computer Science' WHERE (first_name='Jack' OR first_name='Lon' OR first_name='Ken');
Query OK, 3 rows affected (0.00 sec)
Rows matched: 3 Changed: 3 Warnings: 0

mysql> SELECT * FROM student;
+----+-----+-----+-----+
| id | first_name | last_name | department |
+----+-----+-----+-----+
| 1  | Jack      | Hanson   | Computer Science |
| 2  | Lon       | James    | Computer Science |
| 3  | Ken       | Jones    | Computer Science |
+----+-----+-----+-----+
3 rows in set (0.00 sec)

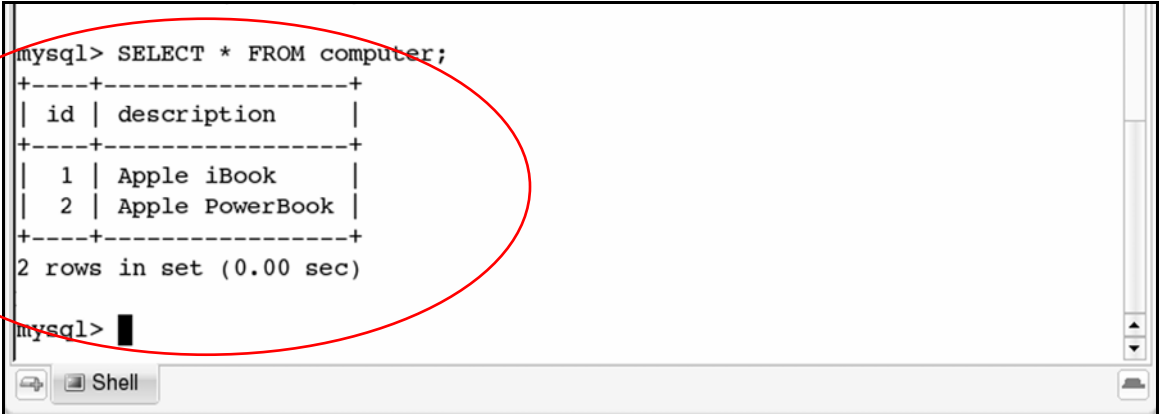
mysql>
```

id	first_name	last_name	department
1	Jack	Hanson	Computer Science
2	Lon	James	Computer Science
3	Ken	Jones	Computer Science

7. One of the computers, the Apple iMac, is not used any more, so **DELETE** it from the **computer** table.

8. Run a query that selects everything from the **computer** table.

It should look like this:



```
mysql> SELECT * FROM computer;
+----+-----+
| id | description |
+----+-----+
|  1 | Apple iBook |
|  2 | Apple PowerBook |
+----+-----+
2 rows in set (0.00 sec)

mysql>
```

id	description
1	Apple iBook
2	Apple PowerBook

9. Close the MySQL database connection.


10. Exit the **Konsole** window.

Running Queries

In this section, you'll learn how to:

- **Sort query results**
- **Add query criteria**

Sort query results

1. On the launcher bar, click the  button to open a browser window.

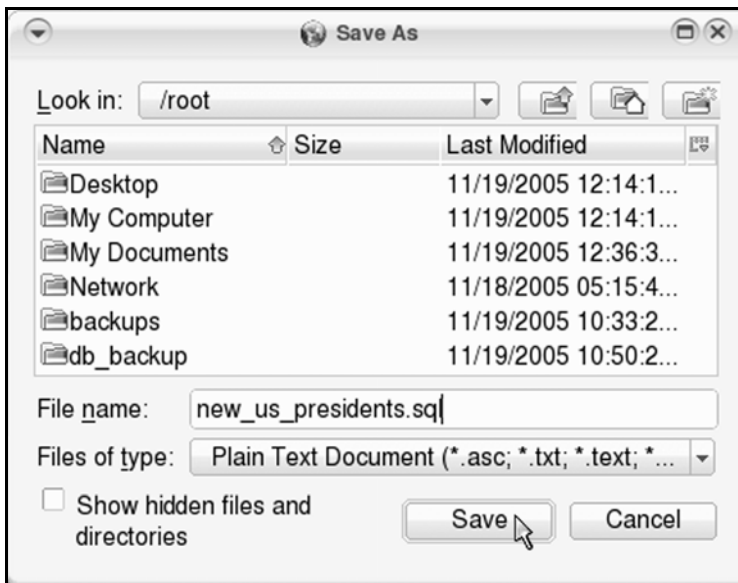


2. When the browser opens, go to:

www.visibooks.com/books/mysql

3. Right-click the [new us presidents.sql](#) link.

Then save the file in your home directory:



4. Open the **Konsole** window and type:

```
mysql -u root -p us_presidents <  
./new_us_presidents.sql
```

then press **ENTER**.

This command string pipes the data from the file you just downloaded (`new_us_presidents.sql`) into the database `us_presidents`.

5. Type your MySQL root password—`textbook`—then press **ENTER** to execute the command string.

6. Type:

```
mysql -u root -p us_presidents
```

then press **ENTER**.

7. Type your MySQL root password, then press **ENTER**.

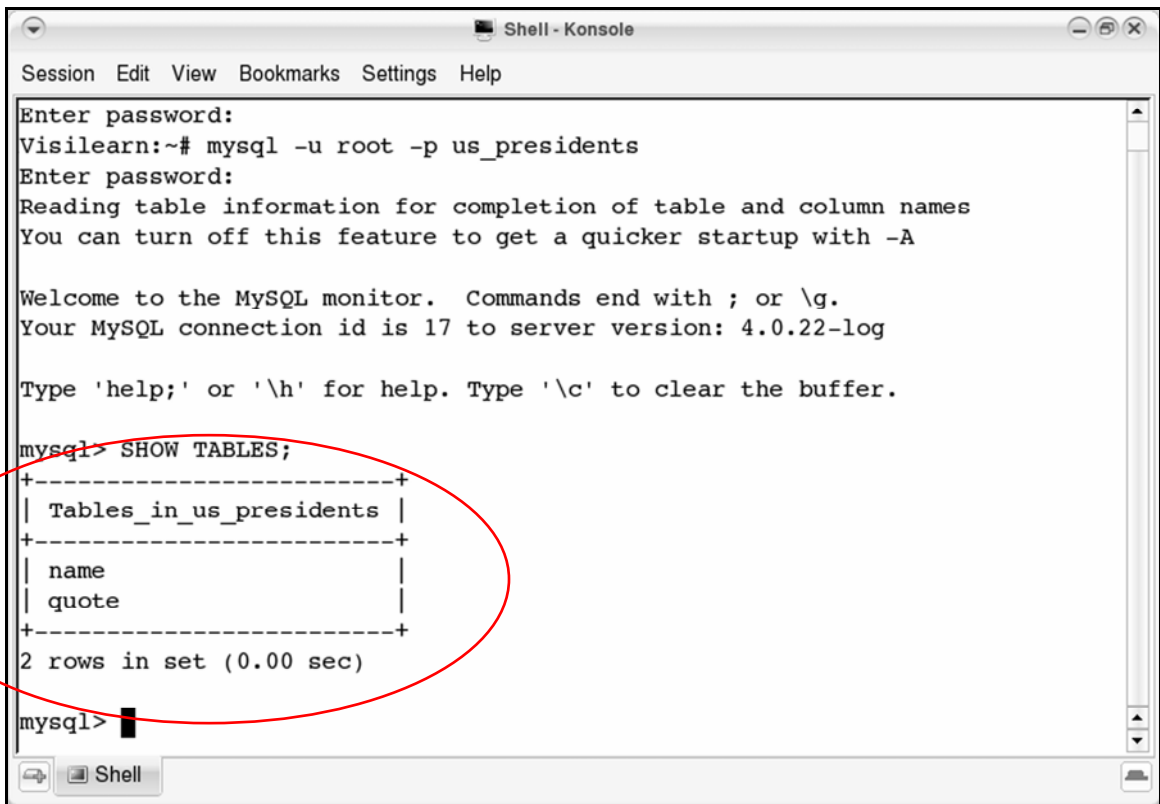
This will connect you to the `us_presidents` database on the MySQL server.

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

```
SHOW TABLES;
```

then press **ENTER**.

This will **SHOW** the **TABLES** in the `us_presidents` database:



```
Shell - Konsole
Session Edit View Bookmarks Settings Help
Enter password:
Visilearn:~# mysql -u root -p us_presidents
Enter password:
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 17 to server version: 4.0.22-log

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

mysql> SHOW TABLES;
+-----+
| Tables_in_us_presidents |
+-----+
| name                     |
| quote                    |
+-----+
2 rows in set (0.00 sec)

mysql>
```

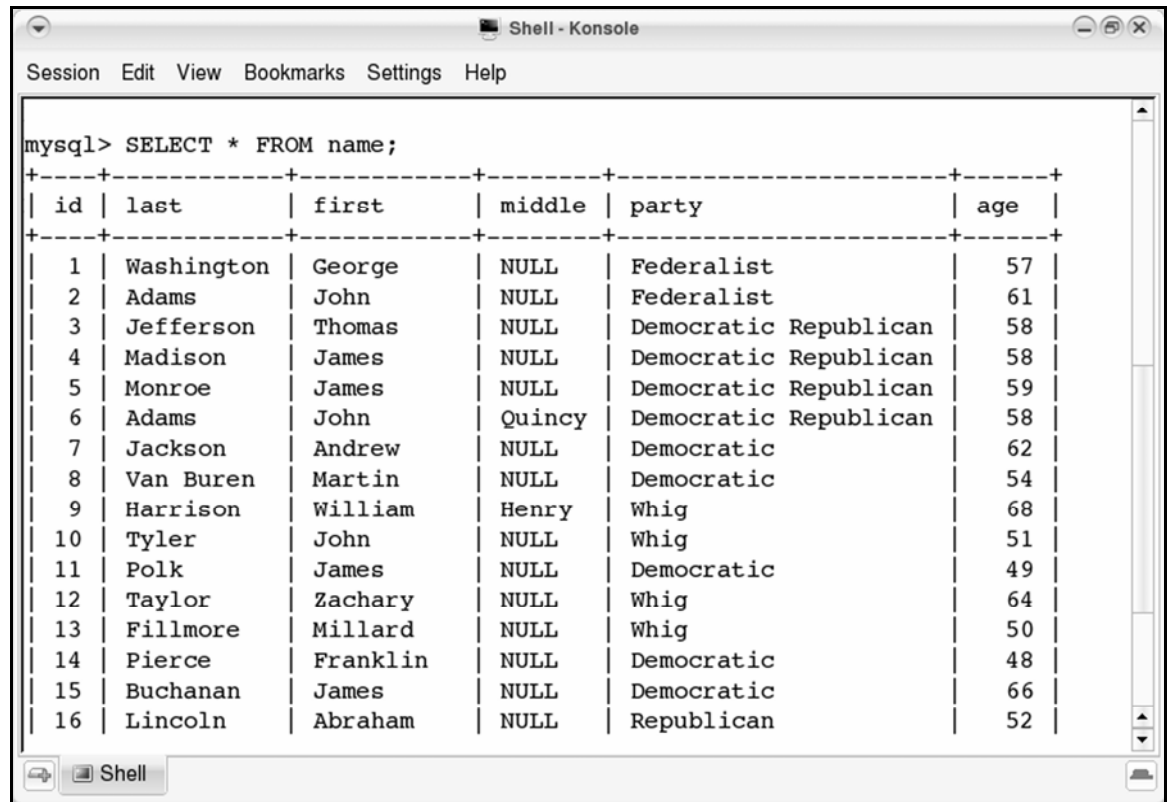
The `new_us_presidents.sql` file you piped in contained two new tables, `name` and `quote`. These are now in the `us_presidents` database.

9. Type:

```
SELECT * FROM name;
```

then press **ENTER**.

The data in the `names` table should look like this:



```
mysql> SELECT * FROM name;
```

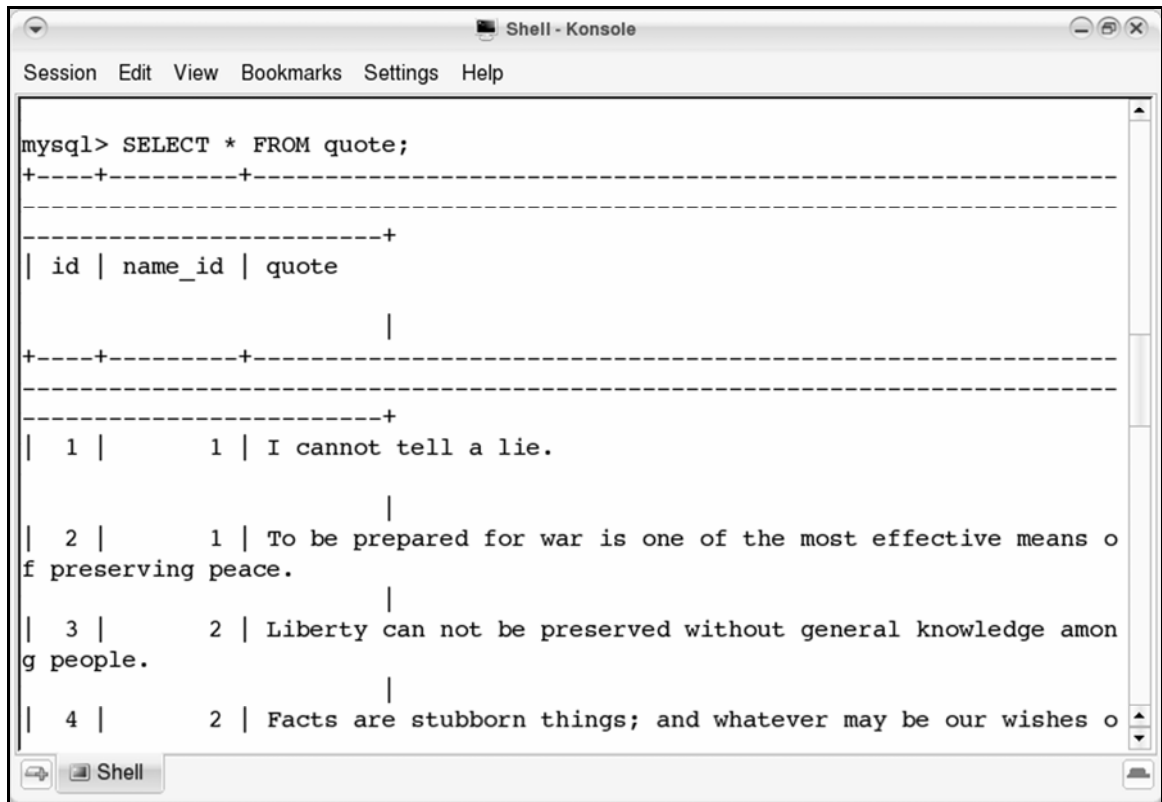
id	last	first	middle	party	age
1	Washington	George	NULL	Federalist	57
2	Adams	John	NULL	Federalist	61
3	Jefferson	Thomas	NULL	Democratic Republican	58
4	Madison	James	NULL	Democratic Republican	58
5	Monroe	James	NULL	Democratic Republican	59
6	Adams	John	Quincy	Democratic Republican	58
7	Jackson	Andrew	NULL	Democratic	62
8	Van Buren	Martin	NULL	Democratic	54
9	Harrison	William	Henry	Whig	68
10	Tyler	John	NULL	Whig	51
11	Polk	James	NULL	Democratic	49
12	Taylor	Zachary	NULL	Whig	64
13	Fillmore	Millard	NULL	Whig	50
14	Pierce	Franklin	NULL	Democratic	48
15	Buchanan	James	NULL	Democratic	66
16	Lincoln	Abraham	NULL	Republican	52

10. Type:

```
SELECT * FROM quote;
```

then press **ENTER**.

The data in the `quotes` table should look like this:



The screenshot shows a terminal window titled "Shell - Konsole" with a menu bar containing "Session", "Edit", "View", "Bookmarks", "Settings", and "Help". The terminal content is as follows:

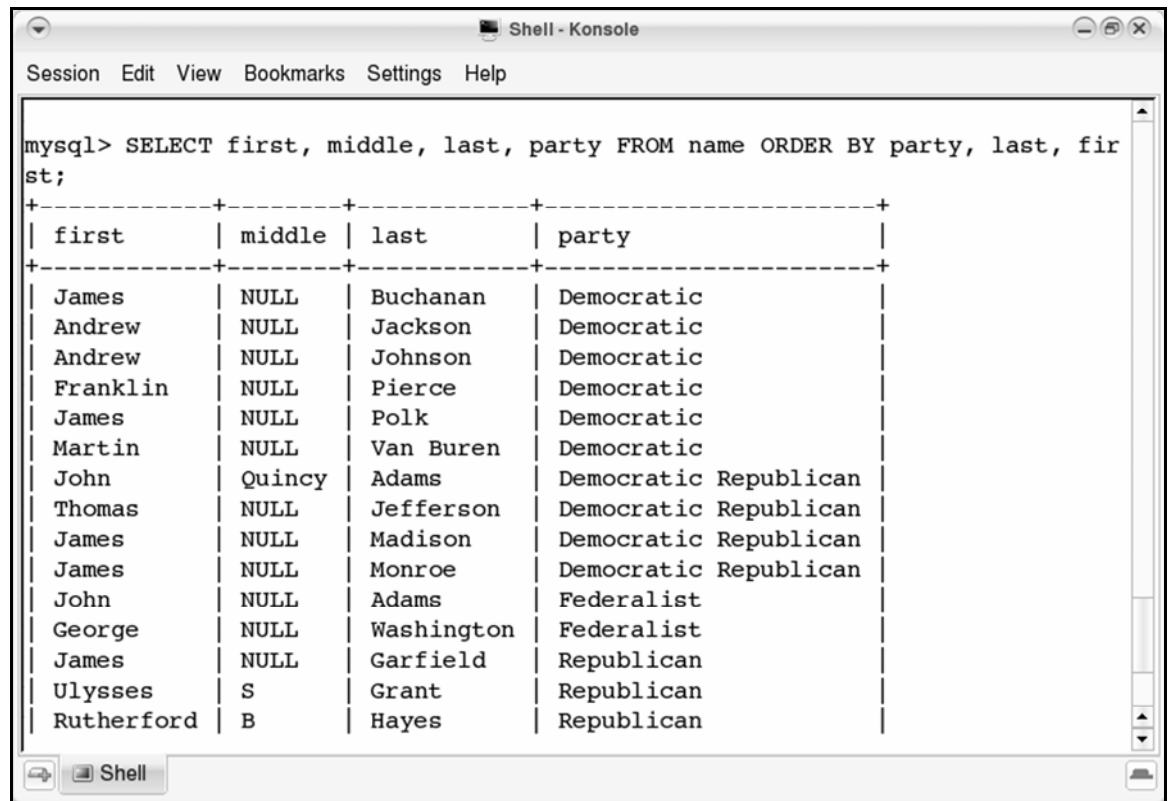
```
mysql> SELECT * FROM quote;
+-----+-----+-----+
| id | name_id | quote
+-----+-----+-----+
| 1 | 1 | I cannot tell a lie.
| 2 | 1 | To be prepared for war is one of the most effective means o
f preserving peace.
| 3 | 2 | Liberty can not be preserved without general knowledge amon
g people.
| 4 | 2 | Facts are stubborn things; and whatever may be our wishes o
```

11. Type:

```
SELECT first,middle,last,party ►►  
FROM name ►►  
ORDER BY party,last,first;
```

then press **ENTER**.

The query results should look like this:



```
mysql> SELECT first, middle, last, party FROM name ORDER BY party, last, fir  
st;  
+-----+-----+-----+-----+  
| first      | middle | last      | party      |  
+-----+-----+-----+-----+  
| James      | NULL   | Buchanan  | Democratic |  
| Andrew     | NULL   | Jackson   | Democratic |  
| Andrew     | NULL   | Johnson   | Democratic |  
| Franklin   | NULL   | Pierce    | Democratic |  
| James      | NULL   | Polk      | Democratic |  
| Martin     | NULL   | Van Buren | Democratic |  
| John       | Quincy | Adams     | Democratic |  
| Thomas     | NULL   | Jefferson | Democratic |  
| James      | NULL   | Madison   | Democratic |  
| James      | NULL   | Monroe    | Democratic |  
| John       | NULL   | Adams     | Federalist |  
| George     | NULL   | Washington| Federalist |  
| James      | NULL   | Garfield  | Republican |  
| Ulysses    | S      | Grant     | Republican |  
| Rutherford | B      | Hayes     | Republican |
```

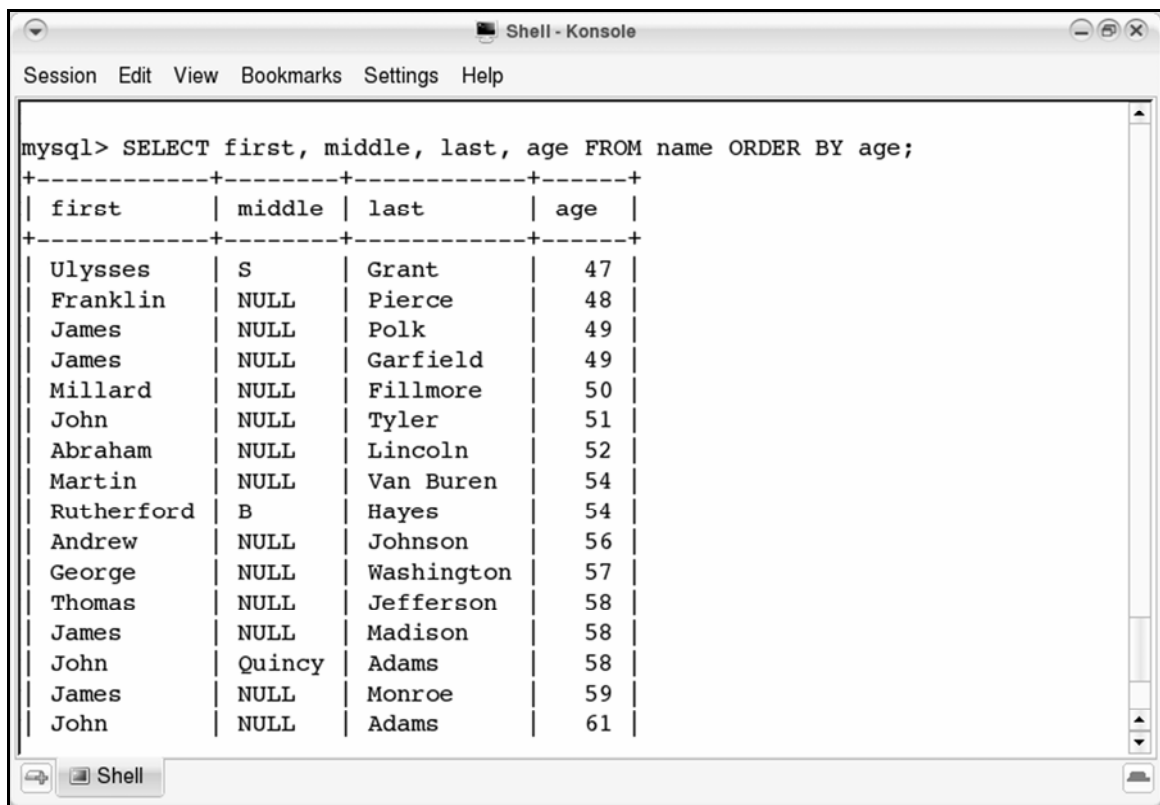
This query lists the presidents' names and parties, then sorts them by party, last name, then first name.

12. Type:

```
SELECT first,middle,last,age ►►  
FROM name ►►  
ORDER BY age;
```

then press **ENTER**.

The query results should look like this:



```
mysql> SELECT first, middle, last, age FROM name ORDER BY age;
```

first	middle	last	age
Ulysses	S	Grant	47
Franklin	NULL	Pierce	48
James	NULL	Polk	49
James	NULL	Garfield	49
Millard	NULL	Fillmore	50
John	NULL	Tyler	51
Abraham	NULL	Lincoln	52
Martin	NULL	Van Buren	54
Rutherford	B	Hayes	54
Andrew	NULL	Johnson	56
George	NULL	Washington	57
Thomas	NULL	Jefferson	58
James	NULL	Madison	58
John	Quincy	Adams	58
James	NULL	Monroe	59
John	NULL	Adams	61

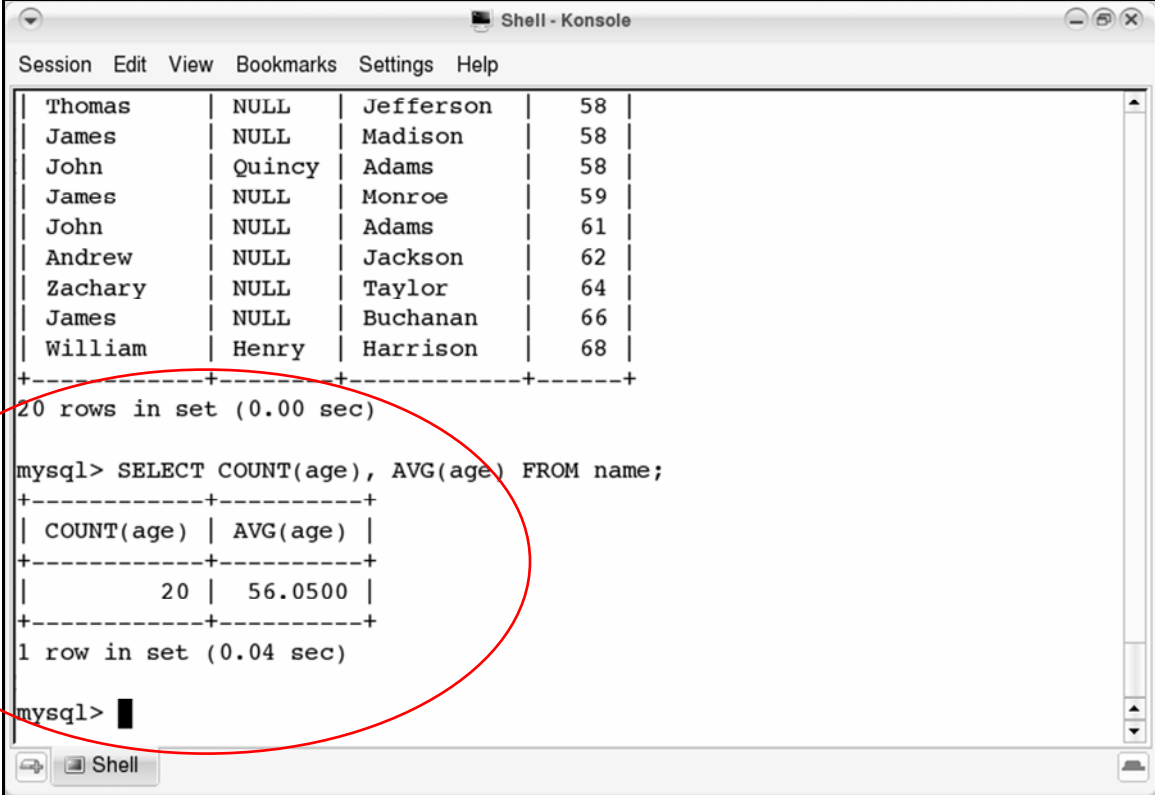
This query lists the presidents in order, by their age when they took office.

13. Type:

```
SELECT COUNT (age) , AVG (age) ►►  
FROM name;
```

then press **ENTER**.

The query results should look like this:



```
Shell - Konsole  
Session Edit View Bookmarks Settings Help  
+-----+-----+-----+-----+  
| Thomas | NULL | Jefferson | 58 |  
| James  | NULL | Madison   | 58 |  
| John   | Quincy | Adams     | 58 |  
| James  | NULL | Monroe    | 59 |  
| John   | NULL | Adams     | 61 |  
| Andrew | NULL | Jackson   | 62 |  
| Zachary | NULL | Taylor    | 64 |  
| James  | NULL | Buchanan  | 66 |  
| William | Henry | Harrison  | 68 |  
+-----+-----+-----+-----+  
20 rows in set (0.00 sec)  
  
mysql> SELECT COUNT(age), AVG(age) FROM name;  
+-----+-----+  
| COUNT(age) | AVG(age) |  
+-----+-----+  
|          20 | 56.0500 |  
+-----+-----+  
1 row in set (0.04 sec)  
  
mysql> █
```

This query does two things:

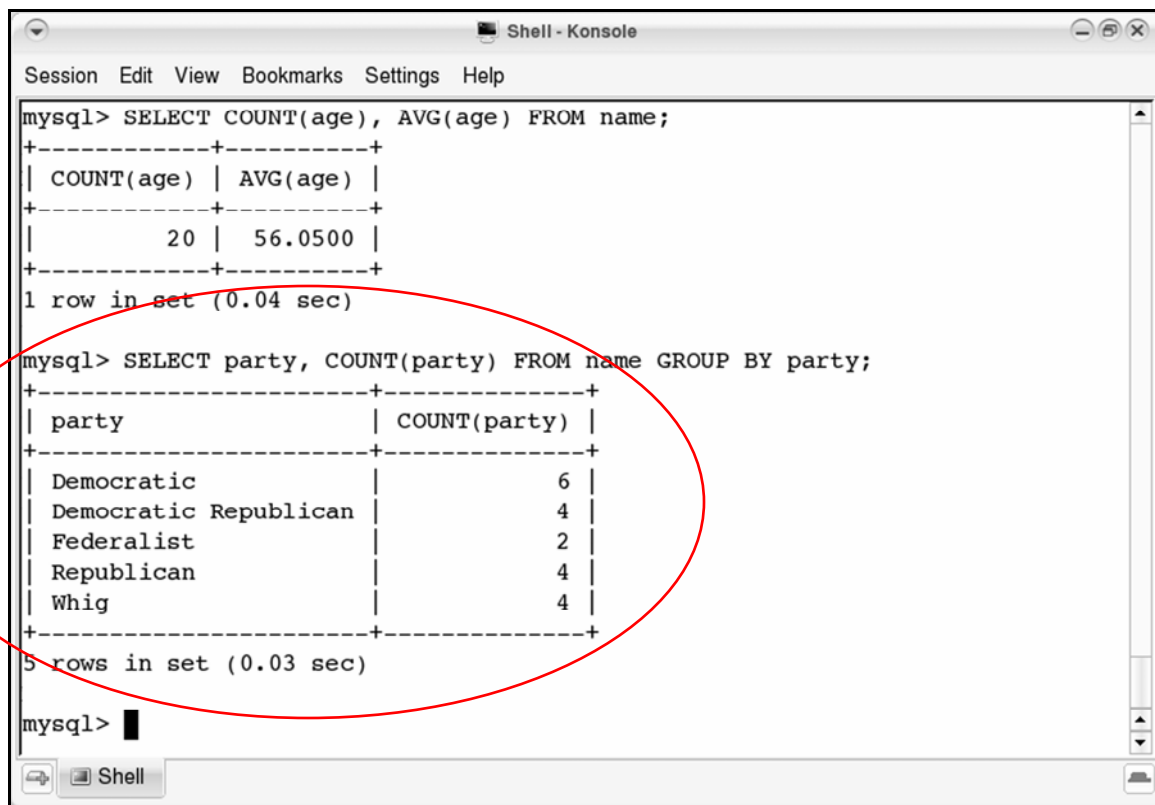
- **COUNT** the number of presidents in the **name** table.
- Calculate the **AVG** (average) age of these presidents when they took office.

14. Type:

```
SELECT party, COUNT (party) ►►
FROM name GROUP BY party;
```

then press **ENTER**.

The query results should look like this:



```
mysql> SELECT COUNT(age), AVG(age) FROM name;
+-----+-----+
| COUNT(age) | AVG(age) |
+-----+-----+
|          20 | 56.0500 |
+-----+-----+
1 row in set (0.04 sec)

mysql> SELECT party, COUNT(party) FROM name GROUP BY party;
+-----+-----+
| party                | COUNT(party) |
+-----+-----+
| Democratic            |             6 |
| Democratic Republican |             4 |
| Federalist            |             2 |
| Republican            |             4 |
| Whig                  |             4 |
+-----+-----+
5 rows in set (0.03 sec)

mysql>
```

This query answers a simple question: how many presidents were in each of the different parties?

If you look at a portion of the query...

```
SELECT party, COUNT (party) FROM name GROUP BY
party;
```

...it lists the party for each president in the `name` table.

Adding the other two parts...

```
SELECT party, COUNT(party) FROM name GROUP BY  
party;
```

...changes things. Instead of listing all 20 presidents, the list will now be **GROUPed** into sub lists of presidents of *like parties*, and then **COUNTed**.

In the end, you see one row for each party – a total of 5 rows. Each row contains the party name and the number of presidents affiliated with that party.

Add query criteria

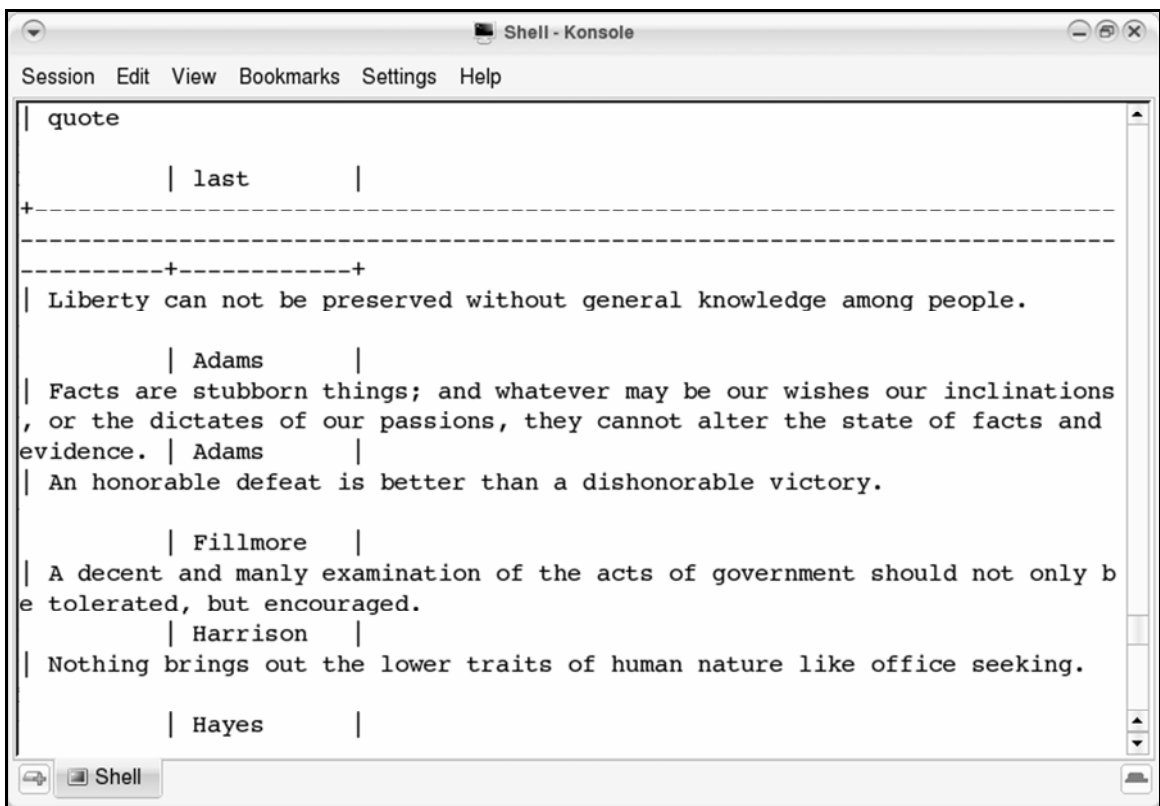
Up to this point, you've only queried from one table. Now use multiple tables in a query:

1. Type:

```
SELECT quote,last FROM quote,name >>>
WHERE quote.name_id=name.id >>>
ORDER BY last;
```

then press **ENTER**.

The query results should look like this:



```
Shell - Konsole
Session Edit View Bookmarks Settings Help
| quote
| last
+-----+
| Liberty can not be preserved without general knowledge among people.
| Adams
| Facts are stubborn things; and whatever may be our wishes our inclinations
, or the dictates of our passions, they cannot alter the state of facts and
evidence. | Adams
| An honorable defeat is better than a dishonorable victory.
| Fillmore
| A decent and manly examination of the acts of government should not only b
e tolerated, but encouraged.
| Harrison
| Nothing brings out the lower traits of human nature like office seeking.
| Hayes
```

This query lists all of the quotes **FROM** the `quote` table, along with the last names of the presidents (pulled from the `name` table) who said them.

Let's look at each portion of the query:

- **SELECT quote,last**

```
SELECT quote,last FROM quote,name ►►  
WHERE quote.name_id=name.id ►►  
ORDER BY last;
```

This part looks the same as in previous queries, except the `quote` and `last` fields being queried are in different tables.

- **FROM quote,name**

```
SELECT quote,last FROM quote,name ►►  
WHERE quote.name_id=name.id ►►  
ORDER BY last;
```

`quote` and `name` are the two tables you're using in the query. The field `quote` is in the `quote` table; the field `last` is in the `name` table.

- **WHERE quote.name_id=name.id**

```
SELECT quote,last FROM quote,name ►►  
WHERE quote.name_id=name.id ►►  
ORDER BY last;
```

The **WHERE** criterion links the `quote` and `name` tables together. This string tells the database that the `name_id` of a record in the `quote` table corresponds to a record with the same `id` in the `name` table.

For instance, the president whose `id` is 1 delivered all quotes with an `name_id` of 1; the president whose `id` is 2 delivered quotes with `name_id` of 2, and so on.

- **ORDER BY last**

```
SELECT quote,last FROM quote,name ►►  
WHERE quote.name_id=name.id ►►  
ORDER BY last;
```

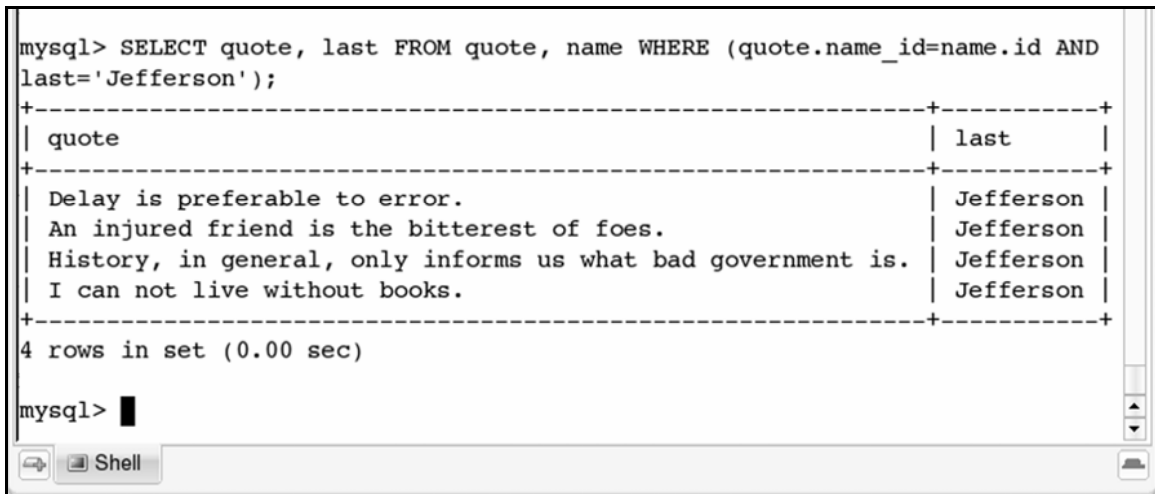
This puts the list in alphabetical order by the presidents' last names.

2. Type:

```
SELECT quote,last FROM quote,name ►►  
WHERE (quote.name_id=name.id ►►  
AND last='Jefferson');
```

then press **ENTER**.

The query results should look like this:



```
mysql> SELECT quote, last FROM quote, name WHERE (quote.name_id=name.id AND  
last='Jefferson');
```

quote	last
Delay is preferable to error.	Jefferson
An injured friend is the bitterest of foes.	Jefferson
History, in general, only informs us what bad government is.	Jefferson
I can not live without books.	Jefferson

```
4 rows in set (0.00 sec)  
  
mysql> █
```

The screenshot shows a terminal window with a title bar that says "Shell". The terminal output includes the SQL query, a table of results with columns "quote" and "last", and the message "4 rows in set (0.00 sec)". The cursor is positioned at the end of the second "mysql>" prompt.

This query joins the two tables `quote` and `name`, but you're using different criteria in the `WHERE` statement:

```
WHERE (quote.name_id=name.id  
AND last='Jefferson')
```

The first condition is the same as before:

```
quote.name_id=name.id
```

`name_id` (in the `quote` table) and `id` (in the `name` table) are the link between the two tables.

The second condition:

```
last='Jefferson'
```

narrows the query to only those quotes from presidents with the last name of `Jefferson`.

The single quotes surrounding `'Jefferson'` tell the database that `Jefferson` is text.

Tip: *If you use numeric criteria in your query, don't use quotes. For instance, you'd type:*

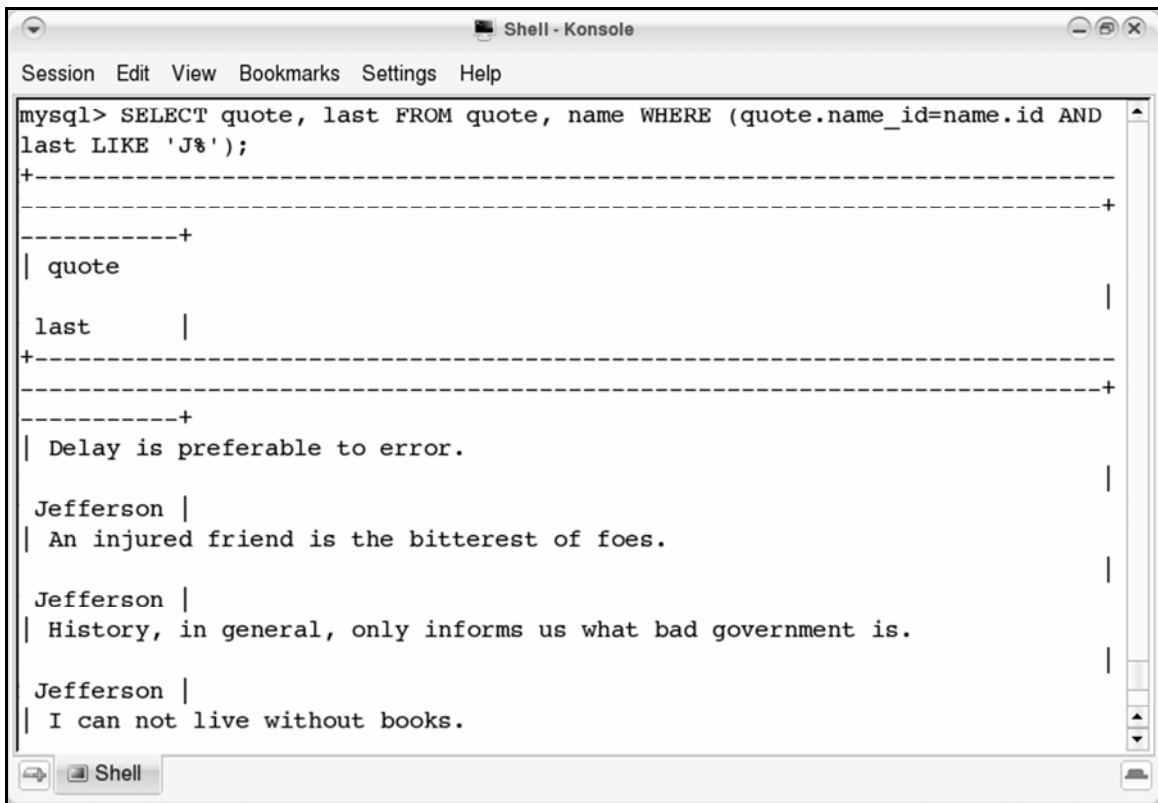
```
SELECT quote,last FROM quote,name  
WHERE (quote.name_id=name.id AND name.id=2);
```

3. Type:

```
SELECT quote,last FROM quote,name ►►  
WHERE (quote.name_id=name.id ►►  
AND last LIKE 'J%');
```

then press **ENTER**.

The query results should look like this:



```
Shell - Konsole  
Session Edit View Bookmarks Settings Help  
mysql> SELECT quote, last FROM quote, name WHERE (quote.name_id=name.id AND  
last LIKE 'J%');  
+-----+  
| quote  
last      |  
+-----+  
| Delay is preferable to error.  
Jefferson |  
| An injured friend is the bitterest of foes.  
Jefferson |  
| History, in general, only informs us what bad government is.  
Jefferson |  
| I can not live without books.
```

Again, this query is similar to the ones you've been working with. The difference is in the second condition of the **WHERE** statement:

```
last LIKE 'J%'
```

LIKE compares two values; in this case, the last name of a president with a letter, J.

% is a wildcard character, that stands for any character or combination of characters.

J% stands for any name starting with a J. For instance, J% could stand for Jefferson, Jackson, or Johnson.

This query returns quotes from presidents whose last names begin with J.

4. Type:

```
\q;
```

to close the MySQL database connection.

5. Type:

```
exit
```

to exit the **Konsole** window.

Practice: Running Queries

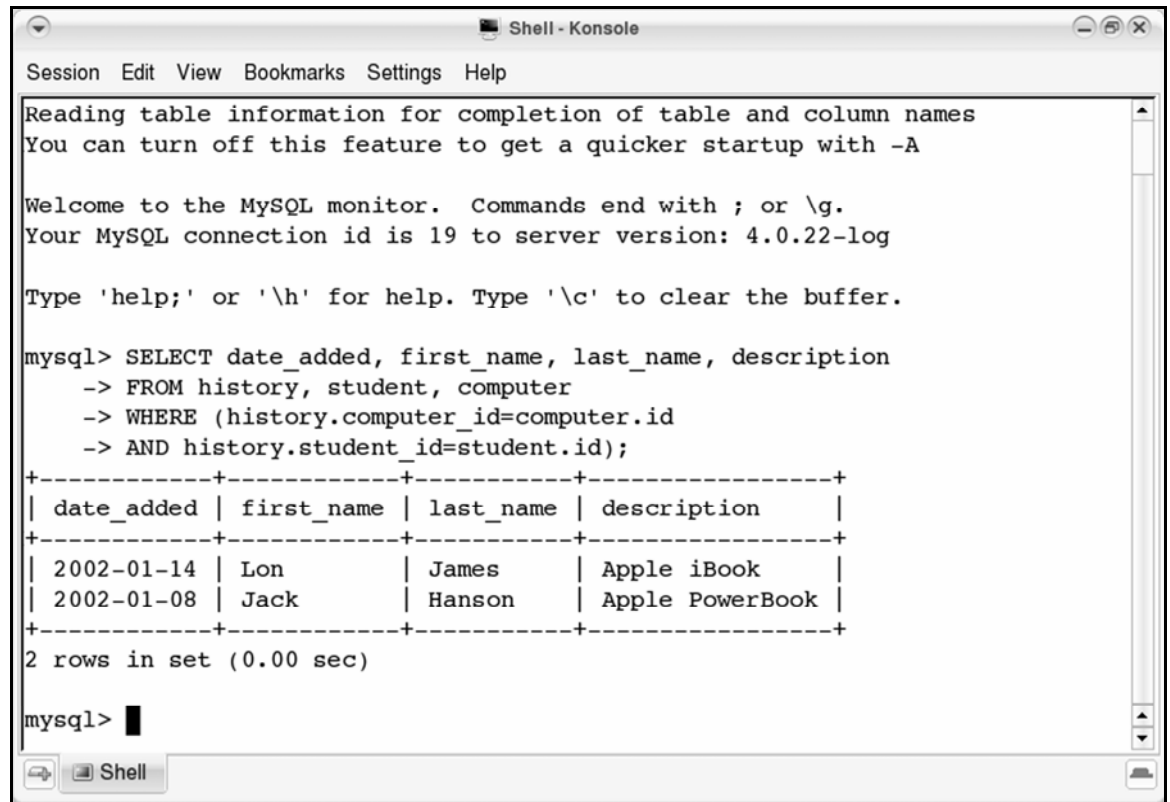
Task: The Dean wants a report from the `hardware` database that lists the type of computer each student received, and the date it was received.

Create a query that gives him this information.

- 1.** Connect to the MySQL database server, using your MySQL root password.
- 2.** Using the `hardware` database, write a query that shows four things:
 - the date the computer was given out
 - the first name of the student
 - the last name of the student
 - a description of the computer

3. Run the query.

The output should look something like this:



```
Shell - Konsole
Session Edit View Bookmarks Settings Help
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 19 to server version: 4.0.22-log

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

mysql> SELECT date_added, first_name, last_name, description
-> FROM history, student, computer
-> WHERE (history.computer_id=computer.id
-> AND history.student_id=student.id);
+-----+-----+-----+-----+
| date_added | first_name | last_name | description |
+-----+-----+-----+-----+
| 2002-01-14 | Lon       | James    | Apple iBook |
| 2002-01-08 | Jack     | Hanson   | Apple PowerBook |
+-----+-----+-----+-----+
2 rows in set (0.00 sec)

mysql> █
```

4. Close the MySQL database connection and exit the **Konsole** window.

Securing a database

In this section, you'll learn how to:

- **Add a local user**
- **Add a remote user**
- **Remove a user**
- **Restrict a user**

Add a local user

1. Open the **Konsole** window.
2. Connect to the MySQL server using your root MySQL password and go to the `mysql` database within it:

```
mysql -u root -p mysql
```

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

```
GRANT ALL PRIVILEGES ON *.* ►►  
TO mary@localhost ►►  
IDENTIFIED BY 'ship3marker';
```

then press **ENTER**.

This command string creates a new account on the MySQL server for the user `mary`. Her password is `ship3marker`.

This **GRANT** command string works like this:

- **GRANT ALL PRIVILEGES**

```
GRANT ALL PRIVILEGES ON *.*  
TO mary@localhost  
IDENTIFIED BY 'ship3marker';
```

The **GRANT** command is used to grant privileges on a database (or table) to users. In this case, you're granting all add/delete/modify privileges for the user `mary`.

- **ON *.***

```
GRANT ALL PRIVILEGES ON *.*  
TO mary@localhost  
IDENTIFIED BY 'ship3marker';
```

The **ON** command restricts the combination of databases and tables the user will have access to. Here, you're granting privileges on any (*) table in every (*) database.

If you wanted to grant rights to a specific database, you'd use something like:

```
GRANT ALL PRIVILEGES ON us_presidents.*
```

To restrict access to only the **name** table in the **us_presidents** database, you'd use:

```
GRANT ALL PRIVILEGES ON  
us_presidents.name
```

- **TO mary@localhost**

```
GRANT ALL PRIVILEGES ON *.*  
TO mary@localhost  
IDENTIFIED BY 'ship3marker';
```

TO specifies the account you are granting privileges to: a user named **mary** who can connect to **localhost**.

- **IDENTIFIED BY 'ship3marker'**

```
GRANT ALL PRIVILEGES ON *.*  
TO mary@localhost  
IDENTIFIED BY 'ship3marker';
```

This string sets the password for the user **mary**.

Add a remote user

1. Type:

```
GRANT ALL PRIVILEGES ON *.* ►►
TO marty@'%' ►►
IDENTIFIED BY 'watch4keys' ►►
WITH GRANT OPTION;
```

then press **ENTER**.

This command string is slightly different than the previous one:

- **TO marty@'%'**

```
GRANT ALL PRIVILEGES ON *.*
TO marty@'%'
IDENTIFIED BY 'watch4keys'
WITH GRANT OPTION;
```

The % wildcard allows connections on this account from any domain, not just `localhost`.

If you only wanted connections from the `visibooks.com` domain, you'd use this instead:

```
GRANT ALL PRIVILEGES ON *.*
TO marty@visibooks.com
IDENTIFIED BY 'watch4keys'
WITH GRANT OPTION;
```


- **WITH GRANT OPTION**

```
GRANT ALL PRIVILEGES ON *.* ►►  
TO marty@'%' ►►  
IDENTIFIED BY 'watch4keys' ►►  
WITH GRANT OPTION;
```

The **GRANT OPTION** sets the ability to **GRANT** privileges to other users. In other words, **marty** can create accounts for new users.

Remove a user

1. Type:

```
DELETE FROM user ►►  
WHERE (user='marty' OR user='mary');
```

then press **ENTER**.

The command string `DELETE FROM user` deletes a record from the table `user`. Like `mysql`, `user` is a table that's included in the MySQL Server database).

`WHERE (user='marty' OR user='mary')` means that a record is deleted from the table `user` **WHERE** the user is `'marty'` or `'mary'`.

Restrict a user

1. Type:

```
GRANT SELECT,INSERT >>>
ON us_presidents.* >>>
TO marty@localhost >>>
IDENTIFIED BY 'watch4keys';
```

then press **ENTER**.

This command string restores `marty` as a user of the MySQL server, but lessens his user privileges:

`marty` is now **GRANTED** permission to give only the **SELECT** and **INSERT** commands to the database `us_presidents`.

Tip: *You usually want to give users only the privileges they need. Otherwise, a user may make changes to the database that you don't want or expect.*

2. Type:

```
\q;
```

then press **ENTER**

to close the MySQL database connection.

3. Type:

`exit`

then press **ENTER**

to close the **Konsole** window.

Practice:

Securing a Database

Task: Give restricted privileges to a user so he can access the `hardware` database from your computer.

- 1.** Connect to the MySQL server using your root MySQL password and go to the `mysql` database within it.
- 2.** Create a user `fred` at `localhost` with `SELECT` and `INSERT` privileges on the database `hardware.*` with a password of `'match5pad'`.
- 3.** Close the MySQL database connection and close the **Konsole** window.

Web-enabling Databases

In this section, you'll learn how to:

- **Perform a query using PERL**
- **Join two tables using PERL**
- **Create a CGI script**
- **Write a query in a CGI script**

Perform a query using PERL

What is PERL?

Practical Extraction and Reporting Language, or PERL, is a programming language used for creating programs on Web servers.

PERL is often used to write programs that incorporate Web-based databases.

1. Open the **Konsole** window.

2. Type:

```
mkdir programs
```

then press **ENTER**.

This creates a directory within your home directory called **programs**.

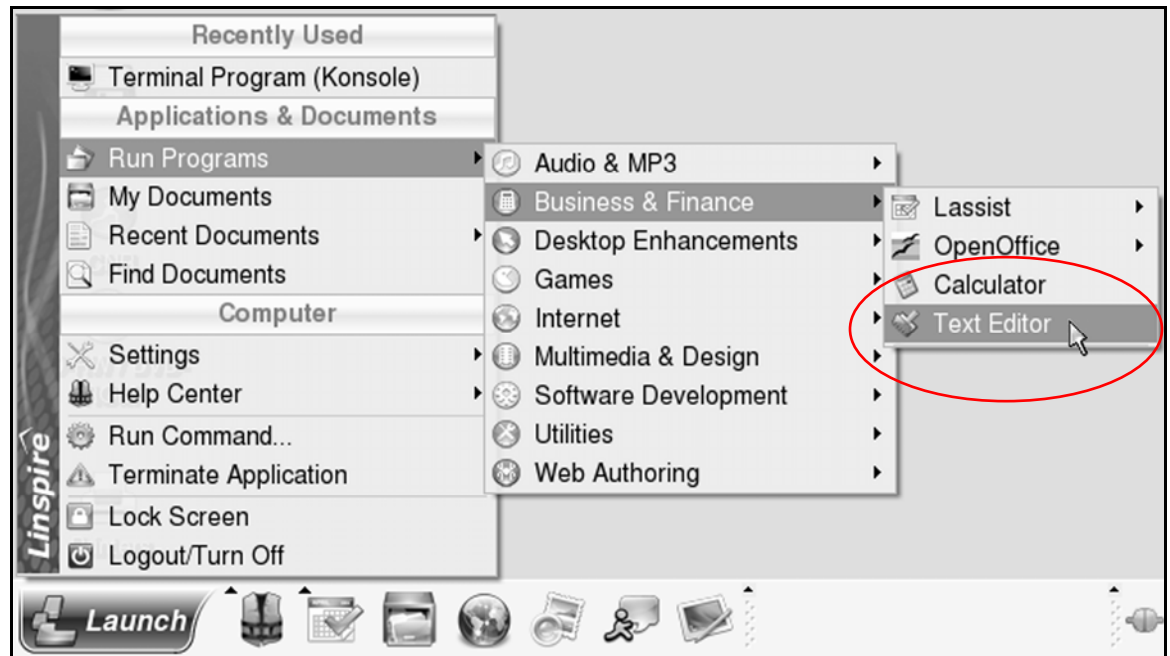
3. Type:

```
exit
```

then press **ENTER**

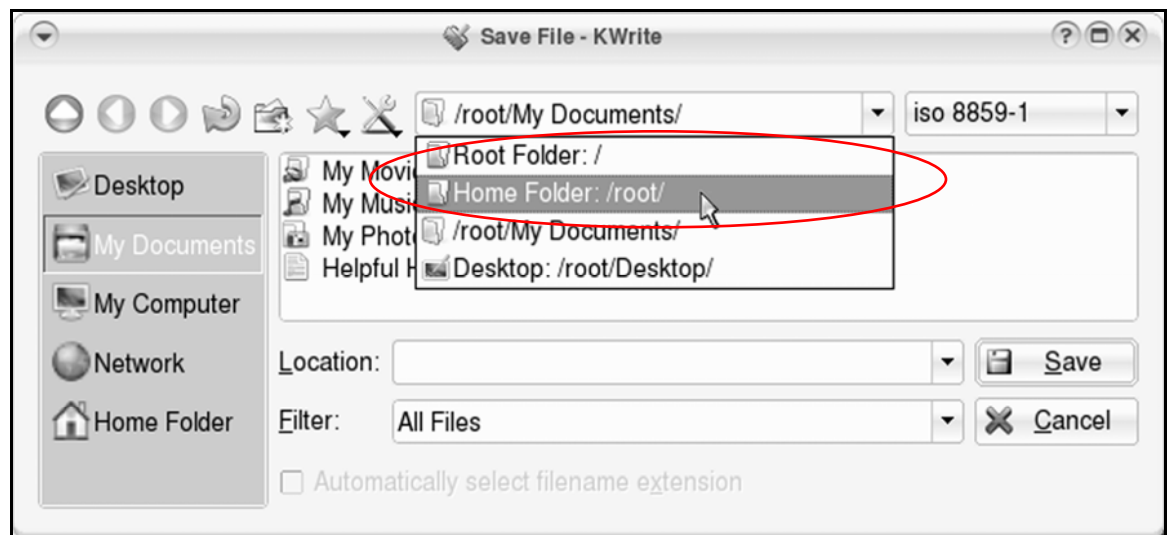
to close the **Konsole** window.

4. Click the  button, then **Run Programs**, then **Business & Finance**, then **Text Editor**.



5. When the **KWrite** window appears, click the  icon.

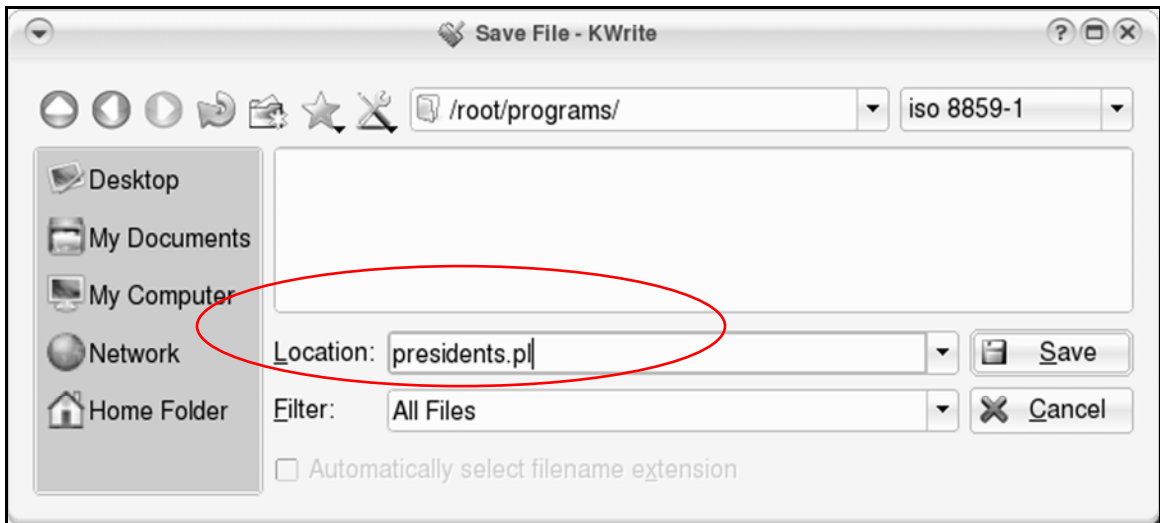
6. When the **Save File** window appears, navigate to your home directory.



7. Double-click the **programs** directory to open it.

8. In the **Location** box, type:

presidents.pl



9. Click the  **Save** button.

10. Type the code below to create the program **presidents.pl**.

Tip: *Or, go to:*

www.visibooks.com/books/mysql/presidents

in your Web browser.

*Click **Edit**, then **Select All**.*

*Click **Edit**, then **Copy**.*

*Go back to the KWrite program where **presidents.pl** is open.*

*Click **Edit**, then **Paste**.*

The code for the **presidents.pl** program should look like this:

```
#!/usr/bin/perl

use DBI;
use strict;

# database information
my $db="us_presidents";
my $host="localhost";
my $port="3306";
my $userid="marty";
my $passwd="watch4keys";
my
$connectionInfo="DBI:mysql:database=$db;$host:$port";

# make connection to database
my $dbh =
DBI->connect($connectionInfo,$userid,$passwd);

# prepare and execute query
my $query = "SELECT id,first,middle,last FROM name
ORDER BY id";
my $sth = $dbh->prepare($query);
$sth->execute();

# assign fields to variables
my ($id,$first,$middle,$last);
$sth->bind_columns(undef, \$id, \$first, \$middle,
\$last);

# output president's names listing
print "The presidents in order:\n";
while($sth->fetch()) {
    print "$first ";
    print "$middle " if ($middle);
    print "$last\n";
}

# clean up
$sth->finish();

# disconnect from database
$dbh->disconnect;
```

While this isn't a book about PERL, you should at least be familiar with how PERL works. So, let's go through the different sections of the **presidents.pl** program and describe what they do:

- `#!/usr/bin/perl`

This specifies the path to the PERL program on the computer.

- `use DBI;`
`use strict;`

The `use DBI` line means `use DataBase Interface`. It refers to the PERL module that interacts with your MySQL database. You might think of this module as a MySQL client that speaks PERL. It does most of the things the MySQL client does, but through PERL.

The `use strict` line is a matter of personal preference and programming etiquette. Variables are “containers” in a PERL script that hold specific information. In Perl, using the `strict` mode requires you to reserve all variables before they are used. The next bullet shows how this works.

- ```
database information
my $db="us_presidents";
my $host="localhost";
my $port="3306";
my $userid="marty";
my $passwd="watch4keys";
my $connectionInfo=
"DBI:mysql:database=$db;$host:$port";
```

Like the comment says (what comes after a # character is a comment—a note in the program to be read by people, not the computer), this is information about the database.

- ```
my $db="us_presidents";
```

Variables are reserved by using the `my` command – e.g. `my $db`.

Recall the `use strict` line above. Because the program uses this mode, variables cannot be used unless the `my` command is enacted first.

This is useful because if you make a mistake like misspell `$db` as `$dv` later on in your program, PERL will remind you that `$dv` does not exist and end the program.

If you were not using `strict` mode, the program would continue and the wrong MySQL database (a database with no name) would be referenced.

`us_presidents` is the name of the database we want to use upon connecting.

- `my $host="localhost";`

The address of the MySQL server.

Tip: *If the MySQL database is hosted on the same computer that will run the program, you can use 'localhost'. Otherwise, you would enter the IP address of the computer housing the MySQL database. In that case, the line would look like this:*

```
my $host="10.1.3.82";
```

Or alternatively, you could use the name of the computer:

```
my $host="mysql.visilearn.com";
```

If you don't know the IP address or name of the computer, contact your network administrator.

- `my $port="3306";`

The server port that the MySQL Server is “listening” to (the default is 3306).

What are Ports?

Ports are essentially windows into a computer. Most port-windows are closed, but sometimes a program will open one. MySQL Server, by default, opens port 3306 for access by MySQL clients.

Similarly, Web servers normally open port 80 for access by Web browsers. When you visit visibooks.com, your Web browser sends a request to port 80 at the Visibooks Web server to see if a website is available. In the case of the Visibooks Web server, the port is open and the homepage would be sent back to your Web browser.

- `my $userid="marty";`

The username you're using to connect with the MySQL server.

- `my $passwd="watch4keys";`

The password that goes with this username.

- `my $connectionInfo=
"DBI:mysql:database=$db;$host:$port";`

This last line puts the `$db`, `$host`, and `$port` variables together in the format PERL needs to "talk" to your MySQL database.

- ```
make connection to database
my $dbh = DBI->
connect($connectionInfo,$userid,$passwd);
```

Using the `$connectionInfo`, `$userid`, and `$passwd` provided, the PERL database interface (DBI) module connects to the MySQL server using the filehandle `$dbh`.

**Tip:** *A filehandle is a type of variable used to mark a place in a file. Since the `$dbh` variable is used here with a database, it can be considered a database handle – hence the name `dbh`.*

- ```
# prepare and execute query
my $query = "SELECT id,first,middle,last
FROM name ORDER BY id";
my $sth = $dbh->prepare($query);
$sth->execute();
```

`$query` creates a query to **SELECT** the `id`, `first`, `middle`, and `last` names of the presidents **FROM** the table `name`, then put them in **ORDER BY** `id` number.

Next, using a DBI statement handle (`$sth`), the query is prepared and executed. Think of handles as the paths PERL uses to communicate with different services or parts of a service.

For instance, the database handle is the path PERL uses to talk to the MySQL database. Within that path then the statement handle is used to communicate the SQL query (or statement) to MySQL Server.

- ```
assign fields to variables
my ($id,$first,$middle,$last);
$sth->bind_columns(undef, \$id, \$first,
\$middle, \$last);
```

In preparation for reading in the data from MySQL, you bind the data (in column form) to variables using the `bind_columns` command.

In other words, you are matching up the variables to the data you're requesting from MySQL Server.

- ```
# output president's names listing
print "The presidents in order:\n";
while($sth->fetch()) {
print "$first ";
print "$middle " if ($middle);
print "$last\n";
}
```

In this portion of the PERL program, you translate the data from the returned statement handle into your variables, and then print immediately to the standard output – the screen.

The `fetch` command fills up your variables with data from the database, as the while programming loop moves through the rows (records) in the database.

Some of the presidents in your list don't have a middle name, so you add an `if` statement (`if ($middle)`) to tell the program not to stop if a president doesn't have one.

The `\n` character creates a new line, acting as a carriage return while printing to the screen.

- `# clean up`
`$sth->finish();`

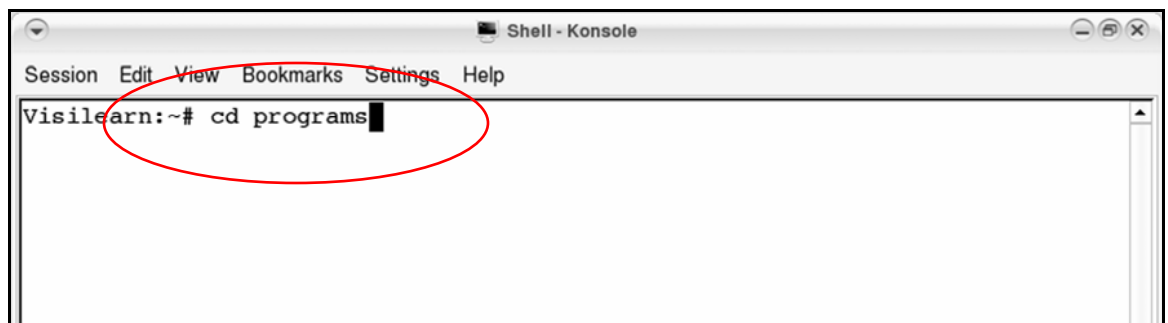
`# disconnect from database`
`$dbh->disconnect;`

Finally, you `finish` the statement handle, and `disconnect` the database handle. This ends the connection between the PERL program and the MySQL Server database.

11. Save the `presidents.pl` file, then close the KWrite program.

12. Open the **Konsole** window and type:

```
cd programs
```



then press **ENTER**.

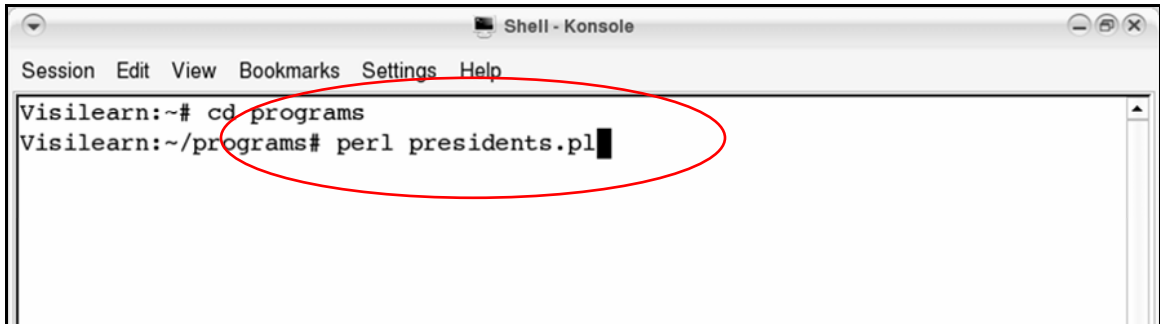
Tip: *This Linux command has two parts:*

cd tells the computer to change directory.

`programs` takes you to the programs directory.

13. Type:

```
perl presidents.pl
```

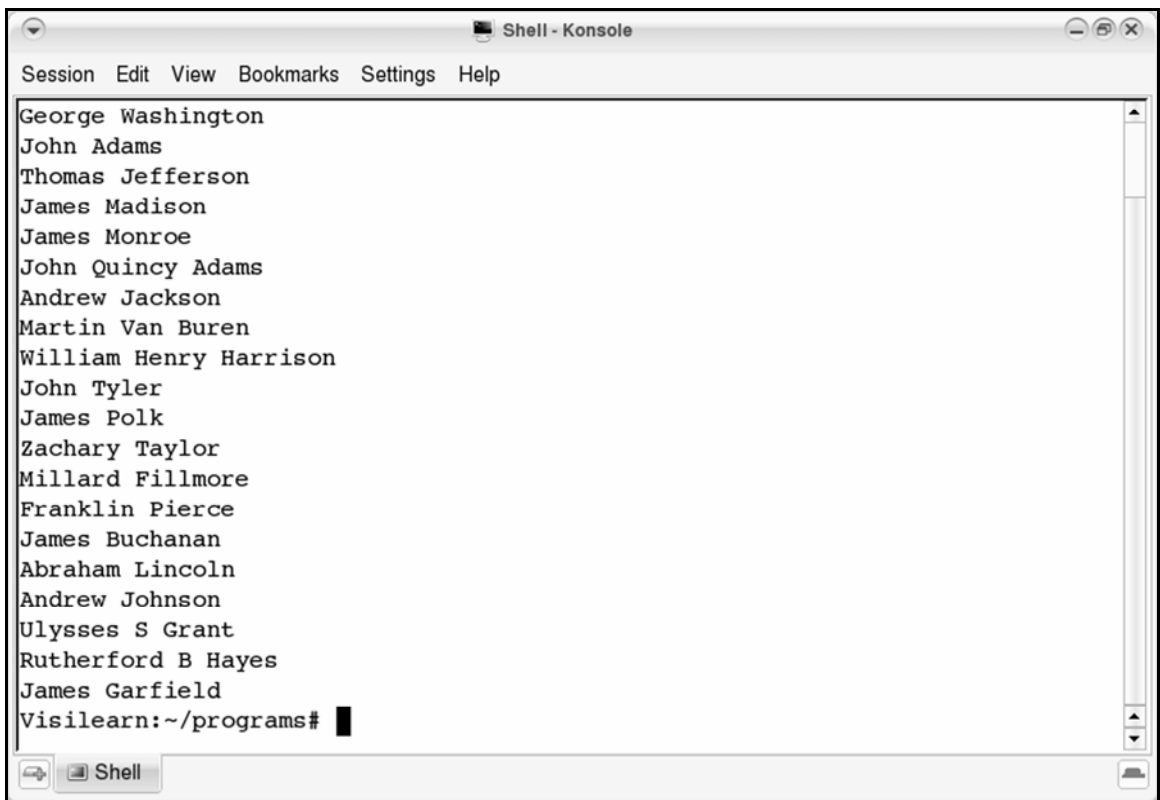


A screenshot of a terminal window titled "Shell - Konsole". The window has a menu bar with "Session", "Edit", "View", "Bookmarks", "Settings", and "Help". The terminal content shows the user "Visilearn" at the prompt "~#". The first command is "cd programs", and the second command is "perl presidents.pl", which is circled in red. The cursor is at the end of the second command.

then press **ENTER**

to run the program.

The program will query the `us_presidents` database on the MySQL Server and print out the results. Its output should look like this:




A screenshot of a terminal window titled "Shell - Konsole" showing the output of the program. The output is a list of US presidents: George Washington, John Adams, Thomas Jefferson, James Madison, James Monroe, John Quincy Adams, Andrew Jackson, Martin Van Buren, William Henry Harrison, John Tyler, James Polk, Zachary Taylor, Millard Fillmore, Franklin Pierce, James Buchanan, Abraham Lincoln, Andrew Johnson, Ulysses S Grant, Rutherford B Hayes, and James Garfield. The terminal prompt is "Visilearn:~/programs#".

14. Type:

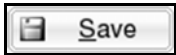
`exit`

then press **ENTER**.

Join two tables in PERL

- 1.** Open the **KWrite** program.
- 2.** Click the  icon.
- 3.** When the **Save File** window appears, navigate to the **programs** directory.
- 4.** Type:

random.pl

in the **Location** box.
- 5.** Click the  button.

Type the following code to create the program **random.pl**.

Or, go to:

www.visibooks.com/books/mysql/random

in your Web browser, copy the code there, and paste it into **random.pl**.

```

#!/usr/bin/perl -w

use DBI;
use strict;

# database information
my $db="us_presidents";
my $host="localhost";
my $port="3306";
my $userid="marty";
my $passwd="watch4keys";
my
$conectionInfo="DBI:mysql:database=$db;$host:$port";

# find a random number between 1 and 20
my $random=int(rand 20) + 1;

# make connection to database
my $dbh = DBI->
connect($conectionInfo,$userid,$passwd);

# prepare and execute query
my $query = "SELECT first,middle,last,quote
FROM quote,name
WHERE quote.id=$random
AND quote.name_id=name.id;";

my $sth = $dbh->prepare($query);
$sth->execute();

# assign fields to variables
my ($first,$middle,$last,$quote);
$sth->bind_columns(undef, \$first, \$middle, \$last,
\$quote);

# output random quote
while($sth->fetch()) {
print "\"$quote\"\n";
    print " - $first ";
print "$middle " if ($middle);
print "$last\n";
}

$sth->finish();

# disconnect from database
$dbh->disconnect;

```

6. Save **random.pl** file, then close the **KWrite** program.

The main difference between this program and the **presidents.pl** program lies in **\$query**.

In this program, instead of selecting data only from the **names** table, the query selects data from two tables: **name** and **quote**:

```
FROM quote,name
```

It returns a president's name and his quote:

```
print "\"$quote\"\n";  
print " - $first ";  
print "$middle " if ($middle);  
print "$last\n";
```

As its name suggests, **random.pl** selects a president's quote at random:

```
FROM quote,name  
WHERE quote.id=$random
```

7. Open the **Konsole** window and type:

```
cd programs
```

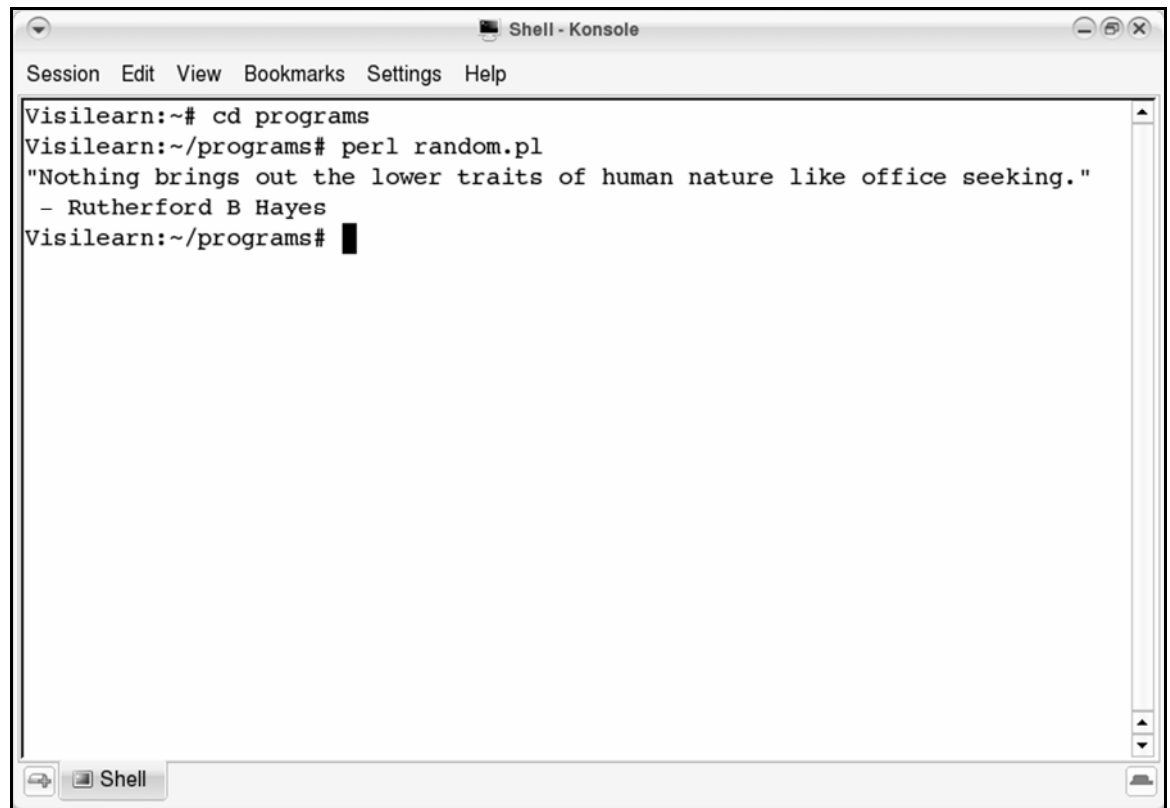
then press **ENTER**.

8. Type:

```
perl random.pl
```

then press **ENTER**.

The output should look like this, but the quote may be different:



The screenshot shows a terminal window titled "Shell - Konsole". The terminal content is as follows:

```
Session Edit View Bookmarks Settings Help
Visilearn:~# cd programs
Visilearn:~/programs# perl random.pl
"Nothing brings out the lower traits of human nature like office seeking."
- Rutherford B Hayes
Visilearn:~/programs# █
```

9. Type:

```
exit
```

then press **ENTER**.

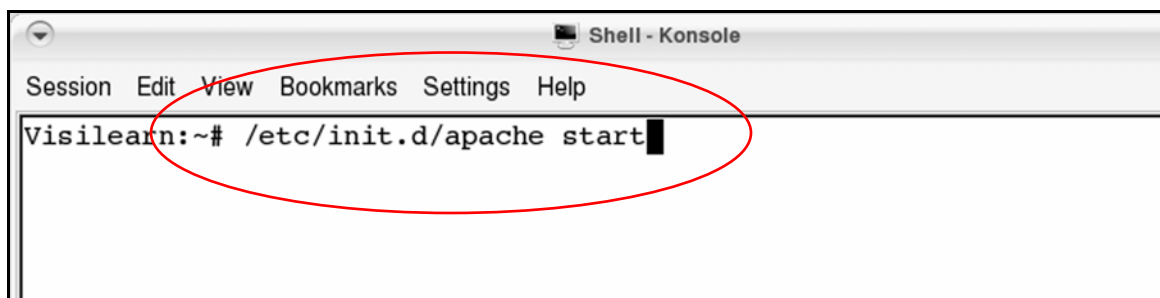
Create a CGI script

1. Open the **Konsole** window.

Tip: *If your terminal prompt is followed by a \$, login as the Root user. Type **su** and press **ENTER**. Type your Root password and press **ENTER** again.*

2. Type:

```
/etc/init.d/apache start
```

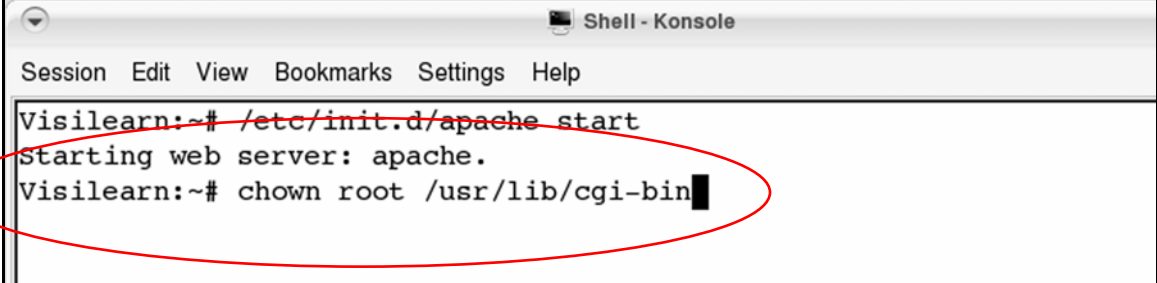


then press **ENTER**.

This starts the Apache web server program on your Linux computer.

3. Next, type:

```
chown root /usr/lib/cgi-bin
```

A screenshot of a terminal window titled "Shell - Konsole". The window has a menu bar with "Session", "Edit", "View", "Bookmarks", "Settings", and "Help". The terminal output shows the following sequence of commands and responses:

```
Visilearn:~# /etc/init.d/apache start
starting web server: apache.
Visilearn:~# chown root /usr/lib/cgi-bin
```

The last line, "Visilearn:~# chown root /usr/lib/cgi-bin", is circled in red. A cursor is visible at the end of this line.

then press **ENTER**.

This runs the **change file owner** command.

Let's look at each part of this command:

- **chown root**

This asks the computer to change the file (or directory) owner to the user known as **root**. If you are not running as the root user, replace “root” with “yourusername”.

```
chown yourusername /usr/lib/cgi-bin
```

- `/usr/lib/cgi-bin`

This is the directory that `root` will have ownership of.

The `/usr/lib/cgi-bin` directory is where all of the CGI scripts are in a default installation of the Linux computer's Apache Web server software.

After running this command, the assigned user has add/delete/modify permissions on this directory. This is not to be taken lightly! Be careful not to remove the `cgi-bin` directory, or your Apache Web server may not be able to run Web-enabled programs.

Tip: *If you had to login as the Root user in step 1, type:*

`exit`

*then press **ENTER***

to relinquish your Super User permissions.

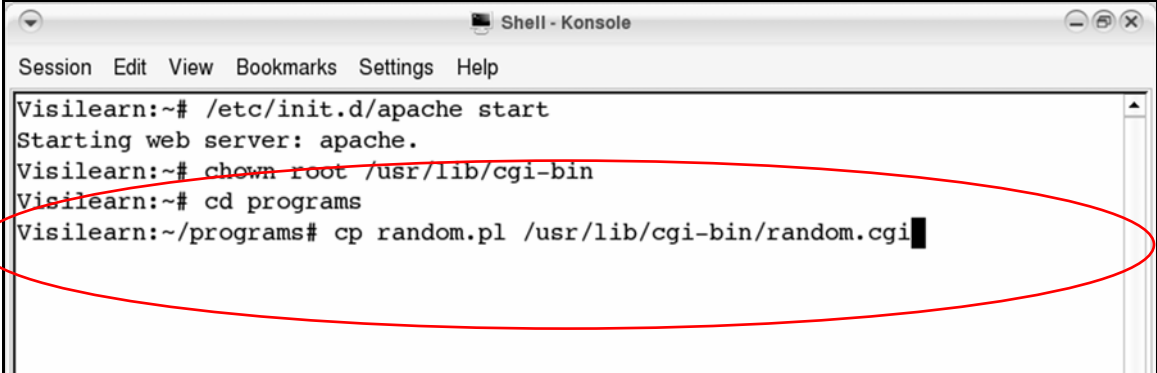
4. Type:

`cd programs`

then press **ENTER**.

5. Type:

```
cp random.pl /usr/lib/cgi-bin/random.cgi
```

A screenshot of a terminal window titled "Shell - Konsole". The window has a menu bar with "Session", "Edit", "View", "Bookmarks", "Settings", and "Help". The terminal output shows the following sequence of commands and responses:

```
Visilearn:~# /etc/init.d/apache start
Starting web server: apache.
Visilearn:~# chown root /usr/lib/cgi-bin
Visilearn:~# cd programs
Visilearn:~/programs# cp random.pl /usr/lib/cgi-bin/random.cgi
```

The last line of the terminal output is circled in red.

then press **ENTER**.

This command string will copy the `random.pl` program to the `/usr/lib/cgi-bin/` directory and at the same time rename it to `random.cgi`.

The `cgi-bin` directory is where you'll place programs, or "scripts," to be run by the Apache web server.

Regardless of what language the program is actually written in (it could be Perl, PHP, C++, or another language), `random.cgi` is referred to as a CGI script.

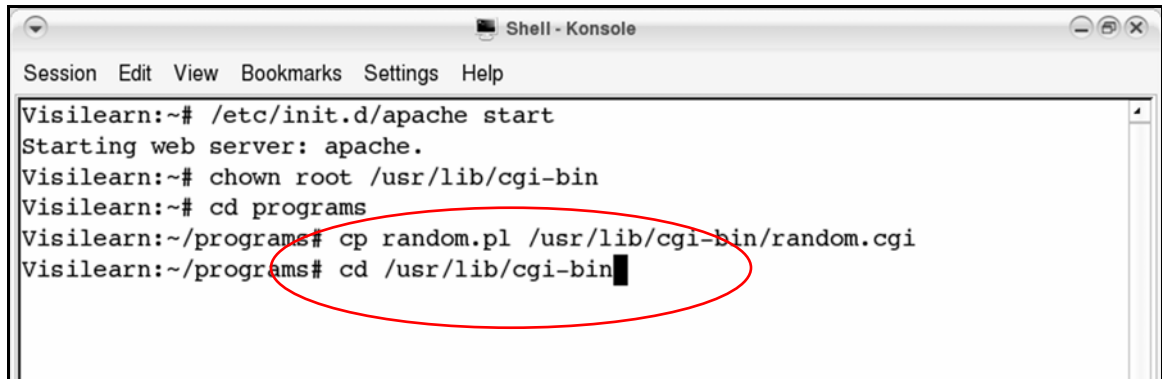
CGI stands for Common Gateway Interface, a common way to run scripts of different languages on a Web server.

The Apache Web server program on your Linux computer will run the scripts in the `cgi-bin` directory. For instance, the `random.cgi` script is now found at:

```
http://localhost/cgi-bin/random.cgi
```

6. Type:

```
cd /usr/lib/cgi-bin
```



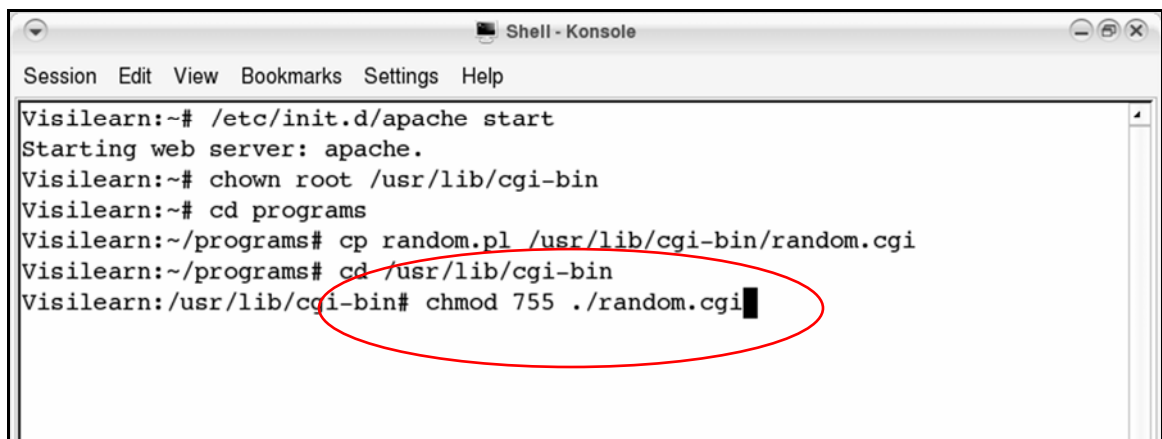
```
Shell - Konsole
Session Edit View Bookmarks Settings Help
Visilearn:~# /etc/init.d/apache start
Starting web server: apache.
Visilearn:~# chown root /usr/lib/cgi-bin
Visilearn:~# cd programs
Visilearn:~/programs# cp random.pl /usr/lib/cgi-bin/random.cgi
Visilearn:~/programs# cd /usr/lib/cgi-bin
```

then press **ENTER**.

This puts you into the `cgi-bin` directory.

7. Type:

```
chmod 755 ./random.cgi
```



```
Shell - Konsole
Session Edit View Bookmarks Settings Help
Visilearn:~# /etc/init.d/apache start
Starting web server: apache.
Visilearn:~# chown root /usr/lib/cgi-bin
Visilearn:~# cd programs
Visilearn:~/programs# cp random.pl /usr/lib/cgi-bin/random.cgi
Visilearn:~/programs# cd /usr/lib/cgi-bin
Visilearn:~/usr/lib/cgi-bin# chmod 755 ./random.cgi
```

then press **ENTER**.

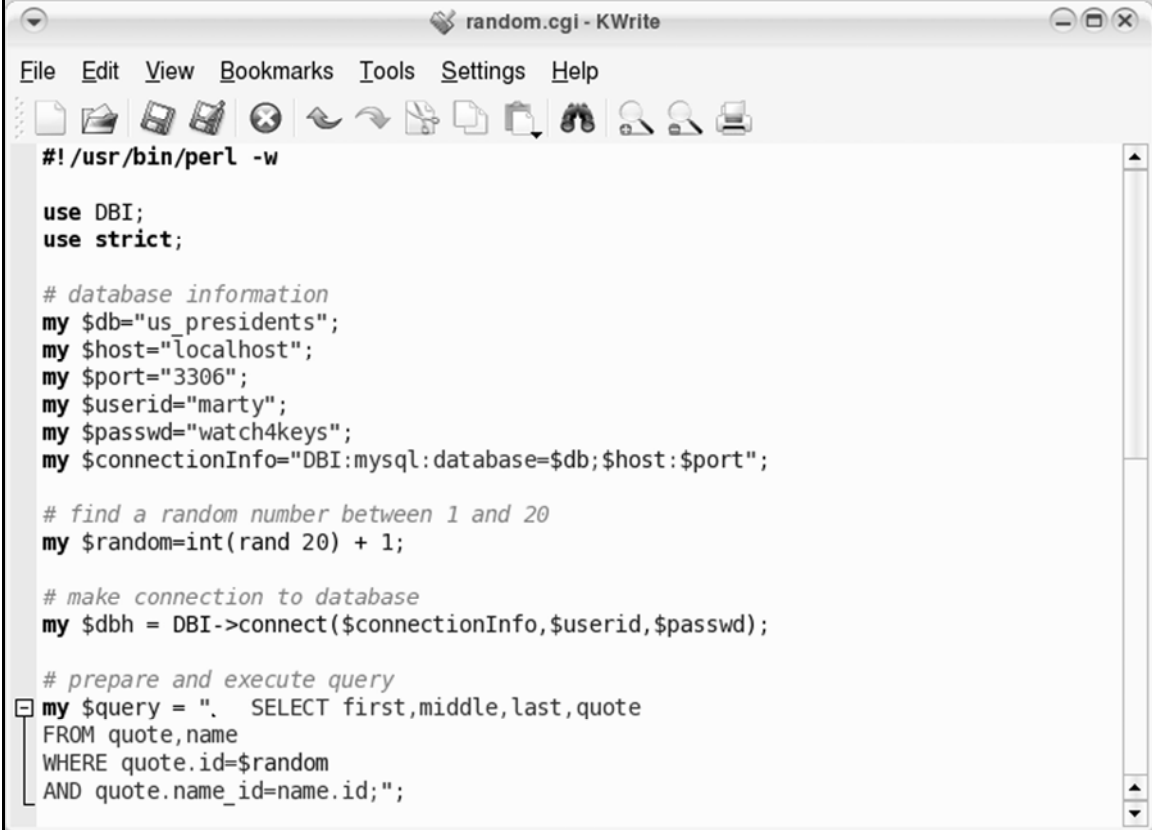
The `chmod` command is particular to Linux and Unix. It's used to change the permissions of a file.

The 755 setting allows people outside this server to execute the script. They can run the script remotely by typing its address into a Web browser.

8. Open **KWrite**, then open **random.cgi**.

Tip: *Navigate to the `/usr/lib/cgi-bin` directory.*

It should show up in the **KWrite** window:



```
#!/usr/bin/perl -w

use DBI;
use strict;

# database information
my $db="us_presidents";
my $host="localhost";
my $port="3306";
my $userid="marty";
my $passwd="watch4keys";
my $connectionInfo="DBI:mysql:database=$db;$host:$port";

# find a random number between 1 and 20
my $random=int(rand 20) + 1;

# make connection to database
my $dbh = DBI->connect($connectionInfo,$userid,$passwd);

# prepare and execute query
my $query = ". SELECT first,middle,last,quote
FROM quote,name
WHERE quote.id=$random
AND quote.name_id=name.id;";
```

9. Edit `random.cgi` to look like this:

```
#!/usr/bin/perl -w

use DBI;

use CGI qw(:standard);

use strict;

# database information
my $db="us_presidents";
my $host="localhost";
my $port="3306";
my $userid="marty";
my $passwd="watch4keys";
my
$connectionInfo="DBI:mysql:database=$db;$host:$port";

# find a random number between 1 and 20
my $random=int(rand 20) + 1;

# make connection to database
my $dbh = DBI-
>connect($connectionInfo,$userid,$passwd);

# prepare and execute query
my $query = " SELECT first,middle,last,quote
FROM quote,name
WHERE quote.id=$random
AND quote.name_id=name.id;";

my $sth = $dbh->prepare($query);
$sth->execute();

# assign fields to variables
my ($first,$middle,$last,$quote);
$sth->bind_columns(undef, \$first, \$middle, \$last,
\$quote);
```

```

# output random quote
while($sth->fetch()) {

print header(), start_html("Random Quotation"),
h1("Random Quotation:"),
br("\"$quote\""),br
br(" - $first ");
print "$middle " if ($middle);
print "$last\n", end_html();

}

$sth->finish();

# disconnect from database
$dbh->disconnect;

```

The edited script varies very little from the original **random.cgi** script.

It has been changed to properly display its output in a Web browser, rather than just your computer's **Konsole** window.

- 10.** Save **random.cgi**.
- 11.** In the **Konsole** window, type:

`exit`

then press **ENTER**.
- 12.** Open the Web browser.

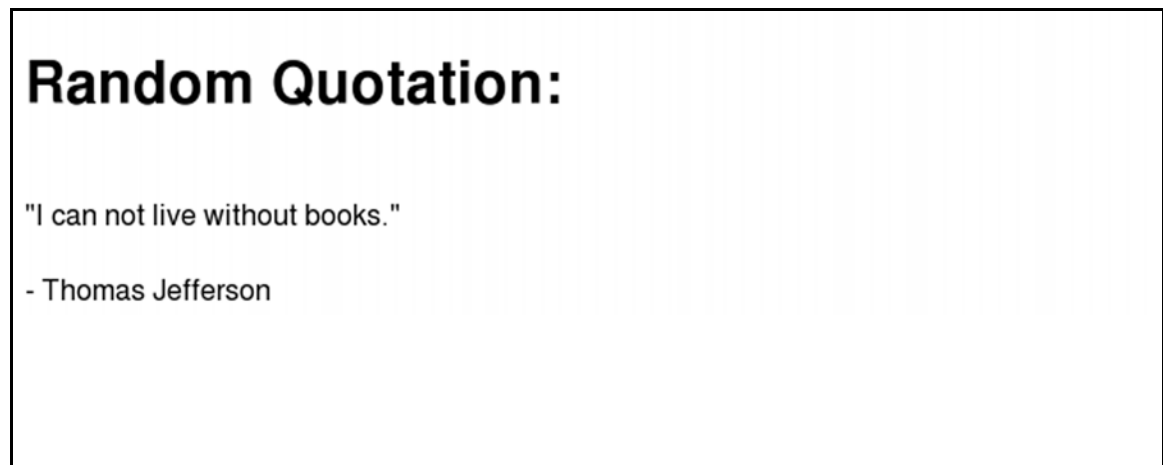
13. When the browser window appears, type in its location bar:

http://localhost/cgi-bin/random.cgi

then press **ENTER**.

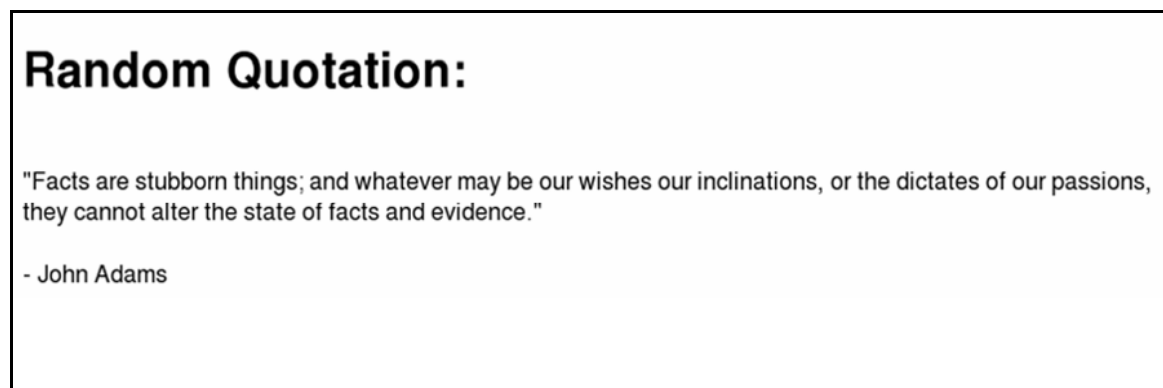
This will run the CGI script **random.cgi**.

You should see a quote in the browser:



14. Click the browser's Reload or Refresh button.

You should see a different quote:



15. Close the Web browser.

Write a query in a CGI script

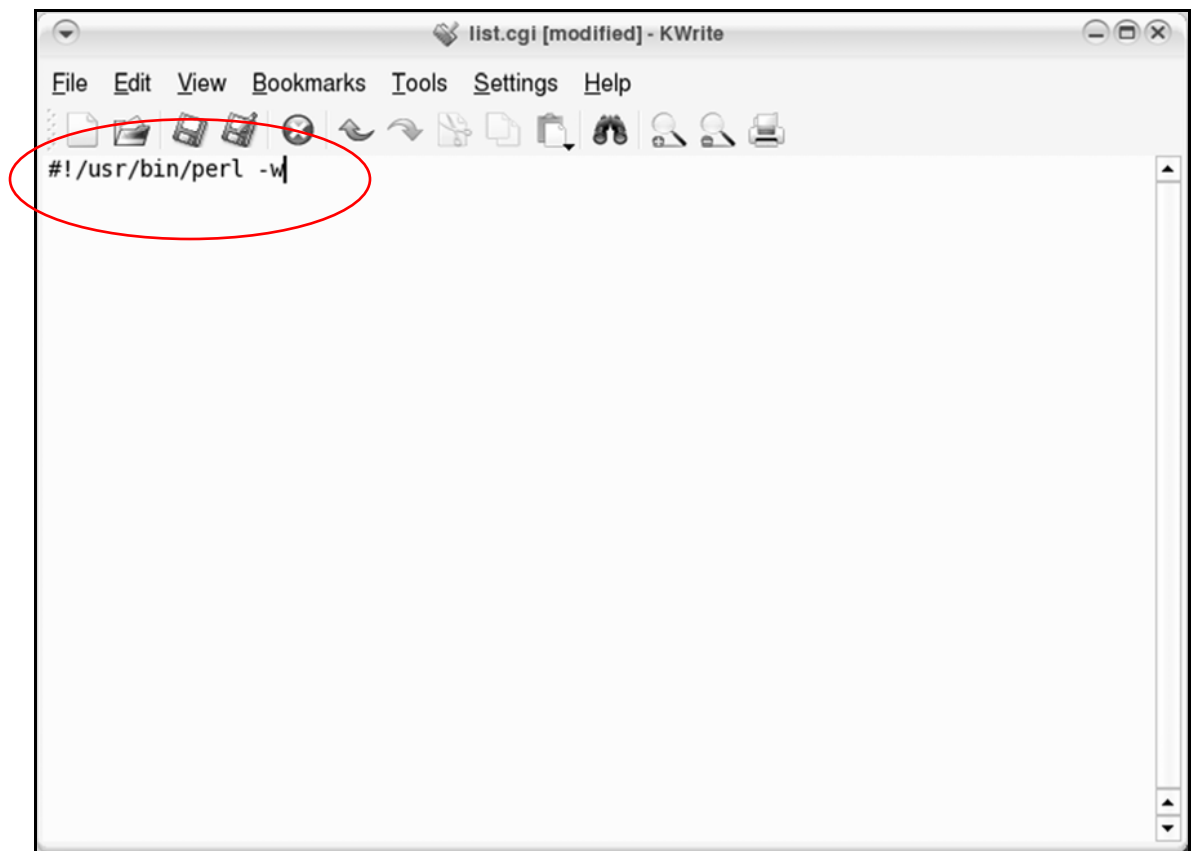
1. Create a new script named **list.cgi** in the **/usr/lib/cgi-bin** directory.

Tip: Refer back to the script **random.cgi** for guidance in writing this script.

2. The program **list.cgi** will start out as a blank file.

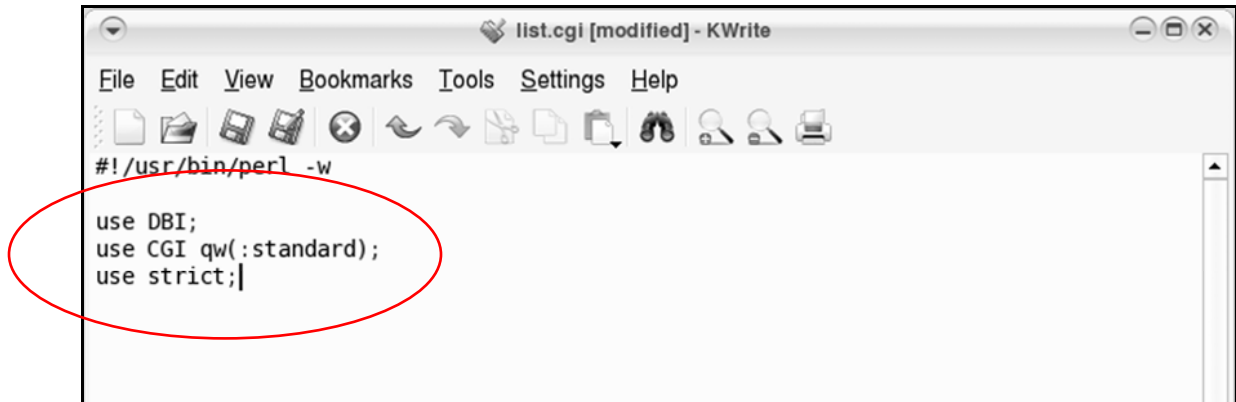
In it, first add the PERL path:

```
#!/usr/bin/perl -w
```



3. Then add the Use lines:

```
use DBI;  
use CGI qw(:standard);  
use strict;
```



4. Add the database information for your MySQL database:

```
my $db= "enter database name here" ;  
my $host= "enter mysql server name here" ;  
my $port= "enter default port here" ;  
my $userid= "enter valid user here" ;  
my $passwd= "enter user's password here" ;  
my $connectionInfo= "enter connection info here" ;
```

Tip: *You're using the database us_presidents.*

Use the MySQL server on the computer you're working on now.

Use the default port on the MySQL server.

A valid user is "marty."

*You can get standard connection info from the **random.cgi** script.*

5. Make a connection to the database:

```
my $dbh = DBI->
connect(specify connection info, user id and
password variables here, separated by commas);
```

Tip: Remember that the PERL database interface (DBI) module connects to the MySQL server with the filehandle `$dbh`, using `$connectionInfo`, `$userid`, and `$passwd`.

6. Prepare a query that selects all of the quotations and the president who said each:

```
my $query = "write your query here";
```

7. Execute the query:

```
my $sth = $dbh->prepare($query);
$sth->execute();
```

8. Assign fields to the variables:

```
my (list variables here, separated by commas);

$sth->bind_columns(undef, \specify first
variable here, \specify second variable here,
\specify third variable here, \specify fourth
variable here);
```

9. Output the quotation list:

```
print "Content-type: text/html\n\n";

print "<h1>A list of presidential
quotations:</h1>\n";

while($sth->fetch()) {

print "specify variable for president's first
name here";

print "$middle " if ($middle);

print "specify variable for president's last
name here ";

print "\"specify variable for quotation
here<p>\n";
}
```

Tip: The `print` command uses quotation marks to specify what to print: In PERL, text strings are enclosed in quotation marks.

So to make sure each president's quotation appears within quotation marks when it shows up in the browser, you put an escape character (`\`) before the quotes:

```
\"
```

This ensures that the quotation marks will appear in the browser:

Random Quotation:

"I can not live without books."

- Thomas Jefferson

10. Disconnect from the database:

```
$sth->finish();
```

```
$dbh->disconnect;
```

11. Set the permissions for `list.cgi` to 755.

12. View the **list.cgi** program in your web browser.

Its output should look like this:

A list of presidential quotations:

George Washington: "I cannot tell a lie."

George Washington: "To be prepared for war is one of the most effective means of preserving peace."

John Adams: "Liberty can not be preserved without general knowledge among people."

John Adams: "Facts are stubborn things; and whatever may be our wishes our inclinations, or the dictates of our passions, they cannot alter the state of facts and evidence."

Thomas Jefferson: "Delay is preferable to error."

Thomas Jefferson: "An injured friend is the bitterest of foes."

Thomas Jefferson: "History, in general, only informs us what bad government is."

Practice:

Web-enabling Databases

Task: Create scripts that make data in the `hardware` database accessible on the web.

Display the *computer* table

- 1.** Open the **Konsole** window.
- 2.** Type:

```
cd /usr/lib/cgi-bin
```

then press **ENTER**.
- 3.** Open **KWrite** and save the blank file as **computers.cgi** in the `cgi-bin` directory at `/usr/lib/cgi-bin`.

After it's finished, **computers.cgi** will display the contents of the `computer` table in a Web browser.

4. Go to:

www.visibooks.com/books/mysql/computers

copy the code for **computers.cgi**, and paste it in the **KWrite** window.

The code should look like this:

```
#!/usr/bin/perl -w

use DBI;
use CGI qw(:standard);
use strict;

# database information
my $db="hardware";
my $host="localhost";
my $port="3306";
my $userid="fred";
my $passwd="match5pad";
my
$connectionInfo="DBI:mysql:database=$db;$host:$port";

# make connection to database
my $dbh = DBI-
>connect($connectionInfo,$userid,$passwd);

# prepare and execute query
my $query = "SELECT description FROM computer ORDER BY
description";
my $sth = $dbh->prepare($query);
$sth->execute();

# assign fields to variables
my ($description);
$sth->bind_columns(undef, \$description);

# output hardware list
print header(), start_html("Hardware Inventory"),
h1("Hardware inventory:");
print "<TABLE BORDER=1>";
while($sth->fetch()) {
    print "<TR><TD>$description</TD></TR>";
}
print "</TABLE>";
print end_html();

$sth->finish();

# disconnect from database
$dbh->disconnect;
```

5. Open the **Konsole** window, then type:

```
chmod 755 ./computers.cgi
```

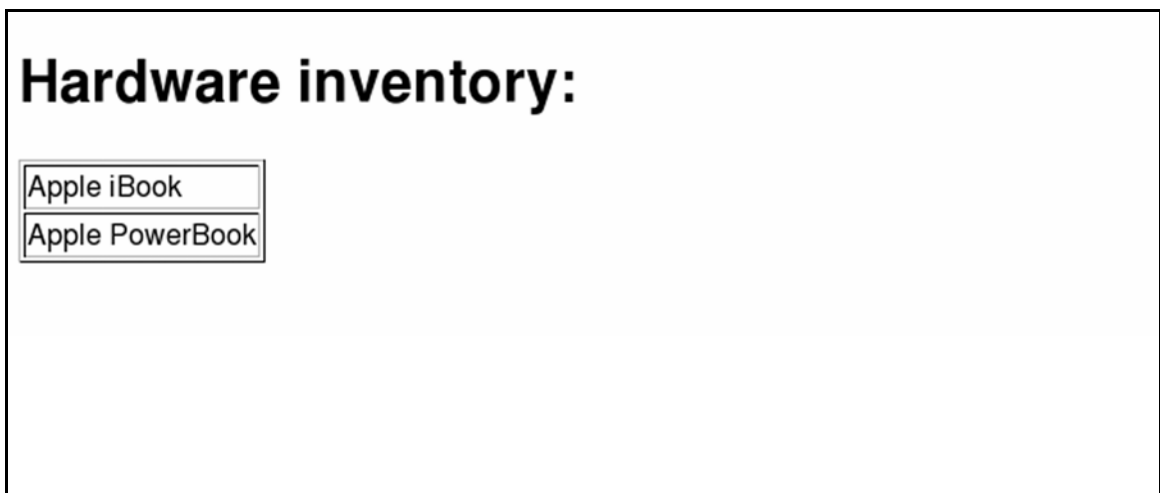
then press **ENTER**.

6. Open the Web browser and type in its location bar:

```
http://localhost/cgi-bin/computers.cgi
```

then press **ENTER**.

You should see this in the browser window:



Display the *students* table

1. Create another CGI script: **students.cgi**.

Save it in the `cgi-bin` directory.

Go to:

www.visibooks.com/books/mysql/students

copy the code for **students.cgi**, and paste it in the **KWrite** window for **students.cgi**.

You supply the database information.

```

#!/usr/bin/perl -w

use DBI;
use CGI qw(:standard);
use strict;

# database information

List database information here, using the my $db, my $host, my $port,
my $userid, my $passwd, and my $connectionInfo variables.

# make connection to database
my $dbh = DBI->connect($connectionInfo,$userid,$passwd);

# prepare and execute query
my $query = "SELECT first_name,last_name FROM student ORDER
BY last_name";
my $sth = $dbh->prepare($query);
$sth->execute();

# assign fields to variables
my ($first_name,$last_name);
$sth->bind_columns(undef, \$first_name, \$last_name);

# output student list
print header(), start_html("Student List"), h1("Student
list:");
print "<TABLE BORDER=1>";
while($sth->fetch()) {
    print "<TR><TD>$firstName $lastName</TD></TR>";
}
print "</TABLE>";
print end_html();

$sth->finish();

# disconnect from database
$dbh->disconnect;

```

2. Type:

```
chmod 755 ./students.cgi
```

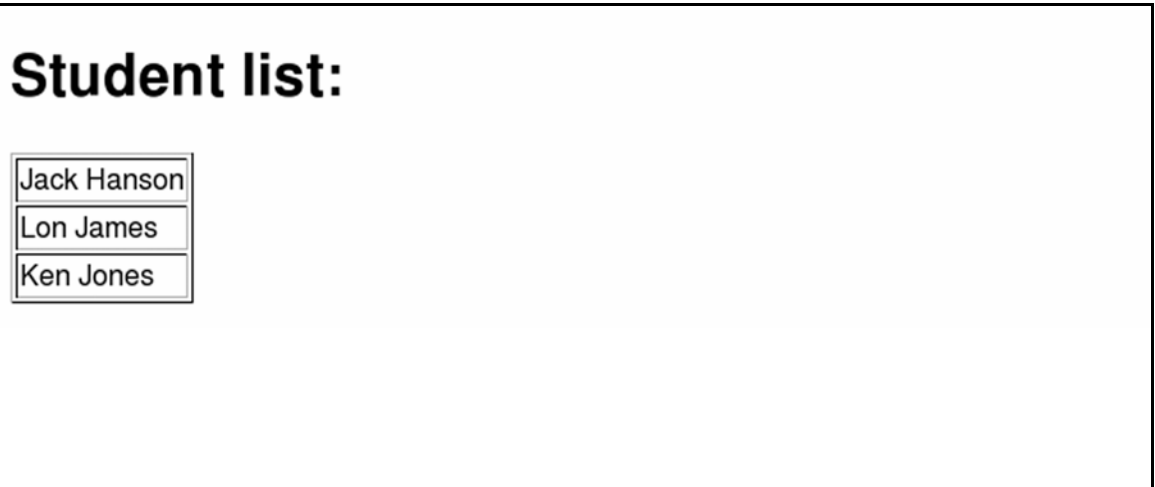
then press **ENTER**.

3. In the browser's Location bar, type:

```
http://localhost/cgi-bin/students.cgi
```

then press **ENTER**.

You should see this in the browser window:



Display the *history* table

- 1.** Create another CGI script: **history.cgi**.

Save it in the `cgi-bin` directory.

- 2.** Go to:

www.visibooks.com/books/mysql/history

copy the code for **history.cgi**, and paste it in the **history.cgi** file using KEdit.

history.cgi combines the functions of the first two scripts, listing:

- the date each hardware item was given out
- its description
- the first and last name of the student
- any comments

The code should look like this:

```
#!/usr/bin/perl -w

use DBI;
use CGI qw(:standard);
use strict;

# database information
my $db="hardware";
my $host="localhost";
my $port="3306";
my $userid="fred";
my $passwd="match5pad";
my $connectionInfo="DBI:mysql:database=$db;$host:$port";

# make connection to database
my $dbh = DBI->connect($connectionInfo,$userid,$passwd);

# prepare and execute query
my $query = "  SELECT
date_added,description,first_name,last_name,comments
FROM history,computer,student
WHERE (history.student_id=student.id
AND history.computer_id=computer.id)
ORDER BY history.id;";
my $sth = $dbh->prepare($query);
$sth->execute();

# assign fields to variables
my ($date,$description,$firstName,$lastName,$comments);
$sth->bind_columns( undef,
  \$date,
  \$description,
  \$firstName,
  \$lastName,
  \$comments);
```



```

# output history list
print header(), start_html("History"), h1("History:"), br;
print <<HTML;
<TABLE BORDER=1>
<TR>
<TD align=center>Date</TD>
<TD align=center>Description</TD>
<TD align=center>Name</TD>
<TD align=center>Comments</TD>
</TR>
HTML
while($sth->fetch()) {
    print <<HTML;
    <TR>
    <TD>$date</TD>
    <TD>$description</TD>
    <TD>$firstName $lastName</TD>
    <TD>$comments</TD>
    </TR>
    HTML
}
print "</TABLE>";
print end_html();

$sth->finish();

# disconnect from database
$dbh->disconnect;

```

3. Set the permissions for **history.cgi** to 755 .

4. View the script in a Web browser:

<http://localhost/cgi-bin/history.cgi>

You should see this in the browser window:

History:

Date	Description	Name	Comments
2002-01-08	Apple PowerBook	Jack Hanson	Cool new laptop
2002-01-14	Apple iBook	Lon James	Wireless web

Add students to database

- 1.** In **KWrite**, create two more CGI scripts: **student_list.cgi** and **student_insert.cgi**.

You can copy their code at:

www.visibooks.com/books/mysql/studentstuff

- 2.** Use the **Konsole** window to give both these scripts 755 permissions.

- 3.** In the browser, go to:

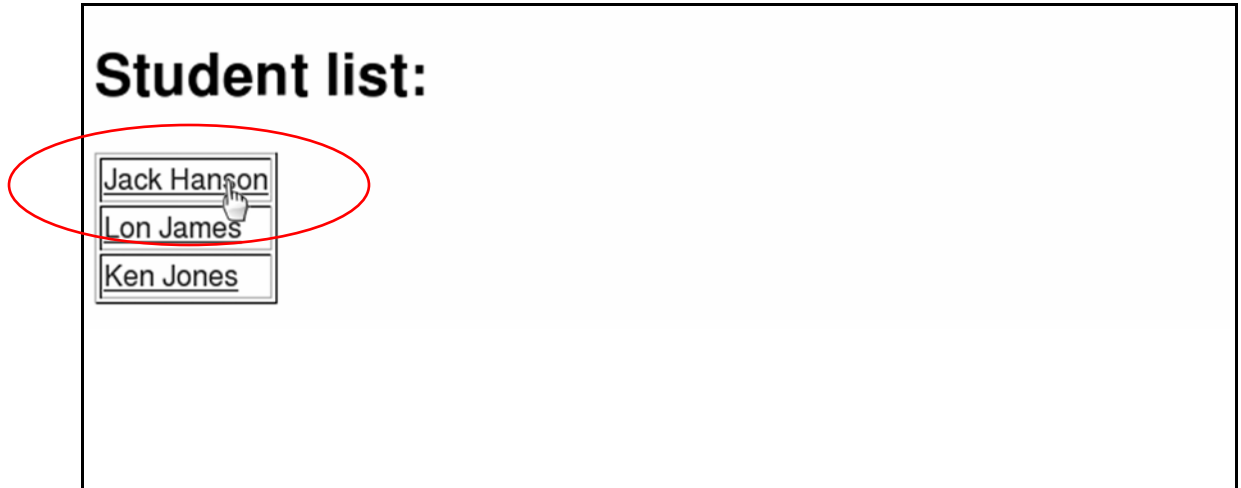
http://localhost/cgi-bin/student_list.cgi

When the script runs, you'll see a list of student names in the browser window, similar to that of the **students.cgi** script you created earlier.

In this script however, the names are links:



4. Click the Jack Hanson link.



The web address in the browser's Location bar should be:

`http://localhost/cgi-bin/student_list.cgi?studentID=1&_first=Jack&last=Hanson`

Everything coming after the `?` is passed along to your Perl script. In this case, it contains three variables:

```
# assign URL-encoded variables
my $a = new CGI;
my $studentIDinput = param("studentID");
my $firstinput = param("first");
my $lastinput) = param("last");
```

This causes the following `if` statement to become true...

```
# if the studentID is not NULL, print out the hardware list
if ($studentIDinput) {
```

...which adds a new section to your web page, listing the hardware for Jack Hanson.

After you clicked on Jack Hanson's name, the page should look like this:

Hardware list for Jack Hanson:

Apple PowerBook	2002-01-08
-----------------	------------

Student list:

Jack Hanson
Lon James
Ken Jones

5. In the browser's Location bar, type:

http://localhost/cgi-bin/student_insert.cgi

then press **ENTER**.

- 6.** Using the Student Insert form, add a student named “Fred Herman.”

Add Student

First name:

Last name:

Student list:

Jack Hanson
Lon James
Ken Jones

If you scroll down, you will see that the student list now includes **Fred Herman**:

First name:

Last name:

Student list:

Jack Hanson
Lon James
Ken Jones
Fred Herman

Search for students in the database

- 1.** In KWrite, create a new file named **student_search.cgi** in the **/usr/lib/cgi-bin/** directory.
- 2.** Using the **student_insert.cgi** CGI script as an outline, create a script that takes a letter input from the user, then searches the **hardware** database for students whose last names begin with that letter.

For instance, a search on the letter **J** should match all students whose last names start with **J**.

Tip: *Use a wildcard in your query.*

- 3.** Save the **student_search.cgi** script, then exit KEdit.
- 4.** Set permissions on **student_search.cgi** to **755**.

- 5.** Open **student_search.cgi** in the browser, then search for students whose last names begin with **J**.

The output should look like this:

Search Results:		
<table border="1"><tr><td>Lon James</td></tr><tr><td>Ken Jones</td></tr></table>	Lon James	Ken Jones
Lon James		
Ken Jones		
Search Database		
Search: <input type="text" value="j"/>		
<input type="button" value="Submit Query"/>		
<hr/>		

SQL Commands

Items bracketed [] are optional.

For a complete list of MySQL supported commands, visit the MySQL website at <http://www.mysql.com>.

ALTER

```
ALTER TABLE table_name ADD [COLUMN] ...;
```

CREATE

```
CREATE DATABASE database_name;  
CREATE TABLE table_name;
```

DELETE

```
DELETE FROM table_name [WHERE ...];
```

DROP

```
DROP DATABASE database_name;  
DROP TABLE table_name;
```

GRANT

```
GRANT privilege ON table_name ►►  
TO user [IDENTIFIED BY 'password'] [WITH GRANT OPTION];
```

INSERT

```
INSERT [INTO] table_name VALUES (...);
```

SELECT

```
SELECT ... [FROM table_name(s)] ►►  
[WHERE ...] [GROUP BY ... ] [ORDER BY ...];
```

SET

```
SET PASSWORD FOR user@localhost = ►►  
PASSWORD("password");  
SET PASSWORD FOR user@"%.visibooks.com" = ►►  
PASSWORD("password");  
SET PASSWORD FOR user@"%" = PASSWORD("password");
```

SHOW

```
SHOW DATABASES;  
SHOW TABLES;
```

UPDATE

```
UPDATE table_name SET column_name=value [WHERE ...];
```

USE

```
USE database_name;
```

Where to Get Visibooks

If you liked using this book, and would like to use more like it, visit:

www.visibooks.com

Visibooks offers more than 30 titles on subjects such as:

- **Computer Basics**
- **Microsoft Office**
- **Desktop Linux**
- **OpenOffice.org**
- **Web Site Layout**
- **Web Graphics**
- **Web Programming**

Visibooks: the simplest way to learn
and teach computer subjects.



www.visibooks.com