cis20.2, spring 2010, lab II.1.
first mysql exercise

1 getting started

- This lab will introduce you to **MySQL**.
- Begin by logging into the class web server:
  `ssh 146.245.250.181`
- Then start up **mysql**:
  `unix-prompt$ mysql -u <username> -p`
- Prof Sklar will give you your username and password in class.
- She has created a database for you that is the same as your username.
- After you have logged in to MySQL, you can select that database:
  `mysql> use <username>;`
- Then do the steps below to become familiar with SQL and the MySQL interactive environment.

2 creating a table

The SQL command for creating a table is CREATE TABLE. The simple syntax is:

```
CREATE TABLE <tablename> ( <column-definitions> );
```

where `<column-definition>` includes the name of each column and its data type. You can define multiple columns, separated by commas.

The numeric data types are: INTEGER, SMALLINT, DECMIAL, NUMERIC, FLOAT, REAL and DOUBLE PRECISION. The date and time data types are: DATETIME, DATE, TIMESTAMP, TIME and YEAR. The string data types are: CHAR, VARCHAR, BINARY, VARBINARY, BLOB, TEXT, ENUM and SET. For detailed descriptions of each type, see the MySQL online documentation.

Here is an example CREATE TABLE statement:

```
CREATE TABLE birthday (thename VARCHAR(30) NOT NULL PRIMARY KEY, themonth INT,
   theday INT, theyear INT);
```

Type the command all on one line and hit enter at the end. Note that I capitalize the SQL keywords, for clarity.

Note that "year" is a SQL data type, so I use the variable name "theyear" instead.

The command `DESCRIBE <table-name>` (also abbreviated `DESC`) will list the definition of a table, for example: `DESC birthday`.

DO THIS:

1. In your database, create the birthday table, as above.
2. Use the `DESC` command to check that you created the table correctly. You should get something like this:

```
mysql> DESC birthday;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| thename  | varchar(30) |      | PRI |         |       |
| themonth | int(11)     | YES  |     | NULL    |       |
| theday   | int(11)     | YES  |     | NULL    |       |
| theyear  | int(11)     | YES  |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

3 storing data in your table

Now that you have a table, you should store some data in it. The SQL command for storing data in a table is INSERT. There are two simple ways to use this command. The first way is:

```
INSERT INTO <table-name> SET <column-name> = <value>;
```

You can insert data into multiple columns with a single statement by repeating `<column-name> = <value>`, separated by commas.
The second way is:

```
INSERT INTO <table-name> (<column-name>) VALUES (<value>);
```

Again, you can insert data into multiple columns with a single statement by repeating `<column-name>` and `<value>`, separated by commas.

Here is an example INSERT statement using the first method:

```
INSERT INTO birthday SET thename='Alex', themonth='5', theday='9', theyear='1995';
```

and another example using the second method:

```
INSERT INTO birthday (thename,themonth,theday,theyear) VALUES ('Alex','5','9','1995');
```

DO THIS:

1. Insert yourself into your birthday table using the first method.
2. Insert yourself into your birthday table using the second method.

The first way should work fine. The second way should give you an error message ("Duplicate entry") because you defined your table such that the `thename` field is the PRIMARY KEY. This means that you cannot have multiple entries (i.e., records or rows) in your table that have the same value for the primary key (i.e., the `thename` field).

4 selecting information from a table

Now that you've put some data in your table, you probably want to be able to see what is there. The SQL command for displaying data in a table is SELECT. This command is extremely useful and has a large number of ways it can be used. We'll show a couple of the simplest ways here.

The very simplest way is to display everything in the table. This syntax is:

```
SELECT * FROM <table-name>;
```

For example:

```
SELECT * FROM birthday;
```

The second way is to display only some of the columns (fields) in the table. That syntax is:

```
SELECT <column-name> FROM <table-name>;
```

Note that you can "select" more than one column, by listing multiple `<column-name>`s, separated by commas. For example:

```
SELECT thename FROM birthday;
```

or

```
SELECT thename,theyear FROM birthday;
```

The third way is to display only some of the rows (entries) in the table. That syntax involves using a "where" clause to list conditions under which you would like to list rows. For example, you could decide to list only the rows of people in your birthday table who have a birthday in March. That statement would be:

```
SELECT thename FROM birthday WHERE themonth=3;
```

DO THIS:

1. Select all the columns and rows in your birthday table.
2. Select only the `thename` column, for all rows in your birthday table.
3. Select all the columns in your birthday table for those rows with a birth month in March.
4. Select only the `thename` column in your birthday table for those rows with a birth month in March.

AND THEN DO THIS:

1. Create entries in your birthday table for everyone in your family.
2. Now repeat the `SELECT` statements listed above.
3. Try selecting different months or days or years.

5 using external files for input

You can imagine that it would be quite tedious to enter a lot of data into MySQL tables by manually typing `INSERT` commands a million times. There are several ways to input data from a file, without having to type everything manually.

The simplest way is to create a *PLAIN TEXT* file (using a text editor like Notepad, NOT a word processor like Word) and fill it with `INSERT` commands. Then at the MySQL command prompt, you can "execute" that file using the `\.` command—this is the general command for executing a MySQL "script". Typically, MySQL script files are given the extension `.sql`.

For example, suppose I have a text file named **bday.sql** that contains the following:

```
INSERT INTO birthday (thename,themonth,theday,theyear) VALUES ('Ann','1','2','2000');
INSERT INTO birthday (thename,themonth,theday,theyear) VALUES ('Bev','3','4','2001');
INSERT INTO birthday (thename,themonth,theday,theyear) VALUES ('Cyd','5','6','2002');
```

Then, I can execute all these commands at once by doing this:

```
mysql> \. bday.sql
```

You can also use the `LOAD DATA` command to read entries from a formatted file, such as a comma-separated values ("csv") file exported from Excel. The syntax is:

```
LOAD DATA INFILE '<filename>' INTO TABLE <table-name>
```

Check the online MySQL manual for more details.

6 writing output to external files

If you are going to perform any analysis of data stored in your database, it is useful to be able to write the output of a query (i.e., `SELECT`) statement (or any other output, for that matter, such as the output of a `DESC` command) to a file. This is done using the `\T` ("tee") command at the MySQL prompt, as follows:

```
mysql> \T myfile.dat
```

After you enter the `\T` command, everything that is displayed will be output to the file named `myfile.dat`.

You turn off this output by using the `\t` ("notee") command:

```
mysql> \t
```

7 getting help

There are two recommended places to get help with mysql.

- Inside mysql, you can get help with the available interactive commands by typing `\?` or `\h` at the `mysql>` command prompt. This will give you a list of the mysql commands. Note that it will NOT give you a list of SQL commands.
- To get help with SQL commands, the best place to look is the on-line manual on the MySQL web site:
  `http://dev.mysql.com/doc/refman/4.1/en/`
  Note that we are currently using version 3.23.58 on the class web server. This is not the most recent version of MySQL (which is version 5), however, it is what we have available right now (it's a long story).

8 more advanced things to do

When you complete these examples, you may want to explore more advanced features of MySQL.

1. Try listing everyone in order of the year they were born:
   ```
   SELECT * FROM birthday ORDER BY theyear;
   ```

2. Try counting the number of people born in each month:
   ```
   SELECT themonth, count(*) FROM birthday GROUP BY the month ORDER BY the month;
   ```

3. Try a "join":
   - Create a second table in your database called `favorite` that contains two columns: `thename` as a primary key and `color` as a text field that will contain the favorite color of the person whose name is `thename`.

- Insert an entry in this second table for each person you have entered in your `birthday` table.
- Now join the two tables:

```
SELECT color FROM birthday,favorite
WHERE birthday.thename=favorite.thename
AND birthday.themonth=5;
```