Lesson 9
Adding Table and Fields

It is very easy to add tables to a database. But you might what to give some thought as to how many fields are in the table and what their data types are.

Text Types

CHAR( )	A fixed section from 0 to 255 characters long.
VARCHAR( )	A variable section from 0 to 255 characters long.
TINYTEXT	A string with a maximum length of 255 characters.
TEXT	A string with a maximum length of 65535 characters.
BLOB	A string with a maximum length of 65535 characters.

The ( ) brackets allow you to enter a maximum number of characters will be used in the column. VARCHAR(20)

CHAR and VARCHAR are the most widely used types. CHAR is a fixed length string and is mainly used when the data is not going to vary much in its length. VARCHAR is a variable length string and is mainly used when the data may vary in length.

CHAR may be faster for the database to process considering the fields stay the same length down the column. VARCHAR may be a bit slower as it calculates each field down the column, but it saves on memory space. Which one to ultimately use is up to you.

Using both a CHAR and VARCHAR option in the same table, MySQL will automatically change the CHAR into VARCHAR for compatibility reasons.

BLOB stands for Binary Large Object. Both TEXT and BLOB are variable length types that store large amounts of data. They are similar to a larger version of VARCHAR. These types can store a large piece of data information, but they are also processed much slower.

Number Types

TINYINT( )	-128 to 127 normal 0 to 255 UNSIGNED.
SMALLINT( )	-32768 to 32767 normal 0 to 65535 UNSIGNED.
MEDIUMINT( )	-8388608 to 8388607 normal 0 to 16777215 UNSIGNED. (millions)
INT( )	-2147483648 to 2147483647 normal 0 to 4294967295 UNSIGNED. (Billions)
BIGINT( )	-9223372036854775808 to 9223372036854775807 normal 0 to 18446744073709551615 UNSIGNED. (Trillions?)
FLOAT	A small number with a floating decimal point.
DOUBLE(,)	A large number with a floating decimal point.

The integer types have an extra option called UNSIGNED. Normally, the integer goes from an negative to positive value. Using an UNSIGNED command will move that range up so it starts at zero instead of a negative number.

DATE TYPES

DATE	YYYY-MM-DD.
DATETIME	YYYY-MM-DD HH:MM:SS.
TIMESTAMP	YYYYMMDDHHMMSS.
TIME	HH:MM:SS.

MISC TYPES

ENUM ( )	Short for ENUMERATION which means that each column may have one of a specified possible values.
SET	Similar to ENUM except each column may have more than one of the specified possible values.

ENUM is short for ENUMERATED list. This column can only store one of the values that are declared in the specified list contained in the ( ) brackets.
ENUM ('y','n')
You can list up to 65535 values in an ENUM list. If a value is inserted that is not in the list, a blank value will be inserted.

SET is similar to ENUM except SET may contain up to 64 list items and can store more than one choice.

In this example I'm going to plan out what fields that I'm going to use in My_Music.

Field Name	**Field Type**	**Description**
Mid	CHAR(4)	Used for a key field to be able to join other tables later
Format	CHAR(10)	Digital, CD, cassette or LP
Artist_name	VARCHAR(100)	The artist name or name of the group
My_notes	TEXT	Any notes I may have on condition or format
Date_acq	DATE	Date acquired

The code will be created in a two step process. The first step is nothing new and is a simple HTML form to pass the table to be created and the number of fields the table will create. Be sure it includes the following line in your HTML.

<FORM METHOD="POST" ACTION="do_showfielddef.php">

Step 1:

Step 1: Table Name and Number of fields

Table Name:

Number of Fields:

Go to Step 2

Step 2:

Check to make sure there were values entered.

```
1  <?
2   //validate important input
3  if ((!$_POST['table_name']) || (!$_POST['num_fields'])) {
4       header("Location: show_createtable.html");
5       exit;
6  }
```

The first thing that needs to be done is build a string called $form_block to create a form on the fly dynamically to input all of the fields that will be created. Don' forget to escape the quotations. On line 12 the table_name is hidden and was pasted from the previous POST and will be used in the next step.

```
8   //begin creating form for display
9   $form_block ="
10  <FORM METHOD=\"POST\" ACTION=\"do_createtable.php\">
11
12  <INPUT TYPE=\"hidden\" NAME=\"table_name\" VALUE=\"$_POST[table_name]\">
13
14  <TABLE CELLSPACING=5 CELLPADDING=5>
15  <TR>
16  <TH>FIELD NAME</TH><TH>FIELD TYPE</TH><TH>FIELD LENGTH</TH></TR>";
17
```

Start a for loop on line 19 to create all of the fields. One line 22 the $form_block is added to add one row for each field you want to create. The use of the [] in the field_name[] is an array for each field defined on the form.

Remember that an array can hold many variables in numbered slots.

```
18  //count from 0 until you reach the number of fields
19  for ($i =0;$i < $_POST['num_fields'];$i++) {
20
21      //add to the form,one row for each field
22      $form_block .="
23      <TR>
24      <TD ALIGN=CENTER><INPUT TYPE=\"text\" NAME=\"field_name[]\" SIZE=\"30\"></TD>
25      <TD ALIGN=CENTER>
26      <SELECT NAME=\"field_type[]\">
27      <OPTION VALUE=\"char \">char</OPTION>
28      <OPTION VALUE=\"date \">date</OPTION>
29      <OPTION VALUE=\"float \">float</OPTION>
30      <OPTION VALUE=\"int \">int</OPTION>
31      <OPTION VALUE=\"text \">text</OPTION>
32      <OPTION VALUE=\"varchar \">varchar</OPTION>
33      </SELECT>
34      </TD>
35      <TD ALIGN=CENTER><INPUT TYPE=\"text\" NAME=\"field_length[]\" SIZE=\"5\"></TD>
36      </TR>";
37  }
```

In the next TD cell on line 24 will create the drop down list for common data types for the fields. On line 35 a field_length[] array is created to hold the size of the field. On line 37 the loop is closed.

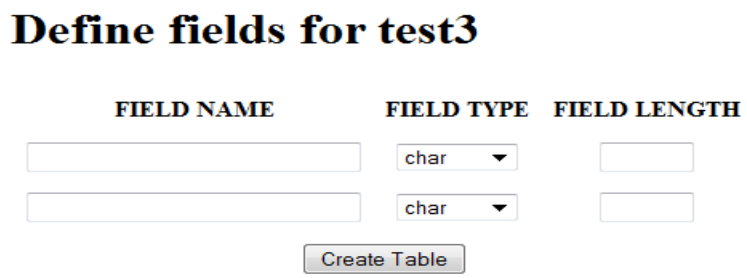
Add the final piece of the $form_block for the submit button outside of the loop and end the PHP.

```
39  //finish up the form
40  $form_block .="
41  <TR>
42  <TD ALIGN=CENTER COLSPAN=3><INPUT TYPE=\"submit\" VALUE=\"Create Table\"></TD>
43  </TR>
44  </TABLE>
45  </FORM>";
46
47  ?>
```

Add some HTML to show the $form_block. And save your file to match the Action called in the first step of the HTML.

```
49  <HTML>
50  <HEAD>
51   <TITLE>Create a Database Table:Step 2</TITLE>
52  </HEAD>
53  <BODY>
54   <H1>Define fields for <? echo "$_POST[table_name]"; ?></H1>
55
56   <? echo "$form_block"; ?>
57
58  </BODY>
59  </HTML>
```

This code will generate a page like this example.



Step 3:

The next step is to create another script to create the table and fields. The database name is hardcoded in this one, but you guys can fix that! On line 7 there is a variable created to hold the results of Mysql_select_db(). On line 10 the t able is created.

```
1  <?
2   //indicate the database you want to use
3   $db_name ="testDB";
4
5   //connect to database
6   $connection = @mysql_connect("localhost","root","") or die(mysql_error());
7   $db = @mysql_select_db($db_name,$connection) or die(mysql_error());
8
9   //start creating the SQL statement
10  $sql = @"CREATE TABLE $_POST[table_name] (";
11
```

Line 13 starts another loop to create the remainder of the SQL statement pasted for the pervious script. The count() on line 13 counts the number of elements in the array. Each of the arrays form the pervious script are passed and placed into the variable $sql or concatenated.

```
12  //continue the SQL statement for each new field
13  for ($i =0;$i < count($_POST['field_name']);$i++){
14      $sql .= $_POST['field_name'][$i]." ".$_POST['field_type'][$i];
15      if ($_POST["field_length"][$i] != "") {
16          $sql .= " (".$_POST ["field_length"][$i]."),";
17      } else {
18          $sql .= ",";
19      }
20  }
```

Lines 15 and 16 accounts for any field that does not have a length. Then the loop is closed on line 20.

On line 24 the subsrt() is used to help SQL clean up characters it can't use like a "," and ()s at the end.

```
23  //clean up the end of the string
24  $sql = substr($sql,0,-1);
25  $sql .= ")";
26
```

On line 28 the variable $result is created to hold the results of the mysql_query(). Line 31 tests if the results are true. The variables passes by post are stored in $msg.

Now just create the HTML to display the $msg.

```
27  //execute the query
28  $result = mysql_query($sql,$connection) or die(mysql_error());
29
30  //get a good message for display upon success
31  if ($result) {
32      $msg ="<P>".$_POST["table_name"]." has been created!</P>";
33  }
34
35  ?>
```

Lab 9

1. Recreate all of the HTML and scripts in this lesson 9.