Table of Contents

We are going to make a simple table; insert a few rows and then run queries against the table. This is the table you are going to create for assignment 01 with some additional rows. Since this is not part of a collection of related tables, I am going to create it in the a_testbed database.

The table will store data about animals- we will keep track of the name of each of our animals, the types of animal this is (Lion, Dog, Beetle, Centaur - all types of animals are welcome!); we want to know how much each animal costs; and we will keep the animal's date of birth and the date that we acquired the animal. Of course, we will also need an id for each of our animals.

This series of attributes lets us experiment with the basic types of data we can store in a table- character data, numeric data and dates.

Don't worry that you do not understand all of this at the start; part of learning about manipulating computer systems is learning to follow a model. And I will supply all of the essential statements.

1. Creating and populating tables using a script

It is possible to enter the SQL commands to create tables and insert rows of data into the tables by entering the SQL at the mysql> prompt in the mysql client. But it is tedious and error prone. I provide scripts to create and populate the tables we need for the assignments.

A script file is simply a file that contains directions to the computer system to do some work. The first script we will use will create a table; the second script will insert some rows into that table.

To do:

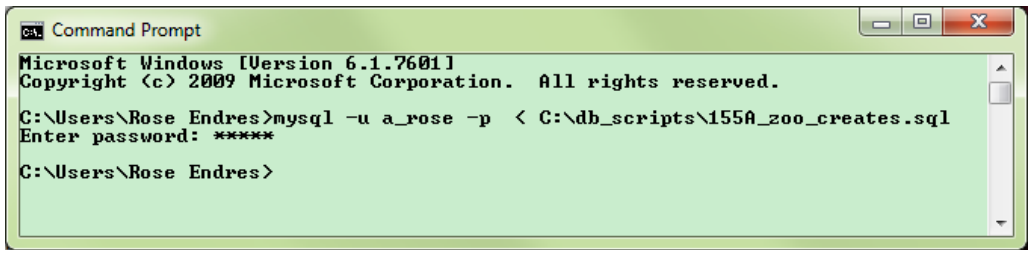
1. Download the zip file with the scripts from the download page to your local computer; the files for the table scripts are at the top of the download page. Unzip the file. You will have two files: 155A_zoo_creates.sql and 155A_zoo_inserts.sql Store these files in a directory on your local computer. I am going to use the directory db_scripts on my C: drive. The complete pathname to the files will be:

   C:\db_scripts\155A_zoo_creates.sql
   C:\db_scripts\155A_zoo_inserts.sql

2. Open a command prompt(terminal window). Enter the following command- use your user name and the path to your copy of the file.

```
mysql -u a_rose -p  < C:\db_scripts\155A_zoo_creates.sql
```

Note that this command is given from a command prompt window- NOT from within the mysql client. I am prompted for my password and the script is run. I am assuming that you have set the path variable.



Running this script created the zoo_2015 table on my account in the a_testbed database.

Now run the second script to populate the table- to insert data rows.

```
mysql -u a_rose -p  < C:\db_scripts\155A_zoo_inserts.sql
```

If you use an incorrect pathname to the script file, you will get an error message. (The system cannot find the path specified.)

Options used in the command:

```
-u is for the user name
-p is to prompt for the password
< is a redirection symbol that indicates the path to the script file.
```

The first time you do anything it is annoying; but this is a much simpler way to run the SQL than typing in the commands or copying them from another file.

2. Creating the zoo_2015 table

We need to decide where to store this table. Since is it not part of any of the major databases we are using, we should store this in the a_testbed database.

The script included the use command to switch to the a_testbed database. If you are already in the a_testbed database, it does not hurt to give this command again.

Now we need to make the table. With a database system, you need to create the place to store the data first; you need to create a database to store tables and you need to create tables to store data.

We need a name for our table; since it stores animals for my zoo, I am going to name the table zoo_2015.

You need to provide names for the columns and specify the data types. In SQL this is done with a Create Table statement. Pay attention to the punctuation marks- but you do not have to add spaces to align the words. The statement ends with a semicolon.

REMEMBER- you should have already run the provided script to create this table. If you run the script again it will drop and recreate the table..

```
Create table a_testbed.zoo_2015 (
  z_id       integer       not null
, z_name     varchar(25)   null
, z_type     varchar(25)   not null
, z_cost     decimal(7,2)  unsigned   not null
, z_dob      datetime      not null
, z_acquired date          not null
)
engine = innoDB;
```

2.1.1. Syntax for Create Table

Don't worry about this too much yet- but this is what that statement says.

"Create table" is a pair of SQL keywords that say that I want to define a table. Follow this with the name of the table. Here I am using the schema name and the table name. If I am "in" the a_testbed database then I only need the tablename.

Inside parentheses, we have a series of column definitions which supply a column name and a data type and a nullability clause.

The data type **varchar** says that the column will store text; varchar(25) means that the text cannot be longer than 25 characters. The data type **integer** says that the column will store integer numbers; these could be positive or negative. The data type **decimal**(7,2) unsigned says that the column will store decimal values that can contain up to 7 digits, 2 of these are after the decimal point The word "unsigned" means that these numbers cannot be negative. The data type **datetime** includes both a date component and a time component. The data type **date** includes just a date component.

If the column is marked as not null, then each row in the table has to have a value. The z_name column could be left empty.

If you have previous experience with SQL tables you might notice that the table does not have a primary key. We will talk more about this in the next unit, but in the assignment people supply sample inserts and if the table had a primary key, then we would need a way to make sure that each inserted row had a different value for the primary key column.

Experiment: When you do the inserts, try inserting data that breaks the rules in the Create table statement. What happens?

2.1.2. Show tables

You can check that your table exists with the `show tables` command. The show tables command will let you see the names of your tables. If your database does not have any tables yet, you get the response: Empty set. If you have tables, this will show the table names. The following display shows that I have one table in my a_testbed database.

```
mysql> show tables;
+---------------------+
| Tables_in_a_testbed |
+---------------------+
| zoo_2015            |
+---------------------+
1 row in set (0.03 sec)
```

2.1.1. What does the table structure look like?

The desc command will show you the column names and data types for a table.

```
mysql> desc zoo_2015;
+------------+-----------------------+------+-----+---------+-------+
| Field      | Type                  | Null | Key | Default | Extra |
+------------+-----------------------+------+-----+---------+-------+
| z_id       | int(11)               | NO   |     | NULL    |       |
| z_name     | varchar(25)           | YES  |     | NULL    |       |
| z_type     | varchar(25)           | NO   |     | NULL    |       |
| z_cost     | decimal(7,2) unsigned | NO   |     | NULL    |       |
| z_dob      | datetime              | NO   |     | NULL    |       |
| z_acquired | date                  | NO   |     | NULL    |       |
+------------+-----------------------+------+-----+---------+-------+
6 rows in set (0.05 sec)
```

You can also use the show columns command for the same display.

```
show columns from zoo_2015;
```

2.1.2. What is the sql that creates a table?

The command to display the sql for your table is Show Create table

Notice that this is not exactly the same as the sql I will give you for the zoo_2015 table. When this is displayed, the default settings are included.

```
mysql> show create table zoo_2015\G
*************************** 1. row ***************************
       Table: zoo_2015
Create Table: CREATE TABLE `zoo_2015` (
  `z_id` int(11) NOT NULL,
  `z_name` varchar(25) DEFAULT NULL,
  `z_type` varchar(25) NOT NULL,
  `z_cost` decimal(7,2) unsigned NOT NULL,
  `z_dob` datetime NOT NULL,
  `z_acquired` date NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8
1 row in set (0.00 sec)
```

You may notice that MySQL displays the table and column names enclosed in back ticks (`); don't worry about that yet.

One thing you should notice is the line `ENGINE=InnoDB`. This means that this table was set up in a particular way. If you did a default installation of MySQL 5.5 or 5.6, your tables should default to InnoDB tables; all tables for this class should be InnoDB tables.

3. Insert statement

The second script inserted rows into the table. This is done with an insert statement. The column names in the first clause align with the data values. You do not have to do that alignment when you enter the statements. But you do need the parentheses and the commas between items and you need the single quotes around the character values, including the values for the dates.

MySQL lets you use single or double quotes ; since other dbms allow only single quotes for delimiters, and that is the ANSI standard rule for delimiters, you might want to get into the habit of using single quotes.

This is a copy of the first insert in the script. Do not run it again here.

```
Insert Into a_testbed.zoo_2015  (z_id, z_name, z_type, z_cost, z_dob,
z_acquired)   Values
   (23,'Sam','Giraffe', 5000.00, '2012-05-15','2012-05-15');
```

4. Select statement

Now we can run the Select statement. We now see the data and the system messages that we have 10 rows in the return set.

Demo 01: select query to see the data in the table

```
mysql> select z_id, z_name, z_type, z_cost, z_dob, z_acquired
    -> from zoo_2015;
+------+---------+-----------+---------+---------------------+------------+
| z_id | z_name  | z_type    | z_cost  | z_dob               | z_acquired |
+------+---------+-----------+---------+---------------------+------------+
|   23 | Sam     | Giraffe   | 5000.00 | 2012-05-15 00:00:00 | 2012-05-15 |
|   25 | Abigail | Armadillo |  490.00 | 2014-12-15 00:00:00 | 2015-01-05 |
|   56 | Leon    | Lion      | 5000.00 | 2010-02-25 00:00:00 | 2010-03-25 |
|   57 | Lenora  | Lion      | 5000.00 | 2014-03-25 00:00:00 | 2014-03-31 |
|   85 | Sally   | Giraffe   | 5000.25 | 2011-05-15 00:00:00 | 2012-05-15 |
|   43 | Huey    | Zebra     | 2500.25 | 2013-06-02 00:00:00 | 2014-06-02 |
```

```
|   44 | Dewey   | Zebra     | 2500.25 | 2013-06-02 00:00:00 | 2014-06-02 |
|   45 | Louie   | Zebra     | 2500.25 | 2013-01-02 00:00:00 | 2014-01-02 |
|   47 | NULL    | Horse     |  490.00 | 2010-05-15 00:00:00 | 2010-04-15 |
|   52 | Dewey   | Giraffe   | 3750.00 | 2013-06-06 00:00:00 | 2013-07-12 |
+------+---------+-----------+---------+---------------------+------------+
10 rows in set (0.00 sec)
```

5. Filter

We might want to see only some of the animals.

Demo 02: We want to see the armadillo

```
Select z_id, z_name, z_type, z_cost
From zoo_2015
Where z_type = 'Armadillo';
+------+---------+-----------+--------+
| z_id | z_name  | z_type    | z_cost |
+------+---------+-----------+--------+
|   25 | Abigail | Armadillo | 490.00 |
+------+---------+-----------+--------+
```

Demo 03: We want to see animals that cost 5000

```
Select z_id, z_name, z_type, z_cost
From zoo_2015
Where z_cost = 5000;
+------+--------+---------+---------+
| z_id | z_name | z_type  | z_cost  |
+------+--------+---------+---------+
|   23 | Sam    | Giraffe | 5000.00 |
|   56 | Leon   | Lion    | 5000.00 |
|   57 | Lenora | Lion    | 5000.00 |
+------+--------+---------+---------+
3 rows in set (0.00 sec)
```

Demo 04: We want to see the unicorns- but I do not have any

```
Select z_id, z_name, z_type, z_cost
From zoo_2015
Where z_type = 'Unicorn';
Empty set (0.00 sec)
```

This has been a quick tour through the main types of SQL statements: a statement to create a structure to hold data; a statement to modify (in this case add) the data in the table; and statements to display the data in the table.

6. The Insert statement

The MySQL Insert statement has a few variations. This is the insert statement show above. This is the full version of the Insert. The table zoo_2015 has six columns. The column names are listed in the first parenthesized list- the column list; and the values are listed in the same order in the values list. It is possible to change the order of the columns in the column list - although that is not generally useful. (You should have already run the insert script do not do these inserts.)

Demo 05: More inserts you should run

```
Insert Into zoo_2015  (
  z_id
, z_name
, z_type
, z_cost
, z_dob
, z_acquired
) values
(
  257
, 'Arnold'
, 'Giraffe'
, 5000.00
, '2014-05-15'
, '2014-05-15'
);
```

If you did not want to supply a value for the animal name (it is not a required column), you could skip that column in the column list and skip an entry in the values list.

```
Insert Into zoo_2015  (
  z_id
, z_type
, z_cost
, z_dob
, z_acquired
) values
(
  258
, 'Giraffe'
, 5000.00
, '2013-05-15'
, '2013-05-15'
);
```

If you supply a value for every column in the table and you supply the values in the same order as they are listed in the Create table statement, then you are allowed to skip the column list. Here I am using the value null for the animal name. I have also added a time component for the z_dob attribute

```
Insert Into zoo_2015 values
(
  259
, null
, 'Giraffe'
, 5000.00
, '2002-05-15 10:45:00'
, '2002-05-15'
);
```

MySQL also supports a multi-row insert. Suppose I want to insert two rows. I can use the following.

```
Insert Into zoo_2015 values
(  260, 'Artemis',  'Giraffe', 1500.00, '2013-06-06 10:45:00','2013-08-15' )
,
(  261, 'Diana', 'Giraffe', 120.95, '2000-06-06 10:47:00','2015-01-15' )
;
```

If there is anything wrong with either of the rows to be inserted, then the entire statement fails and neither row is inserted.

Now we have a tower of giraffe.

```
+------+---------+---------+---------+---------------------+------------+
| z_id | z_name  | z_type  | z_cost  | z_dob               | z_acquired |
+------+---------+---------+---------+---------------------+------------+
|   23 | Sam     | Giraffe | 5000.00 | 2012-05-15 00:00:00 | 2012-05-15 |
|   85 | Sally   | Giraffe | 5000.25 | 2011-05-15 00:00:00 | 2012-05-15 |
|   52 | Dewey   | Giraffe | 3750.00 | 2013-06-06 00:00:00 | 2013-07-12 |
|  257 | Arnold  | Giraffe | 5000.00 | 2014-05-15 00:00:00 | 2014-05-15 |
|  258 | NULL    | Giraffe | 5000.00 | 2013-05-15 00:00:00 | 2013-05-15 |
|  259 | NULL    | Giraffe | 5000.00 | 2002-05-15 10:45:00 | 2002-05-15 |
|  260 | Artemis | Giraffe | 1500.00 | 2013-06-06 10:45:00 | 2013-08-15 |
|  261 | Diana   | Giraffe |  120.95 | 2000-06-06 10:47:00 | 2015-01-15 |
+------+---------+---------+---------+---------------------+------------+
8 rows in set (0.00 sec)
```