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. 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 the zoo table

1.1. Setup

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.

First, switch to the a_testbed database that you should have created in doc 01_03. The MySQL client command to do this is `use` followed by the database name. The following shows that command and the system response. If you are already in the a_testbed database, it does not hurt to give this command again.

```
mysql> use a_testbed;
Database changed
```

1.2. Create table

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.

You need to provide names for the columns and specify the data types. In SQL this is done with a Create Table statement. At a mysql> prompt, type the following 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. (The code for the demos given here are in a file posted in the demo section of the download page.)

Demo 01: Define and create the table.

```
Use a_testbed;

Drop table If Exists zoo;
```

```
   Create table zoo (
     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;
```

As you enter the query, for each line after the first, you will get a prompt that is generated by the mysql client. The last line ends with a semicolon and that signals that the statement is complete and it should be executed.

The standard response for a successful query shows the number of rows affected (this is 0 rows since we just made a table) and the time it took to do this.

If you get a different result, you have probably made a typing error and will get an error message. Often it is the very unhelpful message: ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near . . . at line 999.

1.2.1. 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.

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?

Also you cannot have two tables with the same name in the same database- so don't try to run this again. If necessary, you can issue the command

```
   drop table if exists zoo;
```

and start over.

1.2.2. 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.

Demo 02: Show and describe

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

1.2.1. 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;
+------------+-----------------------+------+-----+---------+-------+
| 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;
```

1.2.2. 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 table. When this is displayed, the default settings are included.

```
mysql> show create table zoo\G
*************************** 1. row ***************************
       Table: zoo
Create Table: CREATE TABLE `zoo` (
  `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=latin1
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.

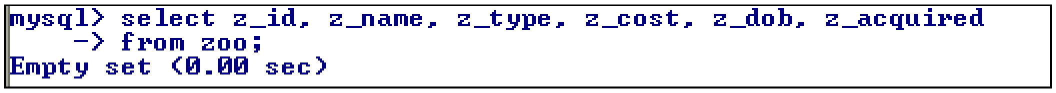
One thing you should notice is the line `ENGINE=InnoDB`. This means that this table was set up in a particular way. If you did a default installation of MySQL 5.5 or 5.6, your tables should default to InnoDB tables; all tables for this class should be InnoDB tables.

2. Select statement

Now that we have a table, we could ask what is in our table.

Demo 03: That is done with a select statement.

```
Select z_id, z_name, z_type, z_cost, z_dob, z_acquired
From zoo;
```



```
mysql> select z_id, z_name, z_type, z_cost, z_dob, z_acquired
    -> from zoo;
Empty set (0.00 sec)
```

Here we have a picture here of how this looks when we enter it. The response that we get is that there are no rows selected- which makes sense because we did not put any rows into the table. But it is OK to ask to see what is in an empty table.

3. Insert statement

So let's put in some rows. 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.

Demo 04:

```
Insert Into zoo  (z_id, z_name, z_type, z_cost, z_dob, z_acquired) values
 (23,'Sam','Giraffe', 5000.00, '2002-05-15 10:45:00', '2002-05-15');

Insert Into zoo  (z_id, z_name, z_type, z_cost, z_dob, z_acquired) values
 (25,'Abigail','Armadillo', 490.00, '2010-05-15 08:30:00 ', '2010-04-15');
```

The response with each of these inserts is "1 row affected" We do not see a display of the data since we did not ask to see the data.

Now we can run that Select statement again. We now see the data and the system messages that we have 2 rows in the return set.

```
mysql> select z_id, z_name, z_type, z_cost, z_dob, z_acquired
    -> from zoo;
+------+---------+-----------+---------+---------------------+------------+
| z_id | z_name  | z_type    | z_cost  | z_dob               | z_acquired |
+------+---------+-----------+---------+---------------------+------------+
|   23 | Sam     | Giraffe   | 5000.00 | 2002-05-15 10:45:00 | 2002-05-15 |
|   25 | Abigail | Armadillo |  490.00 | 2010-05-15 08:30:00 | 2010-04-15 |
+------+---------+-----------+---------+---------------------+------------+
2 rows in set (0.00 sec)
```

There are more inserts posted in the demo for this document. You should insert these rows to continue.

4. Filter

We might want to see only some of the animals.

Demo 05: We want to see the armadillo

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

Demo 06: We want to see animals that cost 5000

```
Select z_id, z_name, z_type, z_cost
From zoo
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 07: We want to see the unicorns- but I do not have any

```
Select z_id, z_name, z_type, z_cost
From zoo
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.

5. 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 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.

```
Insert Into zoo  (
  z_id
, z_name
, z_type
, z_cost
, z_dob
, z_acquired
) values
(
  23
, 'Sam'
, 'Giraffe'
, 5000.00
, '2002-05-15 10:45:00'
, '2002-05-15'
);
```

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  (
  z_id
, z_type
, z_cost
, z_dob
, z_acquired
) values
(
  23
, 'Giraffe'
, 5000.00
, '2002-05-15 10:45:00'
, '2002-05-15'
);
```

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.

```
Insert Into values
(
  23
, 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 values
(  823, 'Sam',  'Lion', 5000.00, '2013-06-06 10:45:00','2013-08-15' )
,
(  824, 'Dave', 'Lion', 5000.00, '2013-06-06 10:47:00','2013-08-15' )
;
```

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