Table of Contents

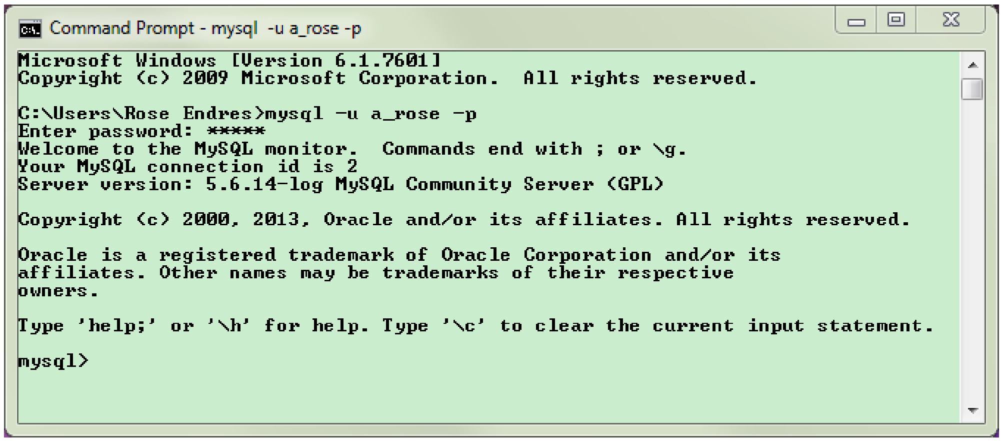
1. Using the mysql command line client

This is a command line client. On a windows machine, where MySQL is properly installed, you can start the client from a C:> prompt using the Command Prompt program. (on a windows machine this is probably in All Programs→ Accessories. You can then pin this to your start menu to make it easier to find. If you right click the title bar for this window you can find options for changing the color scheme etc.) On a Mac, you use the Terminal program to start up the mysql client.

Start up the client with the command mysql and the command option –u followed by your user name and -p to be prompted for the password for your account. My account user name is a_rose.

```
   mysql –u a_rose –p
```

The screen shot below is taken on a windows system, but you would get the same type of login with a Mac.



```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\Rose Endres>mysql -u a_rose -p
Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 5.6.14-log MySQL Community Server (GPL)

Copyright (c) 2000, 2013, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

You can give some commands without being in any database.

The user a_rose can see several databases. In addition to the output of the command itself- which is a table display, you also get the row count and the time it took to run the query.

```
   mysql> show databases;
   +--------------------+
   | Database           |
   +--------------------+
   | information_schema |
   | a_bkinfo           |
   | a_bkorders         |
   | a_emp              |
   | a_oe               |
   | a_prd              |
   | a_testbed          |
   | a_vets             |
   | a_xml              |
   +--------------------+
   9 rows in set (0.00 sec)
```

If I log in as another user and give the show databases command, that user can see only the databases he was given privileges to and information_schema.

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| books              |
| vets               |
+--------------------+
3 rows in set (0.00 sec)
```

The rest of this document uses my regular ' a_rose' account.

```
mysql> select version(), current_date;
+------------+--------------+
| version()  | current_date |
+------------+--------------+
| 5.6.14-log | 2014-01-03   |
+------------+--------------+
1 row in set (0.00 sec)
```

In addition to the output of the command itself- which is a table display, you also get the row count and the time it took to run the query.

1.1. Command terminators

So far we have been using the semicolon to terminate a command and tell the client to run that command. You can also use \G to end the command and get the data displayed vertically. This is useful for displays which have very few rows but the column values might be long. It is not particular helpful for most table displays. (This is an upper case G.)

```
mysql> select version(), current_date\G
*************************** 1. row ***************************
   version(): 5.6.14-log
current_date: 2014-01-03
1 row in set (0.00 sec)
```

You can also use a lower case \g as a command terminator but the display is different. The template requires the upper case G

```
mysql> select version(), current_date\g
+------------+--------------+
| version()  | current_date |
+------------+--------------+
| 5.6.14-log | 2014-01-03   |
+------------+--------------+
1 row in set (0.00 sec)
```

1.2. Entering queries

You can select a literal or expressions.

```
   mysql> select 'Hello World', 60*60*24;
   +-------------+----------+
   | Hello World | 60*60*24 |
   +-------------+----------+
   | Hello World |    86400 |
   +-------------+----------+
   1 row in set (0.00 sec)
```

You can put more than one query on a line and they execute one after the other. This also shows column aliases;

```
   mysql> select 'Hello' as greeting; select now() as "clock time";
   +----------+
   | greeting |
   +----------+
   | Hello    |
   +----------+
   1 row in set (0.00 sec)
   +---------------------+
   | clock time          |
   +---------------------+
   | 2014-01-03 19:57:46 |
   +---------------------+
   1 row in set (0.01 sec)
```

You can enter a single command on multiple physical lines. You get more secondary prompts until you give the semicolon delimiter followed by the Enter key.

```
   mysql> select 'Hello' as Greeting1,
       -> 'Good morning' as Greeting2
       -> ,
       -> now()
       -> ;
   +-----------+--------------+---------------------+
   | Greeting1 | Greeting2    | now()               |
   +-----------+--------------+---------------------+
   | Hello     | Good morning | 2014-01-03 19:58:15 |
   +-----------+--------------+---------------------+
   1 row in set (0.00 sec)
```

1.3. Quitting a command

Sometimes you are entering an SQL command and decide that you do not want to run it. Perhaps you realize that the syntax is wrong or that you selected the wrong data to work with. You can use \c to quit a command if you decide not to run it. If you have an open quote delimiter, you will need to close the delimiter before using the \c command.

For example: Suppose you are trying to run one of the previous demos and accidentally enter the wrong quotes. Here I am opening with a single quote but trying to end the literal with a double quote.

```
   mysql> select 'Hello World", 60 * 60 *24;
       '>
```

Note that the prompt is now a quote followed by >. This is the client's way of trying to tell you that you have not closed the single quote delimiter. If you try to quit this by entering \c, you get another prompt.

```
   mysql> select 'Hello World", 60 * 60 *24;
       '> \c
       '>
```

What you need to do in this case is close the literal with a single quotes and then use the \c command.

```
mysql> select 'Hello World", 60 * 60 *24;
    '> \c
    '> '\c
mysql>
```

1.4. Secondary prompts

The command line interface has a series of secondary prompts with the following meanings.

mysql>	This is the prompt to enter a new command
->	This is the prompt for the next line in a multi-line command

The following will probably happen to you. You type in the leading ' for a string and do not type the ending quote. Note that the prompt changed somewhat in an attempt to let you know that it is waiting for the ending quote. The carriage return you entered is in the string literal. You can enter a closing quote and the terminator (and get error messages on the sql) or terminate the command.

If you open a delimiter and do not close it you will get a following prompt to indicate the delimiter needed to close the literal.

`'>`	waiting for the completion of a string with a single quote
`">`	waiting for the completion of a string with a double quote
`'>`	waiting for the completion of a string with a back tick quote
`/*>`	waiting for the completion of an extended comment

1.5. Warnings

Sometimes you can enter a query that runs but MySQL has a warning about the query. The following query tries to multiply the number 5 by a string value. MySQL will run this but if you look at the output, the product is 0 and the feedback line says there is a warning.

```
mysql> Select 5 * 'cat';
+------------+
| 5 * 'cat' |
+------------+
|          0 |
+------------+
1 row in set, 1 warning (0.00 sec)
```

You can give the command `show warnings` to see the warning messages on the most recent command.

```
mysql> show warnings;
+---------+------+------------------------------------------+
| Level   | Code | Message                                  |
+---------+------+------------------------------------------+
| Warning | 1292 | Truncated incorrect DOUBLE value: 'cat' |
+---------+------+------------------------------------------+
```

You could also use the \W command to signal that you want all of the warning messages displayed without having to ask for them. To turn this option off, use \w with a lower case w. The assignment scripts are run with the Warning option turned on.

```
mysql> \W
Show warnings enabled.
mysql> Select 5 * 'cat';
+-----------+
| 5 * 'cat' |
+-----------+
|         0 |
+-----------+
1 row in set, 1 warning (0.00 sec)
Warning (Code 1292): Truncated incorrect DOUBLE value: 'cat'
```

You should correct the query when there is a warning such as this.

2. Some useful commands to get started.

2.1. What databases do we have access to

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| a_bkinfo           |
| a_bkorders         |
| a_emp              |
| a_oe               |
| a_prd              |
| a_testbed          |
| a_vets             |
| a_xml              |
+--------------------+
```

If you see the database mysql, don't work with it directly. You may have a test database that was installed when you installed mysql. The information_schema database is a system db. The other databases are user created.databases.

2.2. What is the current database?

database() is a function and we are asking to see the return value of that function. If you have just logged in and have not selected any database, then the return value is NULL. Be careful that you do *not* have a space between the function name and the opening parentheses.

```
mysql> select database();
+------------+
| database() |
+------------+
| NULL       |
+------------+
```

2.3. Select a database to use

Generally you want to do your work in the context of a specific database. You can set the database with the Use command.

```
mysql> use a_testbed;
Database changed
mysql> select database();
+--------------+
| database()   |
+--------------+
| a_testbed    |
+--------------+
```