MySQL Database System

MySQL is a simple but powerful, open source database management system, which is used for storing and managing data in systems.The point of this document is to be an introduction to basic functionalities of MySQL and the syntaxes required for moving between databases as well as creating and populating tables with data. This tutorial will be taking place on MobaXTerm, meaning a level of basic proficiency is required to complete the assigned task.

Once logged into PRClab1 on MobaXTerm, connect to the MySQL server by typing in "mysql -u user_name -p". If it is your first time, you will be prompted to set your password with the command "SET PASSWORD = PASSWORD ("new_password")".

```
[capriotn@prclab1:~]$ mysql -u capriotn -p
Enter password:
ERROR 1045 (28000): Access denied for user 'capriotn'@'localhost' (using password: YES)
[capriotn@prclab1:~]$ mysql -u capriotn -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2501
Server version: 5.6.42 MySQL Community Server (GPL)

Copyright (c) 2000, 2018, 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> SET PASSWORD = PASSWORD('Software1234');
Query OK, 0 rows affected (0.00 sec)

mysql>
```

Once you change your password, you want to start with the command "SHOW GRANTS;", this will show you what access you have been granted as well as the permissions you have for your current database, this is important to know before attempting to write or tead information to and from the database tables. Following with the "SHOW DATABASES;" command, we will be able to see all the accessible databases on our network. While we are only able to write in a few databases, there is a clear lapse of security given that we have the ability to see all database names on the network, regardless of my permissions or grants

```
mysql> SHOW GRANTS;
+-----------------------------------------------------------------------------+
| Grants for capriotn@%                                                       |
+-----------------------------------------------------------------------------+
| GRANT SELECT ON *.* TO 'capriotn'@'%' IDENTIFIED BY PASSWORD <secret>       |
| GRANT ALL PRIVILEGES ON `capriotn_db`.* TO 'capriotn'@'%'                   |
+-----------------------------------------------------------------------------+
```

```
mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| CS317DB            |
| Courses_db         |
| DroneGroupProject  |
| FRS_Database       |
| G2_DV_Game_Account |
| G4_Song_DB         |
| G5_Dilbert_Mgr     |
| G6_Drone_Net       |
| G7_ERAU_Housing    |
| G&_ERAU_Housing    |
| Group4_Criminal_DB |
| Recipies           |
| WHY                |
```

From there we can explore the sakila database using the command “use sakila;” , then we will use the command “show tables;” to return all the tables contained in the database we are using. By counting the tables made available to us, we cna determine that the sakila database contains 26 tables of info,

```
mysql> USE sakila;
Reading table information for
You can turn off this feature

Database changed
mysql> SHOW Tables;
+----------------------------+
| Tables_in_sakila           |
+----------------------------+
| actor                      |
| actor_info                 |
| address                    |
| category                   |
| city                       |
| class_schedule             |
| classes                    |
```

In order to see the number of films available, we can use the command “SELECT * FROM film;”, this will show us the complete table of films and we can count the number of films using the film_id, this tells us that there are 1000 films, with the last one bing ZORRO ARK. Another method would be to use the command “COUNT * FROM film;” which would return the number of films in the table, 1000.

```
mysql> SELECT * FROM film;
|    1000 | ZORRO ARK                | A Intrepid Panorama of a Mad Scientist And a Boy who must Redeem a Boy in A Monaster
y                                    |         2006 |           1 |                NULL |              3 |
4.99 |     50 |           18.99 | NC-17  | Trailers,Commentaries,Behind the Scenes                | 2006-02-15 05:03:42 |
```

If you want to see how a table is made up, rather than the actual contents of the table, you can use the command “DESCRIBE ‘name’;” It displays column names and the data type of each column with the associated sizes. This is more about the table, and less about what's in the table

```
mysql> DESCRIBE film;
+----------------------------+
| Field                | Type                                                                                | Null | Key | Default
| Extra                |
+----------------------+-------------------------------------------------------------------------------------+------+-----+-----------------
+----------------------------+
| film_id              | smallint(5) unsigned                                                                | NO   | PRI | NULL
| auto_increment       |
| title                | varchar(255)                                                                        | NO   | MUL | NULL
|
| description          | text                                                                                | YES  |     | NULL
|
| release_year         | year(4)                                                                             | YES  |     | NULL
|
| language_id          | tinyint(3) unsigned                                                                 | NO   | MUL | NULL
|
| original_language_id | tinyint(3) unsigned                                                                 | YES  | MUL | NULL
|
| rental_duration      | tinyint(3) unsigned                                                                 | NO   |     | 3
|
| rental_rate          | decimal(4,2)                                                                        | NO   |     | 4.99
|
| length               | smallint(5) unsigned                                                                | YES  |     | NULL
|
| replacement_cost     | decimal(5,2)                                                                        | NO   |     | 19.99
|
| rating               | enum('G','PG','PG-13','R','NC-17')                                                  | YES  |     | G
|
| special_features     | set('Trailers','Commentaries','Deleted Scenes','Behind the Scenes')                 | YES  |     | NULL
|
| last_update          | timestamp                                                                           | NO   |     | CURRENT_TIMESTAMP
on update CURRENT_TIMESTAMP |
```

After looking into the film table we will be looking at the actor table. To display the actor table we will use the command “SELECT * FROM actor;”. The table columns show the actor id, which assigns an identification number to each actor. We also have columns for first and last names, which tell actors apart if IDs arent used. The last column is called ‘last update’, which shows the last time the actors specific row was modified.

```
mysql> SELECT  * FROM actor;
+----------+------------+--------------+---------------------+
| actor_id | first_name | last_name    | last_update         |
+----------+------------+--------------+---------------------+
|        1 | PENELOPE   | GUINESS      | 2006-02-15 04:34:33 |
```

We can use the “SELECT” command to modify the list in the table, we can do various features for sorting data, such as ordering the table in alphabetical order by last or first name, to do this use the command “SELECT actor_id, first_name, last_name FROM actor ORDER BY last_name ASC;”, and if we want to sort instead by first name, we change the statement after “ORDER BY” to first_name, and to go descending alphabetical order, we can simply change the “ASC” to a “DEC” to descend down the list.

```
mysql> SELECT actor_id, first_name, last_name FROM actor ORDER BY last_name ASC;
+----------+------------+--------------+
| actor_id | first_name | last_name    |
+----------+------------+--------------+
|       58 | CHRISTIAN  | AKROYD       |
|       92 | KIRSTEN    | AKROYD       |
|      182 | DEBBIE     | AKROYD       |
|      118 | CUBA       | ALLEN        |
|      145 | KIM        | ALLEN        |
|      194 | MERYL      | ALLEN        |
|       76 | ANGELINA   | ASTAIRE      |
|      112 | RUSSELL    | BACALL       |
|       67 | JESSICA    | BAILEY       |
|      190 | AUDREY     | BAILEY       |
|      115 | HARRISON   | BALE         |
|      187 | RENEE      | BALL         |
|       47 | JULIA      | BARRYMORE    |
|      158 | VIVIEN     | BASINGER     |
|      124 | SCARLETT   | BENING       |
mysql> SELECT actor_id, first_name, last_name FROM actor ORDER BY first_name ASC;
+----------+------------+--------------+
| actor_id | first_name | last_name    |
+----------+------------+--------------+
|       71 | ADAM       | GRANT        |
|      132 | ADAM       | HOPPER       |
|      165 | AL         | GARLAND      |
|      173 | ALAN       | DREYFUSS     |
|      125 | ALBERT     | NOLTE        |
|      146 | ALBERT     | JOHANSSON    |
```

Next you will create a select query that shows how many actors there are in the actor table, but rather than counting the actor ID, we can use the “COUNT COMMAND”, using the command “SELECT COUNT(actor_id) FROM actor;”, This gives us the number of actors, 200. Then, we can modify the search by using the “WHERE” function, which essentially is used to set

parameters, in this case we will add “WHERE last_name = “Wood”” in order to return a count of the actors, but this time, only counting the actors with the “last_name” Wood.

```
mysql> SELECT COUNT(actor_id) FROM actor;
+-----------------+
| COUNT(actor_id) |
+-----------------+
|             200 |
+-----------------+
1 row in set (0.00 sec)

mysql> SELECT COUNT(last_name) FROM actor WHERE last_name = "Wood";
+------------------+
| COUNT(last_name) |
+------------------+
|                2 |
+------------------+
```

For this exercise, we will be returning to our personal databases using the same command we used earlier in the exercise, type “USE capriotn_db” to access this database. Once in your own database, we will begin by creating a table, the command for creating a table in a database is “CREATE TABLE myhomes(address int(5), street varchar(20), city varchar(20), zipcode int(5)”. You can keep adding columns to the table for other information you want added, such as bedrooms, bathrooms, square feet, and other important data. Tables give you the opportunity to assign characteristics to objects for identification.

```
mysql> CREATE TABLE house_info (address int(5), street varchar(20), city varchar(20), zipcode int(5));
Query OK, 0 rows affected (0.00 sec)
```

We can now use the “SHOW TABLES" command to check that we successfully created the table, once weve done so we can use the command “INSERT INTO house_info (address,street,city,zipcode) VALUES (‘11111’,’City_Name’,’Street_Name’,’zip_code’);”. If you are successful (check your info made it to the table with the “SELECT *FROM house_info” command), repeat as many times as necessary to completely populate your table.

```
+----------------------+
| Tables_in_capriotn_db |
+----------------------+
| house_info           |
+----------------------+

mysql> INSERT INTO house_info (address,street,city,zipcode) VALUES ('77436', 'Clev
eland_sucks', 'Cleveland', '44572');

mysql> SELECT * FROM house_info;
+---------+-----------------+-----------------+---------+
| address | street          | city            | zipcode |
+---------+-----------------+-----------------+---------+
|   11111 | coconino        | gilbert         |   85298 |
|   63647 | robert          | prescott        |   82398 |
|   74828 | Queens          | prescott_valley |   74646 |
|   69420 | Cleveland       | detroit         |   89482 |
|   44232 | higley          | detroit         |   88822 |
|    3159 | coconino        | queen_creek     |   85298 |
|   77436 | Cleveland_sucks | Cleveland       |   44572 |
|   88472 | Car             | Chandler        |   22341 |
+---------+-----------------+-----------------+---------+
```

By using the MySQL tool, you can easily and efficiently navegate database systems as well as quickly create tables for data storage. While it uses a simple syntax, there is a learning curve to be met, however, with the right practice, it will be extremely easy to become familiarized with the navigational commands. MySQL has some security flaws that can make it possible for database systems to be breached, however, it is still a very powerful tool for understanding database fundamentals.