

Activity No. 1

SQL Data Definition Language Commands

Course Code: CPE011

Program: BSCPE21-S3

Course Title: Database Management System

Date Performed: 8/17/2022

Section: S3

Date Submitted: 8/20/2022

Name: Aaron Martin Parallag Caro
Christian Efa

Instructor: Dr. Jonathan Taylar

1. Objective(s):

This activity aims to introduce the Structured Query Language (SQL) using the Data Definition Language (DDL) commands in a MySQL Database

2. Intended Learning Outcomes (ILOs):

The students should be able to:

2.1 Access a MySQL Database using Command Line Interface.

2.2 Create, Alter, and Drop a database schema using the command prompt, MySQL, and XAMPP

2.2 Create, Alter, and Drop a table and its fields in a database schema

2.3 Show the different databases, tables and their structures present in the MySQL Server

3. Discussion:

MySQL is a database management system that allows you to manage relational databases. It is open source software backed by Oracle. It means you can use MySQL for free. In addition, if you want, you can change its source code to suit your needs. Even though MySQL is open source software, you can buy a commercial license version from Oracle to get a premium support service.

MySQL is pretty easy to master in comparison with another database software like Oracle Database, or Microsoft SQL Server. MySQL can run on various platforms UNIX, Linux, Windows, etc. You can install it in a server or even in a desktop. In addition, MySQL is reliable, scalable, and fast.

We can interact with an SQL Database such as MySQL using various SQL commands that are grouped into 2 main categories: **Data Definition Language (DDL)**, and the **Data Manipulation Language (DML)**. Other categories of commands are used for more advanced use-cases such as user permissions, roles, and transactions.

The **Data Definition Language (DDL)** allows you to create and define the structure of your database and your tables. Moreover, it allows you to alter the structure of your database and also to delete/drop them if necessary. The SQL DDL Commands that are commonly used in database programming are: **CREATE**, **DROP**, and **ALTER**.

In this activity you will use SQL DDL Commands to create a database and a database table, then alter the structure of a table, and finally perform cleanup by dropping the tables and the database itself. To interact with a database, you will first need to connect to a MySQL Database Server.

A sample of the command to create a database is shown below:

```
MariaDB [(none)]> CREATE DATABASE db1;  
Query OK, 1 row affected (0.07 sec)  
  
MariaDB [(none)]>
```

For creating tables, you will need to be familiar with the various data types needed:

Data Types

A data type or simply type is a classification identifying one of various types of data.

In MySQL, there are three main types of data: Text, Number and Date/Time. This table shows some of the data types available in MySQL although there are many more variations besides the ones in this discussion.

Data Type	Description
CHARACTER(n)	Character string. Fixed-length n
VARCHAR(n) or CHARACTER VARYING(n)	Character string. Variable length. Maximum length n
BINARY(n)	Binary string. Fixed-length n
BOOLEAN	Stores TRUE or FALSE values
VARBINARY(n) or BINARY VARYING(n)	Binary string. Variable length. Maximum length n
INTEGER(p)	Integer numerical (no decimal). Precision p
SMALLINT	Integer numerical (no decimal). Precision 5
INTEGER	Integer numerical (no decimal). Precision 10
BIGINT	Integer numerical (no decimal). Precision 19
DECIMAL(p,s)	Exact numerical, precision p, scale s. Example: decimal(5,2) is a number that has 3 digits before the decimal and 2 digits after the decimal
NUMERIC(p,s)	Exact numerical, precision p, scale s. (Same as DECIMAL)
FLOAT(p)	Approximate numerical, mantissa precision p. A floating number in base 10 exponential notation. The size argument for this type consists of a single number specifying the minimum precision.
REAL	Approximate numerical, mantissa precision 7
FLOAT	Approximate numerical, mantissa precision 16
DOUBLE PRECISION	Approximate numerical, mantissa precision 16
DATE	Stores year, month, and day values
TIME	Stores hour, minute, and second values
TIMESTAMP	Stores year, month, day, hour, minute, and second values
INTERVAL	Composed of a number of integer fields, representing a period of time, depending on the type of interval
ARRAY	A set-length and ordered collection of elements
MULTISET	A variable-length and unordered collection of elements
XML	Stores XML data

You will need to select the right data type for each field in your database table. More information about each data type is available on the MySQL official documentation: <https://dev.mysql.com/doc/refman/5.7/en/data-types.html>

A sample of the command to create a database is shown below:

```

MariaDB [(none)]> use db1;
Database changed
MariaDB [db1]> CREATE TABLE students(first_name CHAR(50), last_name CHAR(50),
->                                age INT, date_lastenrolled DATE, is_enrolled BOOLEAN);
Query OK, 0 rows affected (0.91 sec)

MariaDB [db1]>

```

XAMPP which stands for (**X** – Cross Platform, **A** – Apache, **M** – MariaDB, **P** – PHP, **P** – PERL) is a cross-platform, open-source package of various software which includes a variant of MySQL DB called MariaDB to help you easily get started with databases and using various SQL commands. We will be using XAMPP to access a MySQL database in this activity.

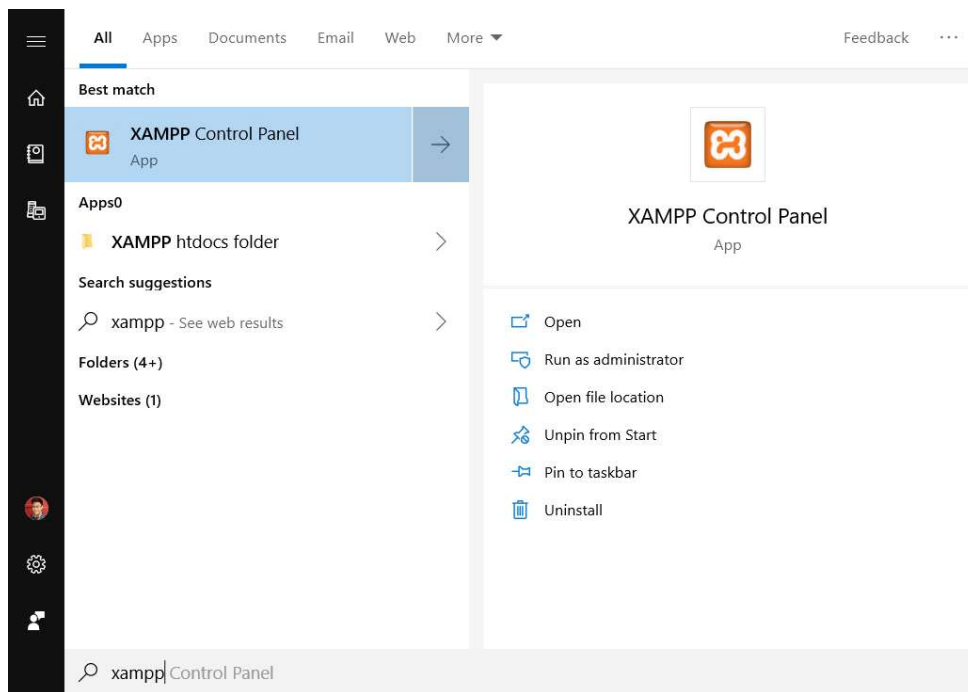
4. Materials and Equipment:

Desktop Computer
 Windows Operating System
 XAMPP Application

5. Procedure:

Starting MySQL using XAMPP

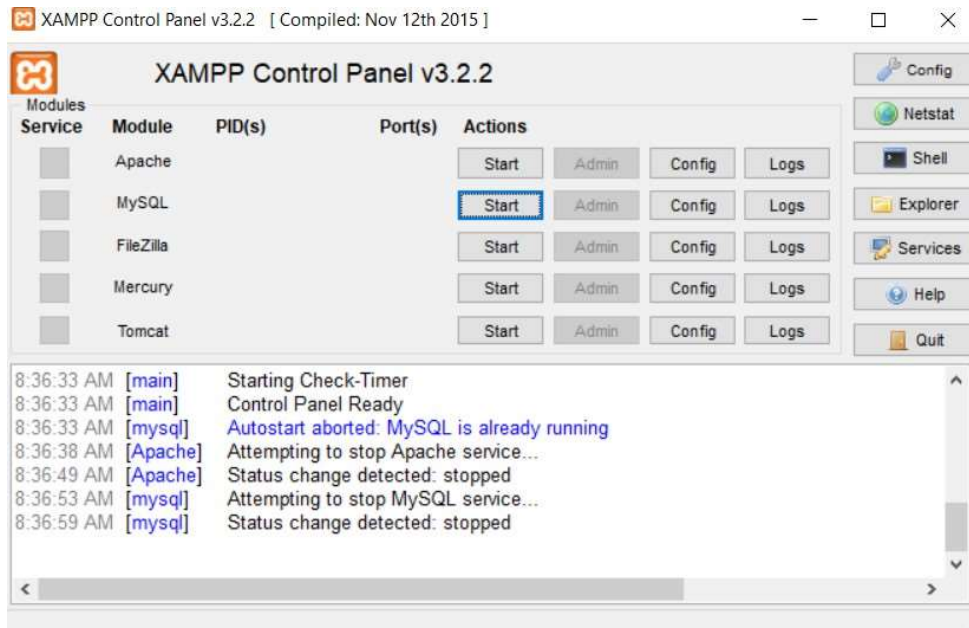
1. Select the XAMPP Control Panel Application from the Desktop or Start Menu



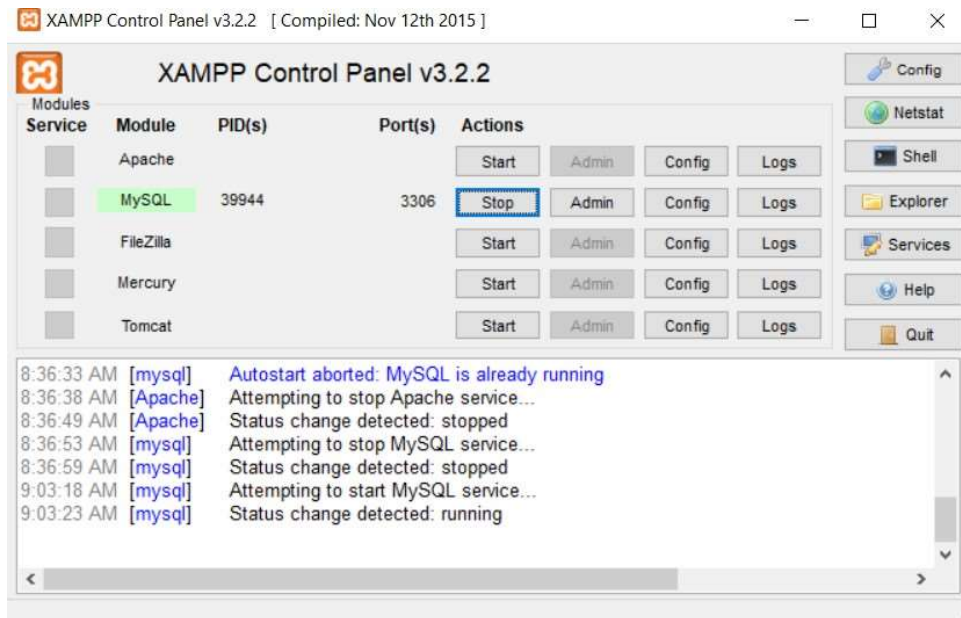
Accessing **XAMPP** is the same for both Windows 10 and lower Windows Operating System versions. It can be searched easily using the Windows Search field.

2. On the dialog box, press *Start* button located on the same row as the MySQL Label.

Make sure that the MySQL label has changed its background color to green and the Start button has changed to Stop. Additional log information will display the Status as running.



After pressing start, the Xampp control panel should look similar to the image below



3. Close the Dialog Window.

Accessing the MySQL Application using Command Line Interface

1. Run the Command Line Interface (cmd) through the start menu or desktop.

2. Type the following command to navigate to the MySQL Folder:

cd c://xampp/mysql/bin

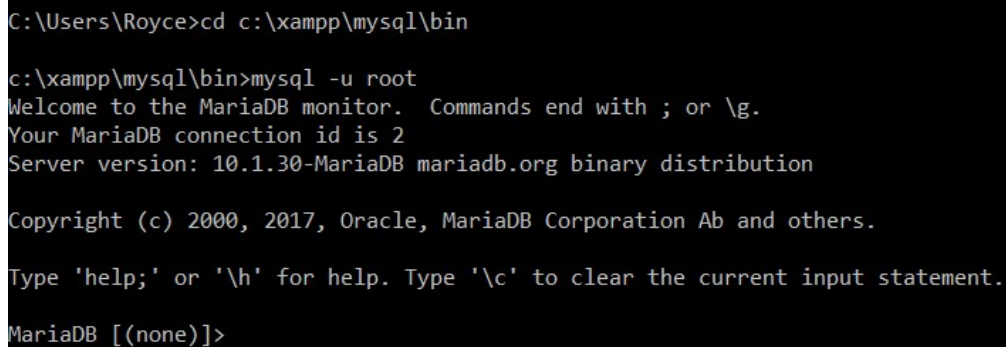
The *cd* command sets the current directory of the command line allowing you to access certain executable files and folders within that folder.

3. Type the following command to connect to the MySQL database server.

mysql.exe -u root

The command runs the program mysql.exe with an additional argument -u which means user and root meaning the root user. The MySQL database comes with a default user called root with no initial password.

4. The output should look similar to the image below:



```
C:\Users\Royce>cd c:\xampp\mysql\bin

c:\xampp\mysql\bin>mysql -u root
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 2
Server version: 10.1.30-MariaDB mariadb.org binary distribution

Copyright (c) 2000, 2017, Oracle, MariaDB Corporation Ab and others.

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

MariaDB [(none)]>
```

Navigating through the MySQL Database

1. Type the following command, show the screenshot of the output with your observation below it in **Section 6 Database Output**.

➤ **SHOW DATABASES;**

Creating a Database

1. Type the following command to create a MySQL Database. Note: You will use this database for activity 1.

➤ **CREATE DATABASE vehiclesdb;**

2. Run the **show databases** command again in the command line. Show the screenshot of the output with your observation below it in Section 6 Database Output.

Deleting a Database

1. To delete an existing database, use the command:

➤ **DROP DATABASE vehiclesdb;**

Note: You will have to recreate vehiclesdb for the next part of the activity.

2. Run the **show databases** command again in the command line. Show the screenshot of the output with your observation below it in Section 6 Database Output.

Using a Database

1. Before you are able to create a table in database, you will need to select the database you want to use. You can use the command use.
 - **USE vehiclesdb;**
2. Show the screenshot of the output with your observation below it in the Data and Results section.

Creating a Table

1. Once the database has been selected, you can create a table with the following attributes using the following command:
 - **CREATE TABLE vehicles(car_maker CHAR(20), car_model CHAR(20),number_of_doors INT);**
2. Show the screenshot of the output with your observation below it in the Data and Results section.
3. To view the tables created on the database, use the command below. Show the screenshot of the output with your observation below it in Section 6 Database Output.
 - **SHOW TABLES;**
4. To view the structure of a particular table, use the command below. Show the screenshot of the output with your observation below it in Section 6 Database Output.
 - **DESCRIBE vehicles;**

Altering the table structure

1. To change or alter the structure of an existing database, follow the sequence of commands as an example. Show the screenshot of the output of each bullet with your observation below it in Section 6 Database Output.
 - **ALTER TABLE vehicles RENAME vehicle;**
 - **DESCRIBE vehicle;**
 - **ALTER TABLE vehicle MODIFY number_of_doors INT(2);**
 - **DESCRIBE vehicle;**
 - **ALTER TABLE vehicle ADD year_model DATE;**
 - **DESCRIBE vehicle;**
 - **ALTER TABLE vehicle CHANGE year_model year_released DATE;**
 - **ALTER TABLE vehicle DROP year_model;**
 - **DESCRIBE vehicle;**
 - **ALTER TABLE vehicle RENAME vehicles;**

The table structure can be modified in the following but not limited to: **Modifying** field structure like its Data Type, **Adding** new columns, and **Renaming** the table and existing field structures.

Deleting a table

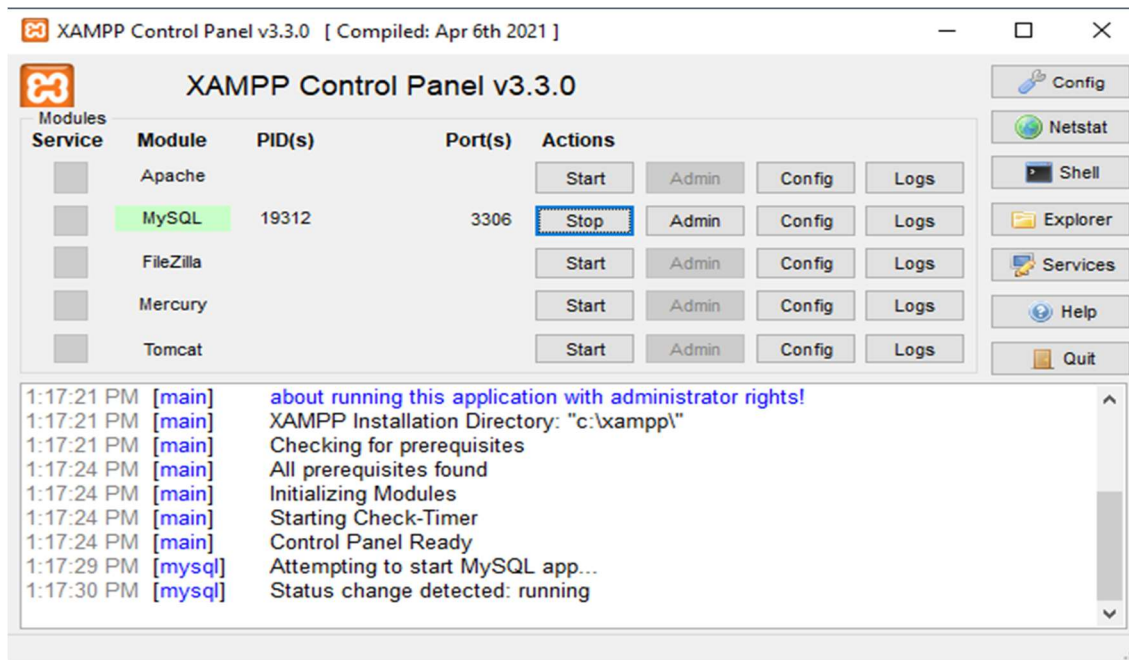
1. To delete an existing table from the database, use the command:
 - **DROP TABLE vehicles;**

6. Database Output:

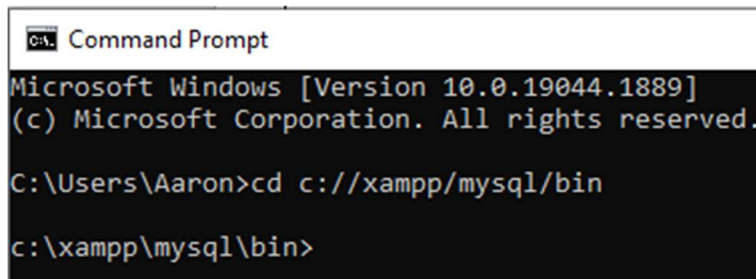
Copy screenshot(s) of your output with observations after completing the procedures provided in Part 5.

>First task is to open the xampp control panel then start mysql.

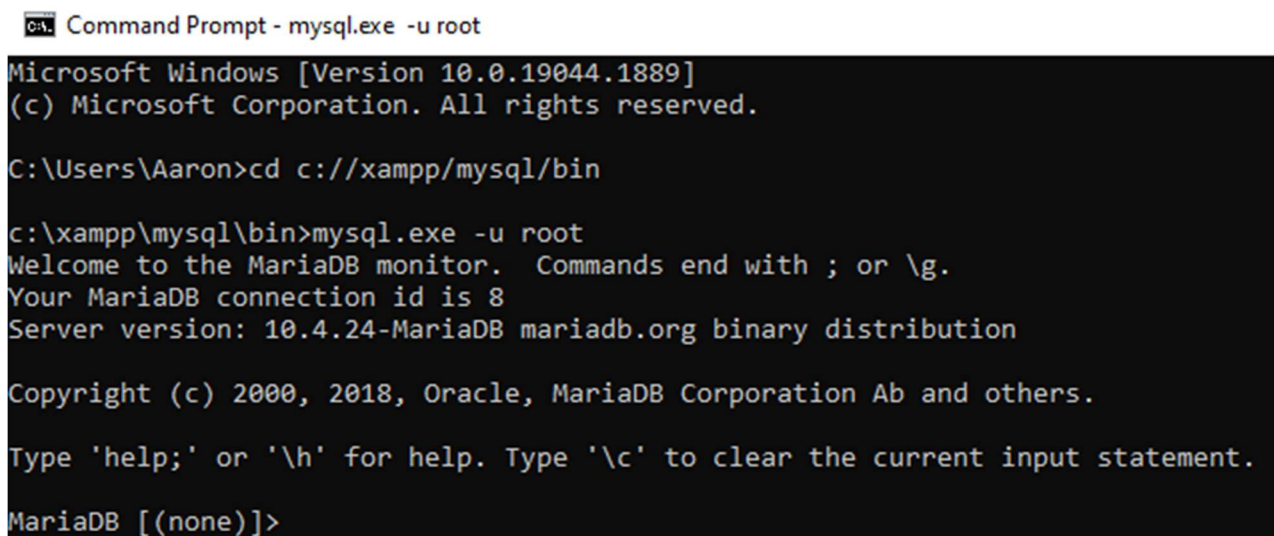
>>observation: after pressing the start button the MySQL became highlighted.



>Second task is to open the command prompt then access xampp.



> third task connect to mysql database server.



>Fourth task to input the command show databases;

>>Observation: It showed the default databases of my computer.

```
MariaDB [(none)]> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
| phpmyadmin |
| test |
+-----+
5 rows in set (0.016 sec)
```

>Fifth task create a new database & show the database.

>>After using the command the output resulted in 1 row being affected, before it showed 5 rows in the database and now it showed six .

```
MariaDB [(none)]> create database vehiclesdb;
Query OK, 1 row affected (0.001 sec)

MariaDB [(none)]>
MariaDB [(none)]> show databases
-> ;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
| phpmyadmin |
| test |
| vehiclesdb |
+-----+
6 rows in set (0.001 sec)
```

>Six task to delete a database.

>>Observation: after deleting vehiclesdb it disappeared from the rows of databases.

```
MariaDB [(none)]> drop database vehiclesdb;
Query OK, 0 rows affected (0.009 sec)

MariaDB [(none)]> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
| phpmyadmin |
| test |
+-----+
5 rows in set (0.001 sec)
```


>New task recreate the database then use it.

>>observation: MariaDB[(none)] changed to MariaDB[(vehiclesdb)]

```
MariaDB [(none)]> CREATE DATABASE vehiclesdb;
Query OK, 1 row affected (0.002 sec)

MariaDB [(none)]> USE vehiclesdb;
Database changed
MariaDB [vehiclesdb]>
```

>Second task create a table in the new database then use the describe command.

>>Observation: When the table was created it showed 1 row but did not show the tables characteristics. When the describe command was inputted it showed six columns with three rows. On the field column it showed the variables and in the type column it showed the data types. The car_maker and car_model displayed a character data type it has a character string description and I believed it has a fixed length of 20 which I specified in the input. Next is the number_of_doors it has a data type of an integer, I believed it showed a default precision number since I did not specified it in the table creation command.

```
MariaDB [vehiclesdb]> CREATE TABLE vehicles(car_maker CHAR(20), car_model CHAR(20),number_of_doors INT);
Query OK, 0 rows affected (0.034 sec)
```

```
MariaDB [vehiclesdb]>
```

```
MariaDB [vehiclesdb]> show tables;
```

```
+-----+
| Tables_in_vehiclesdb |
+-----+
| vehicles              |
+-----+
1 row in set (0.001 sec)
```

```
MariaDB [vehiclesdb]> describe vehicles;
```

```
+-----+-----+-----+-----+-----+-----+
| Field          | Type      | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| car_maker      | char(20)  | YES  |     | NULL    |       |
| car_model      | char(20)  | YES  |     | NULL    |       |
| number_of_doors | int(11)   | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.012 sec)
```

>Third task rename the table then modify a row of data.

>>observation: Even after renaming the table it did not change its characteristics. But I was able to change the characteristic of the number_of_doors yet the column of Null and Default remains unchanged.

```
MariaDB [vehiclesdb]> ALTER TABLE vehicles RENAME vehicle;
Query OK, 0 rows affected (0.025 sec)

MariaDB [vehiclesdb]> describe vehicle;
+-----+-----+-----+-----+-----+-----+
| Field          | Type    | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| car_maker      | char(20) | YES  |     | NULL    |       |
| car_model      | char(20) | YES  |     | NULL    |       |
| number_of_doors | int(11)  | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.007 sec)

MariaDB [vehiclesdb]> ALTER TABLE vehicle MODIFY number_of_doors INT(2);
Query OK, 0 rows affected (0.011 sec)
Records: 0 Duplicates: 0 Warnings: 0

MariaDB [vehiclesdb]> describe vehicle;
+-----+-----+-----+-----+-----+-----+
| Field          | Type    | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| car_maker      | char(20) | YES  |     | NULL    |       |
| car_model      | char(20) | YES  |     | NULL    |       |
| number_of_doors | int(2)   | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.005 sec)
```

>Fourth task add a new a variables and data type.

>>observation: A new row was added with the date data type it can store years months and weeks in numerical value.

```
MariaDB [vehiclesdb]> ALTER TABLE vehicle ADD year_model DATE;
Query OK, 0 rows affected (0.013 sec)
Records: 0 Duplicates: 0 Warnings: 0

MariaDB [vehiclesdb]> describe vehicle;
+-----+-----+-----+-----+-----+-----+
| Field          | Type    | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| car_maker      | char(20) | YES  |     | NULL    |       |
| car_model      | char(20) | YES  |     | NULL    |       |
| number_of_doors | int(2)   | YES  |     | NULL    |       |
| year_model     | date     | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.003 sec)
```

> Fifth task altering a table and dropping a table.

>>observation: year_model was changed to year_released and when the drop command was inputted an error occurred because it did not exist anymore. When the vehicle table name was changed then dropped it did not showed up in the vehiclesdb database anymore.

```
MariaDB [vehiclesdb]> ALTER TABLE vehicle CHANGE year_model year_released DATE;
Query OK, 0 rows affected (0.008 sec)
Records: 0 Duplicates: 0 Warnings: 0
```

```
MariaDB [vehiclesdb]> ALTER TABLE vehicle DROP year_model;
ERROR 1091 (42000): Can't DROP COLUMN `year_model`; check that it exists
MariaDB [vehiclesdb]> describe vehicle;
```

Field	Type	Null	Key	Default	Extra
car_maker	char(20)	YES		NULL	
car_model	char(20)	YES		NULL	
number_of_doors	int(2)	YES		NULL	
year_released	date	YES		NULL	

```
4 rows in set (0.005 sec)
```

```
MariaDB [vehiclesdb]> ALTER TABLE vehicle RENAME vehicles;
Query OK, 0 rows affected (0.015 sec)
```

```
MariaDB [vehiclesdb]> drop tables vehicles;
Query OK, 0 rows affected (0.013 sec)
```

7. Supplementary Activity:

Tasks

1. Create a new database **driversdb_<LASTNAME>**. Show the new list of databases on the MySQL monitor.
2. Create a table named drivers with the following attributes:
 - ID
 - First Name
 - Last Name
 - Age
 - Address
 - Vehicles
 - Years driving
 - Status

Note: (Active or Not Active)
3. Create another table named vehicles with the following attributes:
 - Car Maker
 - Car Model
 - Number of Doors
4. Display all the structure for each table. Place a screenshot of your tables in the initial output.
5. Add a new column on drivers called drivers_license with data type int.
6. Change the column of drivers_license data type to a CHAR(13)
7. Remove the field vehicles from the drivers table.
8. Display all the structure for each table. Place a screenshot of your tables in the final output.
Note: Do not drop the database or any tables.
9. Cite the commands used for this activity and its output per number using the table below:

Task	syntax	Actual SQL Command Used
Create a new Database	create database	create database driversdb_CARO;
Create a table named drivers	create table	create table drivers(id int,first_name char(100),last_name char(100),Age int,Address char(100),Vehicles int,Years_driving date,status char(100));
Create a table named vehicles	Create table	create table vehicles(car_maker char(100),car_model char(100),number_of_doors int);
Add a new column drivers_license	Alter table [name of table] add [new parameter]	alter table drivers add drivers_license int;
Change the data type of drivers_license	Alter table [name of table] change [old name] [new name]	alter table drivers change drivers_license drivers_license char(13);
Remove the field vehicles from the Drivers table	Alter table [name of table] drop [name of field]	alter table drivers drop vehicles;

Initial Output

<INSERT SCREENSHOT HERE>

```
MariaDB [driversdb_caro]> describe drivers;
```

Field	Type	Null	Key	Default	Extra
id	int(11)	YES		NULL	
first_name	char(100)	YES		NULL	
last_name	char(100)	YES		NULL	
Age	int(11)	YES		NULL	
Address	char(100)	YES		NULL	
Vehicles	int(11)	YES		NULL	
Years_driving	date	YES		NULL	
status	char(100)	YES		NULL	

```
8 rows in set (0.005 sec)
```

```
MariaDB [driversdb_caro]> describe vehicles;
```

Field	Type	Null	Key	Default	Extra
car_maker	char(100)	YES		NULL	
car_model	char(100)	YES		NULL	
number_of_doors	int(11)	YES		NULL	

```
3 rows in set (0.009 sec)
```

```
MariaDB [driversdb_caro]> describe drivers;
```

Field	Type	Null	Key	Default	Extra
id	int(11)	YES		NULL	
first_name	char(100)	YES		NULL	
last_name	char(100)	YES		NULL	
Age	int(11)	YES		NULL	
Address	char(100)	YES		NULL	
Vehicles	int(11)	YES		NULL	
Years_driving	date	YES		NULL	
status	char(100)	YES		NULL	
drivers_license	int(11)	YES		NULL	

```
9 rows in set (0.003 sec)
```


Final Output
<INSERT SCREENSHOT HERE>

```
MariaDB [driversdb_caro]> describe drivers;
```

Field	Type	Null	Key	Default	Extra
id	int(11)	YES		NULL	
first_name	char(100)	YES		NULL	
last_name	char(100)	YES		NULL	
Age	int(11)	YES		NULL	
Address	char(100)	YES		NULL	
Years_driving	date	YES		NULL	
status	char(100)	YES		NULL	
drivers_license	char(13)	YES		NULL	

8 rows in set (0.005 sec)

```
MariaDB [driversdb_caro]> describe vehicles;
```

Field	Type	Null	Key	Default	Extra
car_maker	char(100)	YES		NULL	
car_model	char(100)	YES		NULL	
number_of_doors	int(11)	YES		NULL	

3 rows in set (0.009 sec)

Questions:

1. Try and run your SQL Commands in opposite case or in mixed cases (ex. SHOW TABLES – show tables). What is the output and what is the feature of SQL that you can see in doing this task?

The output of both SQL Commands was the same whether it was mix cases, all caps, or uncapitalized. The output displayed the tables that were made from a certain database. SQL had reserved keywords when I entered the commands.

2. Attempt creating a database with a similar name, but with varying capitalization. What is the result? Why?

ERROR 1007 (HY000): Can't create database 'monke'; database exists is the result. I think it is because the database is Not case sensitive and different capitalizations always resulted in all small database name. The result showed an error message, stating that another database already existed with the same name. It was probably done to reduce the confusion, for the user, and the app.

3. Attempt creating a table with two similar names. What is the result? Why?

I was able to create two tables with similar names because just a single character difference Is enough to make into a different variable. The result showed an error message, stating that a table already exists with the same name. It was probably done to avoid confusion and unnecessary data.

4. Try creating two fields in table with two similar names with same then with different data types. What are the results? Why?

It is impossible to create two columns with exactly the same name even with different capitalization because SQL would not know in what column you want to insert values. But if the name is not the same just similar then it will work. Two tables can have exactly the same column names but if they are not connected in some other way then they have nothing to do with each other.

5. From the output of Question #4. How does MySQL perform the check to determine the output the result in Question#4? Does its data type matter?

MariaDB [driversdb caro]> alter table drivers add StaTus int; ERROR 1060 (42S21): Duplicate column name 'StaTus' .the data type does not matter because the result is a duplication error.

6. Why do you think the keywords in SQL commands written in the instructions of this manual and online references are capitalized? Does it affect the query in anyway? Why?

I think it is for readability or to highlight the syntax from the variables and parameters we use it does not affect the query in any way because SQL is not very case sensitive.

8. Conclusion:

I have concluded that SQL is an amazing database management system as it can organize, assign, create, delete and alter informations in the systems database. It can even create different databases and structures to suite the users needs. SQL DDL commands are used for creating new database objects (CREATE command), modifying existing database objects (ALTER command), and deleting or removing database objects (DROP and TRUNCATE commands).

9. Assessment Rubric: