NMK20303 Database Management System LAB
MODULE 2

Introduction To Database

FACULTY OF ELECTRONIC ENGINEERING TECHNOLOGY

Universiti Malaysia Perlis

1. Objectives

1. To understand the Structured Query Language that query or data management language.
2. To differentiate DDL and DML.
3. To use the MySQL in the database management system.

2. Equipment/software

1. XAMPP Apache + MariaDB + PHP
2. Netbeans

3. INTRODUCTION

3.1 Structured Query Language (SQL) is the most relational DBMS today that provide some types of query or data management language to access database. SQL allowed users to access information in database in a simple and flexible manner.

3.2 Why user needed SQL?

- One way to access information in a database by writing program.
- Then using the programs to access the information.
- Users must know how to write, compile, run and test running program.
- Users can perform most of their data management functions using SQL.
- They don't have to learn specific database languages such as dBASE, Informix or Oracle.

3.3 SQL Components

SQL has 2 components called:

3.3.1 Data Definition Language (DDL)

- DDL is used to create data structures such as views, schemas, tables and indexes.
- It is also used to specify integrity constrains such as domain and referential integrity.
- The main commands are as follow:-

CREATE, ALTER, DROP, TRUNCATE, COMMENT

3.3.2 Data Manipulation Language (DML)

- The DML of SQL provides powerful query features to help users manage their data.
- DML provides 4 basic commands; SELECT, INSERT, UPDATE, DELETE

4. 4.0 DATABASE MANAGEMENT SYSTEMS (DBMS)

4.1 Data type – Introduction

Definition: Data type is the characteristic of columns and variables that defines what types of data values they can store. The characteristic indicate whether a data item represents

a number, date, character string, etc. Before creating a table, identify whether a column should be a text, number, or date type. Each column in a table is made of a data type. The size of the value should be the smallest value depending upon the largest input value.

1) Numeric Data Type

Numeric Data Type	Explanation
BIT	a synonym for **TINYINT(1)**
TINYINT[(M)]	A very small integer. The signed range is **-128** to **127**. The unsigned range is **0** to **255**
BOOL, BOOLEAN	These types are synonyms for **TINYINT(1)**. A value of zero is considered false. Non-zero values are considered true.
SMALLINT	A small integer. The signed range is **-32768** to **32767**. The unsigned range is **0** to **65535**
MEDIUMINT	A medium-sized integer. The signed range is **-8388608** to **8388607**. The unsigned range is **0** to **16777215**
INT	A normal-size integer. The signed range is **-2147483648** to **2147483647**. The unsigned range is **0** to **4294967295**
INTEGER	This type is a synonym for **INT**
BIGINT	A large integer. The signed range is **-9223372036854775808** to **9223372036854775807**. The unsigned range is **0** to **18446744073709551615**
FLOAT	A small(single-precision) floating-point number. The values are from **3.402823466E+38** to **-1.175494351E-38**, **0**, and **1.175494351E-38** to **3.402823466E+38**
DOUBLE	A normal-size(double-precision) floating-point number. The values are from **1.7976931348623157E+308** to **-2.2250738585072014E-308**, **0**, and **2.2250738585072014E-308** to **1.7976931348623157E+308**
DECIMAL	The maximum number of digits(M) for **DECIMAL** is **64**.

2) Date and Time Data Type

Date and Time Data Type	Explanation
DATE	A Date. The range is 1000-01-01 to 9999-12-31. The date values are displayed in YYYY-MM-DD format.
TIME	A Time. The range is -838:59:59 to 838:59:59. The time values are displayed in HH:MM:SS format.
DATETIME	A Date and Time combination. The range is 1000-01-01 00:00:00 to 9999-12-31 23:59:59. The datetime values are displayed in YYYYMM-DD HH:MM:SS format.
TIMESTAMP	A Timestamp. The range is 1970-01-01 00:00:01 UTC to partway through the year 2037. A TIMESTAMP column is useful for recording the date and time of an INSERT or UPDATE operation.
YEAR	A Year. The year values are displayed either in two-digit or four-digit format. The range of values for a four-digit is 1901 to 2155. For two digits, the range is 70 to 69, representing years from 1970 to 2069. For all the date and time columns, we can also assign the values using either string or numbers.

3) String Data Type

String Data Type	Explanation
CHAR()	It is a fixed length string and is mainly used when the data is not going to vary much in its length. It ranges from 0 to 255 characters long. While storing CHAR values they are right padded with spaces to the specified length. When retrieving the CHAR values, trailing spaces are removed.
VARCHAR()	It is a variable length string and is mainly used when the data may vary in length. It ranges from 0 to 255 characters long. VARCHAR values are not padded when they are stored.
TINYTEXT, TINYBLOB	A string with a maximum length of 255 characters.
TEXT	A columns are treated as character strings (non-binary strings). It contains a maximum length of 65535 characters.
BLOB	BLOB stands for Binary Large OBject. It can hold a variable amount of data. BLOB columns are treated as byte strings(binary strings). It contains a maximum length of 65535 characters.
MEDIUMTEXT, MEDIUMBLOB	It has a maximum length of 16777215 characters.
LONGTEXT, LONGBLOB	It has a maximum length of 4294967295 characters.
BINARY	The BINARY is similar to the CHAR type. It stores the value as binary byte strings instead of non-binary character strings.
VARBINARY	The VARBINARY is similar to the VARCHAR type. It stores the value as binary byte strings instead of non-binary character strings.
ENUM()	An enumeration. Each column may have one of a specified possible value. It can store only one of the values that are declared in the specified list contained in the ( ) brackets. The ENUM list ranges up to 65535 values.
SET()	A set. Each column may have more than one of the specified possible values. It contains up to 64 list items and can store more than one choice. SET values are represented internally as integers.

5. 5.0 INTRODUCTION TO DATABASE

5.1 Introduction to Database System

5.1.1 Creating a Database

1) Begin by creating a sample database and the tables within it, populating its tables, and performing some simple queries on the data contained in those tables.

2) By using a database involves several steps:
   a) Creating (initializing) the database.
   b) Creating the tables within the database.
   c) Manipulating the tables by inserting, retrieving, modifying, or deleting data

5.1.2 Statement in MySQL

1) Select Database

The SELECT statement is the core of SQL, and it is likely that the vast majority of your SQL commands will be SELECT statements. You might expect that creating the database would also make it the default (or current) database, but it doesn't. You can see this by executing the following statement to check what the default database is:

Syntax:
>> SELECT DATABASE();

NULL = no database is selected

2) Creating Database

Syntax:
>> CREATE DATABASE <database_name>;

Example:
>> CREATE DATABASE NDJ20803;

3) Using Database

Syntax:
>> USE <database_name>;

Example:
>> USE NDJ20803;

4) Creating Tables

The create table statement is used to create a new table.

Syntax:
>>CREATE TABLE <table_name>

(<column_name1> <DATA TYPE>(<size>),

< column_name2> <DATA TYPE>(< size >),

< column_name3> <DATA TYPE>(< size >));

Example:

Make sure you separate each column definition with a comma. All SQLstatements should end with "**;**". The table and column names must start with a letter and can be followed by letters, numbers, or underscores - not to exceed a total of 30 characters in

length. Do not use any SQL reserved keywords as names for tables or column names (such as "**select**", "**create**", "**insert**", etc).

```
>>CREATE TABLE STUDENTS

  (STUDENTID INT(20),

  NAME VARCHAR(25),

  TELEPHONE INT(15));
```

5) Viewing Table Structure

Now that you've told MySQL to create a couple of tables, check to make sure that it did so as you expect.

Syntax;
>>DESCRIBE <table_name>;

Example:
>>DESCRIBE STUDENTS;

6) Inserting into a Table

The insert statement is used to insert or add a row of data into the table.

Syntax:
>>INSERT INTO <tablename>
(first_column,...last_column)
values (first_value,...last_value);

Example:

Note: All strings should be enclosed between single quotes: 'string'

```
>>INSERT INTO STUDENTS
   (STUDENTID, NAME,TELEPHONE)
    values (111,'AHMAD',0101555);
```

7) Viewing Inserted data

Syntax:
>>SELECT * FROM <table_name>;

Example:
>>SELECT * FROM STUDENTS;

8) Drop a Table

The drop table command is used to delete a table and all rows in the table. To delete an entire table including all of its rows, issue the drop table command followed by the table name. Drop table is different from deleting all of the records in the table. Deleting all of the records in the table leaves the table including column and constraint information. Dropping the table removes the table definition as well as all of its rows.

Syntax:-
>>DROP TABLE <tablename>;

Example:
>>DROP TABLE STUDENTS;

9) Drop a Database

The drop database command is used to remove a database.

Syntax:
>>DROP DATABASE <database_name>;

Example:
>>DROP DATABASE NDJ20803;

6.0 STORING DATA INSIDE THE DATABASE

How the data is stored in a database is probably much simpler than you might think. Databases use a series of **Tables** to store the data. A table simply refers to a two dimensional representation of your data using columns and rows. As shown below:

	Column 1	Column 2	…..	Column n
Row 1	data	data	data	data
Row 2	data	data	data	data
…	data	data	data	data
Row n	data	data	data	data

Each database table is given a unique name. Next, each column in the table also has a unique name. The columns would be something like first_name, last_name, email as examples above. This doesn't mean each column that you name has to be unique within the entire database. It only has to be unique within the table you have created. Also notice that the names don't use any spaces.

When naming tables and columns we must be sure to keep it simple with letters and numbers. We must be careful with spaces and symbols, that can be illegal characters that will mess up our works, so if you need to clarify a name use the "_" instead of spaces.

Use the table below to begin your exercise.

STUDENTS

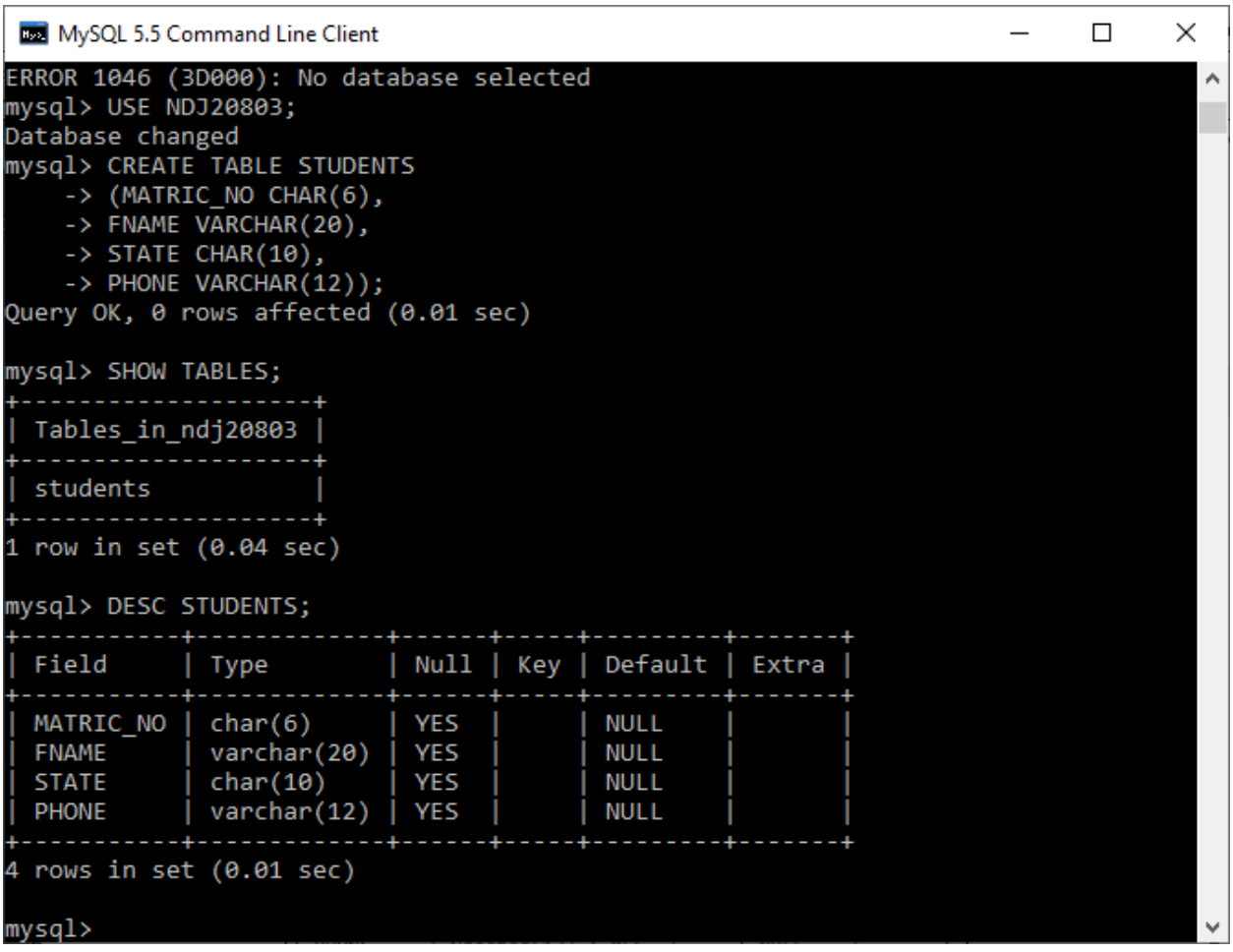
MATRIC_NO	FNAME	STATE	PHONE

A11111	**AHMAD**	**PERAK**	**011-7862312**
A11112	**ALAN**	**JOHOR**	**013-8791109**
A11113	**CHANDRAN**	**MELAKA**	**010-7681100**
A11114	**ROKIAH**	**TERENGGANU**	**019-2110929**

First of all, make sure that you already have a database. Here, database **NDJ20803** is used. Then, create a new table student as shown below:-

Syntax:

```
mysql> USE NDJ20803;
mysql> CREATE TABLE STUDENTS;
-> (MATRIC_NO CHAR(6),
-> FNAME VARCHAR(20),
-> STATE CHAR(10),
-> PHONE VARCHAR(12));
mysql> SHOW TABLES;
mysql> DESC STUDENTS;
```



SQL COMMAND : INSERT

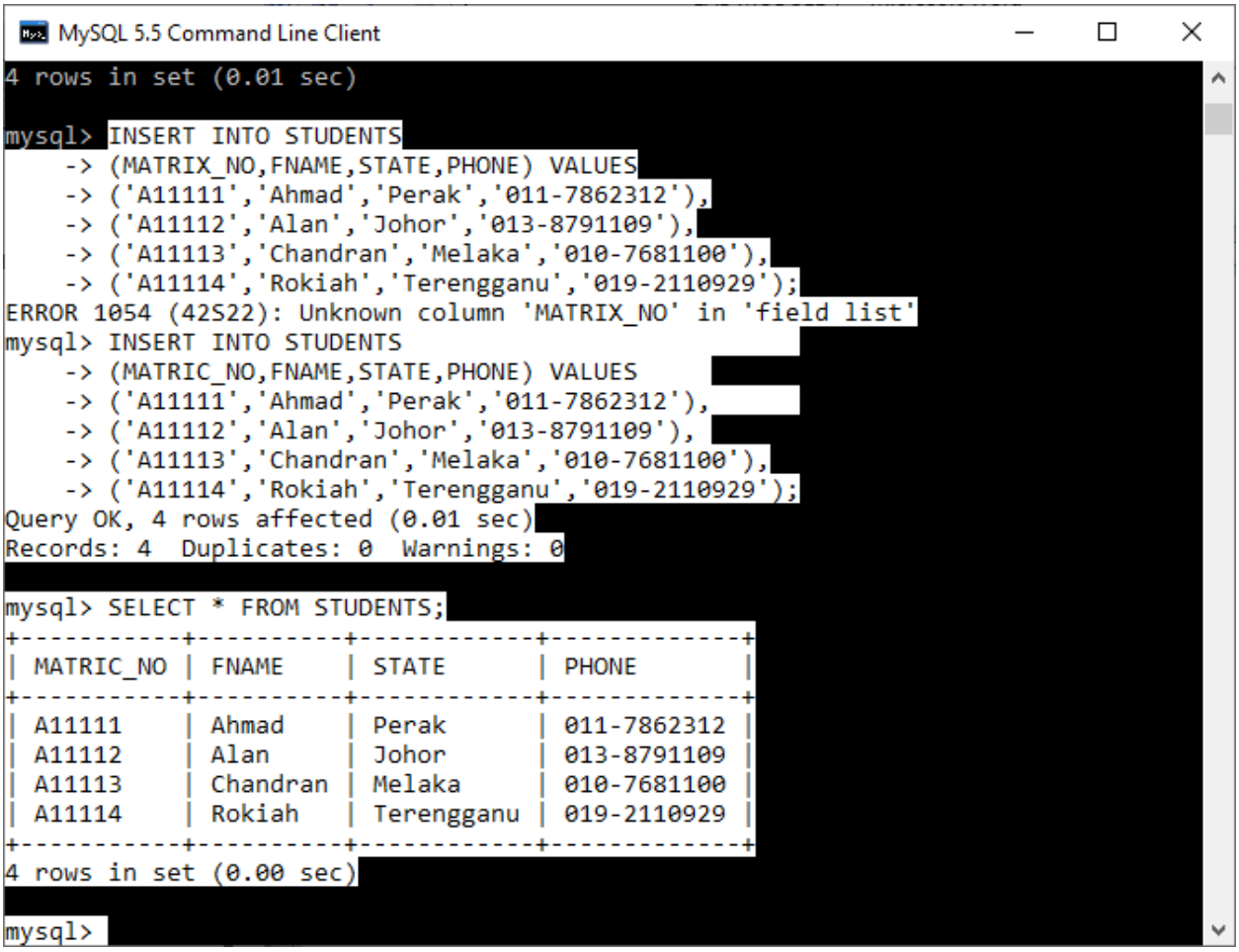
This command is used to insert a row of data in a table. All inserted values are enclosed using single quote strings.

Syntax:
mysql> INSERT INTO <table_name> (<column_list>) VALUES (<data_list>)

The SQL clause will instructs SQL to expect a grouped expression of values to follow.
VALUE (data_1, data_2 ..., data_5). There should be one value for each specified column, separated by commas. These values may be expressions themselves (e.g., an operation between two values), or constants.

Syntax:

```
mysql> INSERT INTO STUDENTS
-> (MATRIC_NO,FNAME,STATE,PHONE) VALUES
-> ('A11111','Ahmad','Perak','011-7862312'),
-> ('A11112','Alan','Johor','013-8791109'),
-> ('A11113','Chandran','Melaka','010-7681100'),
-> ('A11114','Rokiah','Terengganu','019-2110929');
```



SQL COMMAND: INSERT

Example full SQL command inserting data in the table is:-

Syntax:
mysql> INSERT INTO STUDENT (MATRIC_NO, FNAME, STATE, PHONE)
VALUES (‘A11111’, ‘Ahmad’, ‘Penang’, ‘012-77777777’);

From the above example, the **<column_list>** are **(MATRIC_NO, FNAME, STATE, PHONE)**, it will shows where the data in the **<data_list>** will be store. As example the first column, **MATRIC_NO** will receive the first data in the **<data_list>, ‘A11111’,** the second column **NAME** will receive the second data in the **<data_list>, ‘Ahmad’**. It shows that each data in the **<data_list>** will be store to column in the **<column_list>** that in the same position.

You can change the arrangement of the data in the **insert** command by changing the arrangement of the **<column_list>** and the **<data_list>**. For example if you re-arrange the columns like the example below, you can still have all the data in their appropriate columns.

Syntax:

mysql> INSERT INTO STUDENT (FNAME,MATRIC_NO,STATE,PHONE)
VALUES
('Ko','A55555','Sarawak','012-2778787');

```
mysql> INSERT INTO STUDENTS
    -> (FNAME,MATRIC_NO,STATE,PHONE) VALUES
    -> ('Ko','A11115','Sarawak','012-2778787');
Query OK, 1 row affected (0.01 sec)

mysql> SELECT * FROM STUDENTS;
+-----------+----------+------------+-------------+
| MATRIC_NO | FNAME    | STATE      | PHONE       |
+-----------+----------+------------+-------------+
| A11111    | Ahmad    | Perak      | 011-7862312 |
| A11112    | Alan     | Johor      | 013-8791109 |
| A11113    | Chandran | Melaka     | 010-7681100 |
| A11114    | Rokiah   | Terengganu | 019-2110929 |
| A11115    | Ko       | Sarawak    | 012-2778787 |
+-----------+----------+------------+-------------+
5 rows in set (0.00 sec)
```

But if you re-arrange the **<column_list>** but without re-arrange the **<data_list>** like the example below:-

Syntax:
mysql> INSERT INTO STUDENT (MATRIC_NO, NAME, STATE,
VALUES ('SIDI', 'A11116', 'Kelantan', '019-5566889');

```
mysql> INSERT INTO STUDENTS
    -> (MATRIC_NO,FNAME,STATE,PHONE) VALUES
    -> ('Sidi','A11116','Kelantan','019-5566889');
Query OK, 1 row affected (0.03 sec)

mysql> SELECT * FROM STUDENTS;
+-----------+----------+------------+-------------+
| MATRIC_NO | FNAME    | STATE      | PHONE       |
+-----------+----------+------------+-------------+
| A11111    | Ahmad    | Perak      | 011-7862312 |
| A11112    | Alan     | Johor      | 013-8791109 |
| A11113    | Chandran | Melaka     | 010-7681100 |
| A11114    | Rokiah   | Terengganu | 019-2110929 |
| A11115    | Ko       | Sarawak    | 012-2778787 |
| Sidi      | A11116   | Kelantan   | 019-5566889 |
+-----------+----------+------------+-------------+
6 rows in set (0.00 sec)

mysql>
```

INSERT DATA INSIDE CERTAIN COLUMN USING SQL COMMAND: INSERT

You will not always have to list all the columns from a table to insert a new record. For example if you have only data **NAME**, and **MATRIC_NO** for a student, you still can insert the data like below:-

INSERT INTO STUDENT (MATRIC_NO, NAME) VALUES ('A77777', 'Ummul');

The command above will store data **'A77777'** and **'Ummul'** in columns **MATRIC_NO** and **NAME**. It is the same if you write:-

Syntax:
mysql> **INSERT INTO STUDENT (MATRIC_NO, FNAME)**
VALUES ('A77777', 'Ummul');

...where defining a null data. You can insert other left data using SQL command: UPDATE which will be discuss in the next section.

```
mysql> INSERT INTO STUDENT (MATRIC_NO,FNAME) VALUES
    -> ('A77777','Ummul');
ERROR 1146 (42S02): Table 'ndj20803.student' doesn't exist
mysql> INSERT INTO STUDENTS (MATRIC_NO,FNAME) VALUES
    -> ('A77777','Ummul');
Query OK, 1 row affected (0.01 sec)

mysql> SELECT * FROM STUDENTS;
+-----------+----------+------------+-------------+
| MATRIC_NO | FNAME    | STATE      | PHONE       |
+-----------+----------+------------+-------------+
| A11111    | Ahmad    | Perak      | 011-7862312 |
| A11112    | Alan     | Johor      | 013-8791109 |
| A11113    | Chandran | Melaka     | 010-7681100 |
| A11114    | Rokiah   | Terengganu | 019-2110929 |
| A11115    | Ko       | Sarawak    | 012-2778787 |
| Sidi      | A11116   | Kelantan   | 019-5566889 |
| A77777    | Ummul    | NULL       | NULL        |
+-----------+----------+------------+-------------+
7 rows in set (0.00 sec)

mysql>
```

7.0 CHANGING DATA STORE IN DATABASE

SQL COMMAND: UPDATE

This command is used to update records for a single table only.

Syntax:
mysql> UPDATE <table_name> SET <column_name>=<new_value> WHERE <filter_condition>

There are 3 main parts on the above command line:-
a) **UPDATE <table_name>** specifies a target table to update. You can include multiple sources of data for the update operation in the **FROM** clause.
b) **SET <column_name>=<new_value>** specifies the column in the table to update and it new values. It possible to update more then one column using update command, just added another column to be update and it new data. The list of columns to be update must be separate

by a comma (**,**). Example **SET<column_name>=<new_value>, <column_name2>=<new_value2>.**

c) **WHERE <filter_condition>** specifies one or more filter conditions that records must meet to be updated with new values. If no filter condition exist SQL will update all the record inside the table.

Syntax:
mysql> UPDATE STUDENTS SET STATE='JOHOR' WHERE MATRIC_NO ='A44444';
mysql> SELECT * FROM STUDENTS;

Command above will update all the record that had value '**A44444**' in it **MATRIC_NO** column.

Value for **STATE** column will be update to '**JOHOR**'.

```
mysql> UPDATE STUDENTS SET STATE='JOHOR' WHERE MATRIC_NO
    -> ='A44444';
Query OK, 0 rows affected (0.00 sec)
Rows matched: 0  Changed: 0  Warnings: 0

mysql> SELECT * FROM STUDENTS;
+-----------+----------+------------+-------------+
| MATRIC_NO | FNAME    | STATE      | PHONE       |
+-----------+----------+------------+-------------+
| A11111    | Ahmad    | Perak      | 011-7862312 |
| A11112    | Alan     | Johor      | 013-8791109 |
| A11113    | Chandran | Melaka     | 010-7681100 |
| A11114    | Rokiah   | Terengganu | 019-2110929 |
| A11115    | Ko       | Sarawak    | 012-2778787 |
| Sidi      | A11116   | Kelantan   | 019-5566889 |
| A77777    | Ummul    | NULL       | NULL        |
+-----------+----------+------------+-------------+
7 rows in set (0.00 sec)
```

UPDATE ALL RECORD INSIDE THE TABLE

SQL command **UPDATE** also allowed a data change to entire record of a database table. In the query editor write:-

Syntax:
mysql> UPDATE <table_name> SET <attribute>=<new data>

Example:-
Syntax:
mysql> UPDATE STUDENTS SET STATE='Perlis';
mysql> SELECT * FROM STUDENTS;

All the record for **STATE** will change to **Perlis**.

```
mysql> UPDATE STUDENTS SET STATE='Perlis';
Query OK, 7 rows affected (0.01 sec)
Rows matched: 7  Changed: 7  Warnings: 0

mysql> SELECT * FROM STUDENTS;
+-----------+----------+--------+-------------+
| MATRIC_NO | FNAME    | STATE  | PHONE       |
+-----------+----------+--------+-------------+
| A11111    | Ahmad    | Perlis | 011-7862312 |
| A11112    | Alan     | Perlis | 013-8791109 |
| A11113    | Chandran | Perlis | 010-7681100 |
| A11114    | Rokiah   | Perlis | 019-2110929 |
| A11115    | Ko       | Perlis | 012-2778787 |
| Sidi      | A11116   | Perlis | 019-5566889 |
| A77777    | Ummul    | Perlis | NULL        |
+-----------+----------+--------+-------------+
7 rows in set (0.00 sec)
```

8.0 DELETE TABLE RECORD

SQL COMMAND: DELETE

The **DELETE** command removes from a table those rows that fulfill some specific condition.

Syntax:
mysql> DELETE FROM <table_name> WHERE <filter_condition>

The function of **WHERE <filtercondition>** in the command line is to delete specific rows.

Syntax:
mysql> DELETE FROM STUDENTS WHERE FNAME='Ahmad';

```
mysql> DELETE FROM STUDENTS WHERE FNAME='Ahmad';
Query OK, 1 row affected (0.00 sec)

mysql> SELECT * FROM STUDENTS;
+-----------+----------+--------+-------------+
| MATRIC_NO | FNAME    | STATE  | PHONE       |
+-----------+----------+--------+-------------+
| A11112    | Alan     | Perlis | 013-8791109 |
| A11113    | Chandran | Perlis | 010-7681100 |
| A11114    | Rokiah   | Perlis | 019-2110929 |
| A11115    | Ko       | Perlis | 012-2778787 |
| Sidi      | A11116   | Perlis | 019-5566889 |
| A77777    | Ummul    | Perlis | NULL        |
+-----------+----------+--------+-------------+
6 rows in set (0.00 sec)
```

The command above will find all rows containing **'Ahmad'** data inside column **FNAME** and delete all the rows. If you have more than one filter condition you can add other condition using **Boolean operator** like **AND** or **OR**.

Syntax:
mysql> DELETE FROM STUDENTS WHERE STATE = 'Selangor' AND MATRIC_NO='A22222';

The command above will delete rows that fulfill the above filter condition only. The OR operator can be use to delete rows that fulfill either one filter condition.

To delete all record inside a table, cut out the **WHERE** part from the **DELETE** command line.

Syntax:
mysql> DELETE FROM STUDENTS;

9.0 ALTERING TABLE STRUCTURE

SQL COMMAND: ALTER TABLE

Add new column inside a table.

Syntax:
mysql> ALTER TABLE <table_name> ADD <new_colum> <data_type>

As example, added a new column "**AGE**" into table **STUDENTS**:-

Syntax:
mysql> ALTER TABLE STUDENTS ADD AGE INT(2);

```
mysql> ALTER TABLE STUDENTS ADD AGE INT(2);
Query OK, 6 rows affected (0.05 sec)
Records: 6  Duplicates: 0  Warnings: 0

mysql> SELECT * FROM STUDENTS;
+-----------+----------+--------+-------------+------+
| MATRIC_NO | FNAME    | STATE  | PHONE       | AGE  |
+-----------+----------+--------+-------------+------+
| A11112    | Alan     | Perlis | 013-8791109 | NULL |
| A11113    | Chandran | Perlis | 010-7681100 | NULL |
| A11114    | Rokiah   | Perlis | 019-2110929 | NULL |
| A11115    | Ko       | Perlis | 012-2778787 | NULL |
| Sidi      | A11116   | Perlis | 019-5566889 | NULL |
| A77777    | Ummul    | Perlis | NULL        | NULL |
+-----------+----------+--------+-------------+------+
6 rows in set (0.00 sec)

mysql> DESC STUDENTS;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| MATRIC_NO | char(6)     | YES  |     | NULL    |       |
| FNAME     | varchar(20) | YES  |     | NULL    |       |
| STATE     | char(10)    | YES  |     | NULL    |       |
| PHONE     | varchar(12) | YES  |     | NULL    |       |
| AGE       | int(2)      | YES  |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
5 rows in set (0.02 sec)
```

Make sure you chose the suitable data type for the new column. If you want to add **IC_NO** after **FNAME** then use keyword **AFTER**.

Syntax:
mysql> ALTER TABLE STUDENTS ADD IC_NO VARCHAR(14) AFTER FNAME;

```
mysql> ALTER TABLE STUDENTS ADD IC_NO VARCHAR(14) AFTER FNAME;
Query OK, 6 rows affected (0.05 sec)
Records: 6  Duplicates: 0  Warnings: 0

mysql> DESC STUDENTS;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| MATRIC_NO | char(6)     | YES  |     | NULL    |       |
| FNAME     | varchar(20) | YES  |     | NULL    |       |
| IC_NO     | varchar(14) | YES  |     | NULL    |       |
| STATE     | char(10)    | YES  |     | NULL    |       |
| PHONE     | varchar(12) | YES  |     | NULL    |       |
| AGE       | int(2)      | YES  |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
6 rows in set (0.04 sec)
```

UPDATE USING ARITHMETIC FORMULA

For any columns that hold numeric data like integer and decimal, it is possible to update using arithmetic formula. Example:-

Syntax:
mysql> UPDATE STUDENTS SET AGE=18 WHERE MATRIC_NO = 'A11112';

mysql> UPDATE STUDENTS SET AGE=AGE+2 WHERE MATRIC_NO = 'A11112';

```
mysql> UPDATE STUDENTS SET AGE=18 WHERE MATRIC_NO='A11112';
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> SELECT * FROM STUDENTS;
+-----------+----------+-------+--------+-------------+------+
| MATRIC_NO | FNAME    | IC_NO | STATE  | PHONE       | AGE  |
+-----------+----------+-------+--------+-------------+------+
| A11112    | Alan     | NULL  | Perlis | 013-8791109 |   18 |
| A11113    | Chandran | NULL  | Perlis | 010-7681100 | NULL |
| A11114    | Rokiah   | NULL  | Perlis | 019-2110929 | NULL |
| A11115    | Ko       | NULL  | Perlis | 012-2778787 | NULL |
| Sidi      | A11116   | NULL  | Perlis | 019-5566889 | NULL |
| A77777    | Ummul    | NULL  | Perlis | NULL        | NULL |
+-----------+----------+-------+--------+-------------+------+
6 rows in set (0.00 sec)

mysql> UPDATE STUDENTS SET AGE=AGE+2 WHERE MATRIC_NO='A11112';
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> SELECT * FROM STUDENTS;
+-----------+----------+-------+--------+-------------+------+
| MATRIC_NO | FNAME    | IC_NO | STATE  | PHONE       | AGE  |
+-----------+----------+-------+--------+-------------+------+
| A11112    | Alan     | NULL  | Perlis | 013-8791109 |   20 |
| A11113    | Chandran | NULL  | Perlis | 010-7681100 | NULL |
| A11114    | Rokiah   | NULL  | Perlis | 019-2110929 | NULL |
| A11115    | Ko       | NULL  | Perlis | 012-2778787 | NULL |
| Sidi      | A11116   | NULL  | Perlis | 019-5566889 | NULL |
| A77777    | Ummul    | NULL  | Perlis | NULL        | NULL |
+-----------+----------+-------+--------+-------------+------+
6 rows in set (0.00 sec)
```