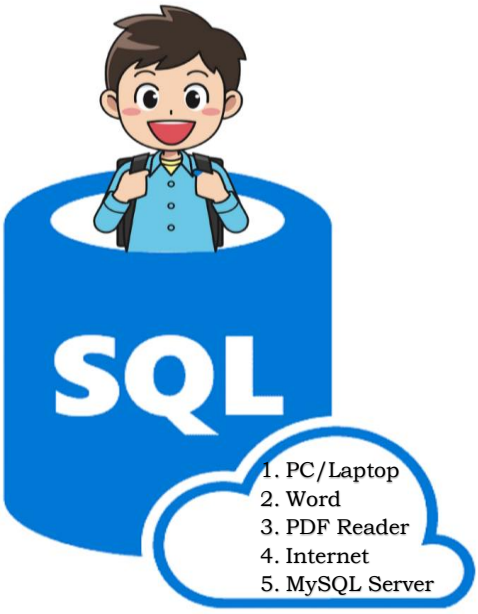
[Laboratory No. 5:
Basic DDL SQL command – NO Constraints and Referential Integrity]

Objectives

1. Setting up and writing SQL commands using CLI (Command Line Interface)
2. To create a database via a CLI, show list and be able to remove it.
3. To create a table and its structure using basic SQL DDL commands

Materials



Background

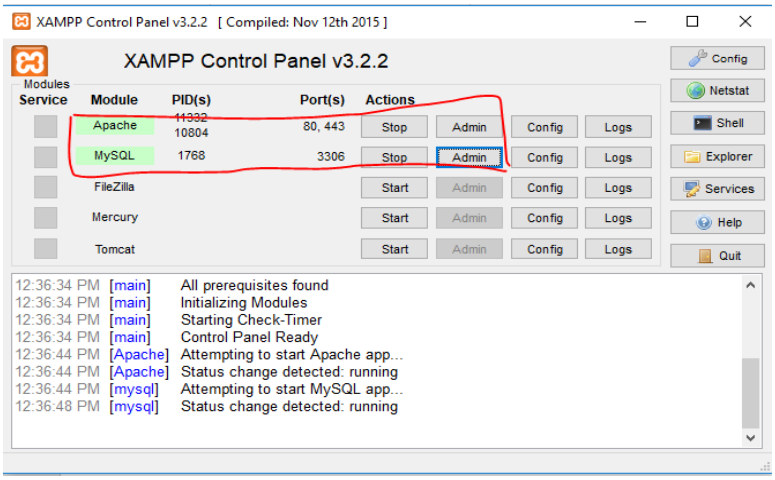
Dr. E. F. Codd published the paper, "A Relational Model of Data for Large Shared Data Banks," in June 1970 in the Association of Computer Machinery (ACM) journal, Communications of the ACM. Codd's model is now accepted as the definitive model for relational database management systems (RDBMS). The Structured English Query Language (SEQUEL) was developed by IBM Corporation, Inc., to use Codd's model. SEQUEL later became SQL (still pronounced "sequel"). In 1979, Relational Software, Inc. (now Oracle) introduced the first commercially available SQL implementation. Today, SQL is accepted as the standard RDBMS language. *(*https://docs.oracle.com/cd/B12037_01/server.101/b10759/intro001.htm*)*.

SQL is the standard language used to communicate with relational database management systems, including Oracle, Microsoft SQL Server, Sybase, Informix, and even Microsoft Access. With SQL, you can build databases, enter data into the database, manipulate data, and query the data used to make intelligent personal or business decisions. SQL is a simple, English-like language that is relatively easy to learn and use at any database user level. Today, we will get to familiarize some of the most basic SQL statements you must know in database development.

Instructions:

Setting-up Command Prompt

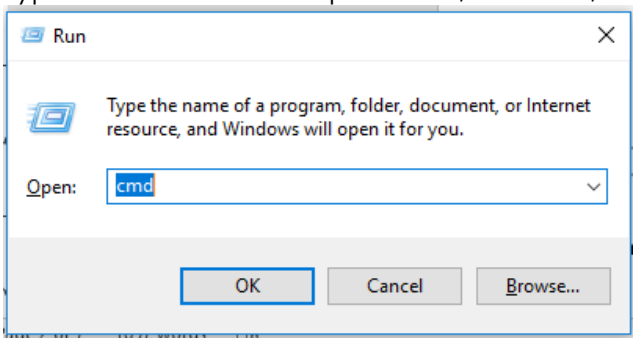
1. Start your **XAMPP** control panel, and then hit the **START** button to ***Apache*** and ***MySQL***.



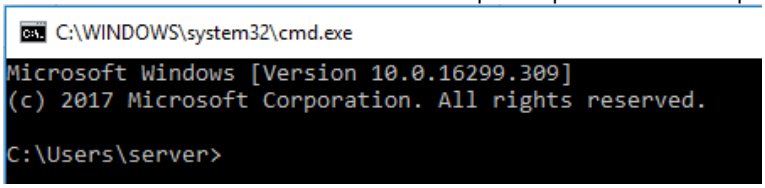
Prepared by: ***Martzel P. Baste***

2. On your keyboard, key in  to run the command prompt. (Or **terminal** in MAC device)

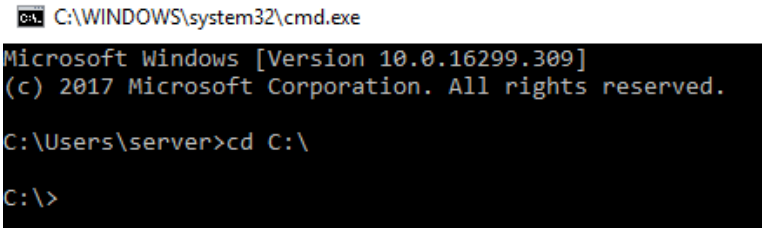
3. Type **"cmd"** in the field provided, click OK, or hit **'ENTER.'**



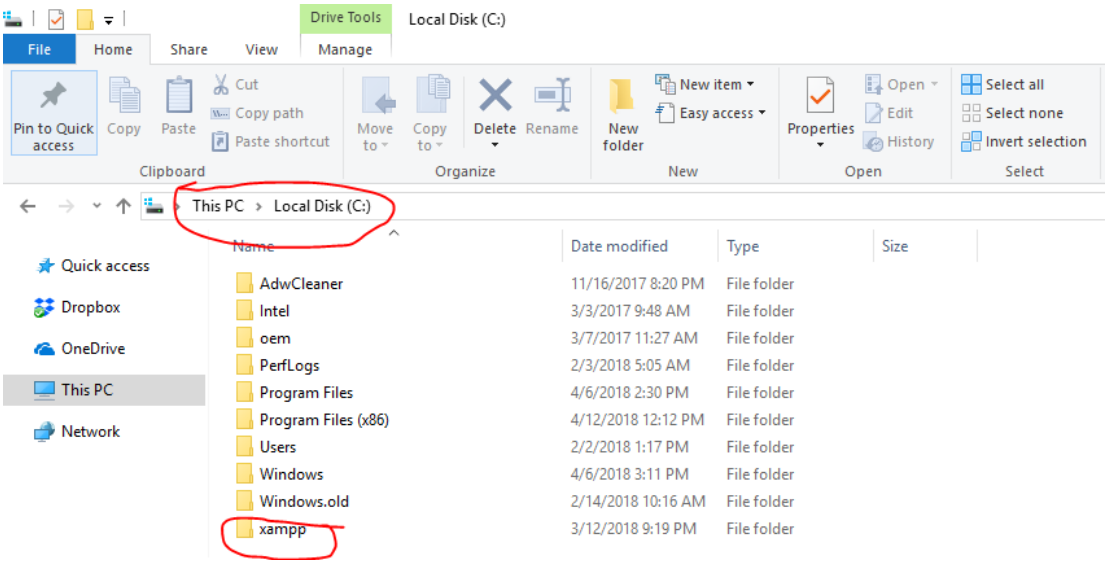
4. You will be shown the command prompt like the display below:



5. The command below means you're changing the directory to drive **C:\.**



NOTE: This will only work if you installed your XAMPP in the drive C:\. To check, head to that directory and see for yourself. If it was on another drive/directory, use that location instead.



6. Now, invoke the following commands below, then hit **ENTER**.

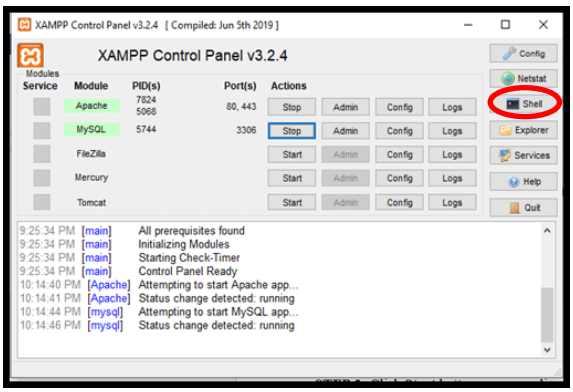
```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 10.0.16299.309]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\server>cd C:\

C:\>cd xampp\mysql\bin

C:\xampp\mysql\bin>
```

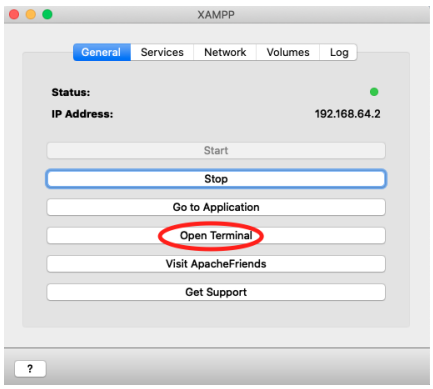
7. See commands underlined in red. Just hit **ENTER** throughout until you will see this:

   NOTE: Don't type anything on the password unless you want to. By now, it is optional.

```
C:\WINDOWS\system32\cmd.exe - mysql  -u root -p
Microsoft Windows [Version 10.0.16299.309]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\server>cd C:\

C:\>cd xampp\mysql\bin

C:\xampp\mysql\bin>mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 56
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)]>
```

   If you don't want to go through steps 2 to 6 on your **XAMPP** control panel, look for ***shell*** (Windows) or ***Terminal*** (MAC OS).





8. Now, you are all set. You can show database files on your localhost by invoking the command below:

```
C:\WINDOWS\system32\cmd.exe - mysql  -u
Copyright (c) 2000, 2017, Oracle,

Type 'help;' or '\h' for help. Ty

MariaDB [(none)]> show databases;
+----------------------+
| Database             |
+----------------------+
| central_perk database |
| db_books             |
| db_grading           |
| db_pos               |
| db_pos2              |
```

9. Great job, fellas!

Prepared by: ***Martzel P. Baste***

*** [DDL-Table Structures and Specifications] ***

DDL- Data Definition Language (DDL) statements define the database structure or schema. Data Definition Language understanding with database schemas describes how the data should be in the database. Therefore, language statements like *CREATE TABLE or ALTER TABLE belong to the DDL*. **DDL is about "metadata."**

DDL includes commands such as CREATE, ALTER, and DROP statements used to CREATE, ALTER, or DROP the database objects (Table, Views, Users).

Data Definition Language (DDL) is used different statements:

- **CREATE** - to create objects in the database
- **ALTER** - alters the structure of the database
- **DROP** - delete objects from the database
- **TRUNCATE** - remove all records from a table, including all spaces allocated for the records that are removed
- **COMMENT** - add comments to the data dictionary
- **RENAME** - rename an object

*Preparatory task

1. Create a database with a name following the format

   DB_<last name>_activityBasicSQL.

   Example: ***db_BASTE_ActivityBasicSQL.***

```
MariaDB [(none)]> CREATE database db_BASTE_ActivityBasicSQL;
Query OK, 1 row affected (0.04 sec)
```

   Your turn:

SQL/Output

2. To show if a database is created, use the following command: ***SHOW databases;***

```
C:\WINDOWS\system32\cmd.exe - mysql  -u root -p
MariaDB [(none)]> CREATE database db_BASTE_ActivityBasicSQL;
Query OK, 1 row affected (0.04 sec)

MariaDB [(none)]> show databases;
+--------------------------+
| Database                 |
+--------------------------+
| central_perk database    |
| db_baste_activitybasicsql |
```

   Your turn:

SQL/Output

3. Now, whenever you want to invoke changes to a specific database, like adding tables, changing the name of the tables, etc., you will need to use the **USE *database_name*** command. Below is an example:

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

Your turn:

SQL/Output

4. Now let's get started by creating tables on your database.

NOTE: Like in any Programming Language, you must know the idea of datatype. In a database, it means every field on your table holds a value. So, by familiarizing the different datatypes in the database, you will know what appropriate datatype you will use for a field. To learn more about datatypes, kindly visit these URLs.

https://www.techonthenet.com/mysql/datatypes.php
https://www.tutorialspoint.com/mysql/mysql-data-types.htm

Your turn (Choose at least 10):

Datatype	***Sample Declaration such as gender char(6)***

****Formative task***

5. Write a database description (tables) for each relation shown below using DDL (Data Definition Language).

Please follow the instructions and provide a screenshot (or paste it into a notepad) ***for each action you performed. Your name MUST be visible in each output.***

READ THIS

NOTE that spelling errors are intended for a specific purpose; kindly follow the instructions and format carefully.

Prepared by: ***Martzel P. Baste***

1. Create a table **"Pet."** Your table should have the following field specifications:

Fields	Datatype
Pet_ID	int(8)
Pet_Name	varchar(50)
Pet_Gender	char(6)
Pet_Color	varchar(30)
Pet_Age	tinyint(3)
Pet_Type	char(20)

Again, just a friendly reminder, your name MUST be visible in each output just like below:

```
MariaDB [db BASTE ActivityBasicSQL]> CREATE table PET (pet_id int(8), pet_name varchar(50), pet_gender char(6),
pet_color varchar(15), pet_age tinyint(3), pet_type char(20));
```

The above SQL script will display the output below if successful and no error is recognized.

```
Query OK, 0 rows affected (0.033 sec)
```

SQL/Output

2. To show the tables you have created, you can use the following statement:

```
SHOW tables;
```

Now try to write the command and show the output below.

```
MariaDB [db_BASTE_ActivityBasicSQL]> SHOW tables;
+-----------------------------------+
| Tables_in_db_BASTE_ActivityBasicSQL |
+-----------------------------------+
| PET                               |
+-----------------------------------+
```

NOTE: SQL in nature is NOT case sensitive. Usually, MySQL converts all table names to lowercase on storage and lookup. But this only applies to Windows-based OS. Tables and Columns are Case Sensitive in Linux. However, in OSX, it depends on how it was formatted.

Use lower cases for your database, table, and fields as recommended.

SQL/Output

3. If ever you want to look at the **metadata** of a table, you may use either any of the following DDL-SQL statements:

```
MariaDB [db_BASTE_ActivityBasicSQL]> DESCRIBE PET;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| pet_id     | int(8)      | YES  |     | NULL    |       |
| pet_name   | varchar(50) | YES  |     | NULL    |       |
| pet_gender | char(6)     | YES  |     | NULL    |       |
| pet_color  | varchar(15) | YES  |     | NULL    |       |
| pet_age    | tinyint(3)  | YES  |     | NULL    |       |
| pet_type   | char(20)    | YES  |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
```

The above command may be written as

```
MariaDB [db_BASTE_ActivityBasicSQL]> DESC PET;
```

```
MariaDB [db_BASTE_ActivityBasicSQL]> SHOW FIELDS FROM PET;
```

*NOTE that by invoking the statements above, you have just shown the **Data Dictionary** of your table.*

Now try to write the command and show the output below.

SQL/Output

4. In case you want to change the name of your table, you can always change it using the command below:

   To rename the table, you should use **RENAME TABLE TO** statements.

   Syntax : RENAME TABLE tb1 TO tb2;
   Example : *RENAME TABLE pet to pet_shop;*

   Now, try it by renaming your created table to ***tbl_pet.***

SQL/Output

Read: Did you know that conventions in naming are essential in Software development and so in databases? It is used for the readability and clarity of the table to what it represents. Hence, as recommended, when you name your table, you should always start with ***tbl_name_of_the_table*** (pascal-based) or ***tblNameOftheTable*** (camel-cased). Note that database scripts are not case-sensitive; hence it is best to use pascal cases.
Example:

tbl_pet or tblPet would represent records for Pet.

Thus, for payroll, employees, and professors, you should use

tbl_payroll, tbl_employees, tbl_professor.

Also, you should name-prefix your columns/fields based on the table that represents it, such as ***pet_id or petID, pet_name or petName, professor_name or professorName***, etc.

Be consistent in naming!

5. Before you proceed, can you invoke this SQL command below:

```
MariaDB [db_BASTE_ActivityBasicSQL]> ALTER TABLE tbl_owner ADD pet_status varchar(30) COMMENT 'status of pet
    '>
```

Sometimes, in the middle of our endeavor towards writing some SQL scripts, it is normal to encounter the above issues. If you experience it, you may type **"exit"** and start from the beginning. But note that you won't lose any data or table structures that were invoked previously. However, to avoid going back through the start, you will need to write the below scripts right after.

```
'> \c'
-> c
-> \c
```

OR

```
'> '\c
```

And it will return to the normal CLI, and you may proceed.

```
MariaDB [db_BASTE_ActivityBasicSQL]>
```

```
ALTER TABLE tbl_pet ADD pet_status varchar(30) COMMENT 'status of pet';
```

The above command will create a comment (as a description) for the field **pet_status**. You will see it in your phpMyAdmin.

Or by invoking the SQL command below.

```
MariaDB [db_BASTE_ActivityBasicSQL]> SHOW FULL COLUMNS FROM tbl_pet;
+------------+-------------+--------------------+------+-----+---------+-------+---------------------------------+---------------+
| Field      | Type        | Collation          | Null | Key | Default | Extra | Privileges                      | Comment       |
+------------+-------------+--------------------+------+-----+---------+-------+---------------------------------+---------------+
| pet_id     | int(8)      | NULL               | YES  |     | NULL    |       | select,insert,update,references |               |
| pet_name   | varchar(50) | utf8mb4_general_ci | YES  |     | NULL    |       | select,insert,update,references |               |
| pet_gender | char(6)     | utf8mb4_general_ci | YES  |     | NULL    |       | select,insert,update,references |               |
| pet_color  | varchar(15) | utf8mb4_general_ci | YES  |     | NULL    |       | select,insert,update,references |               |
| pet_age    | tinyint(3)  | NULL               | YES  |     | NULL    |       | select,insert,update,references |               |
| pet_type   | char(20)    | utf8mb4_general_ci | YES  |     | NULL    |       | select,insert,update,references |               |
| pet_status | varchar(3)  | utf8mb4_general_ci | YES  |     | NULL    |       | select,insert,update,references | status of pet |
+------------+-------------+--------------------+------+-----+---------+-------+---------------------------------+---------------+
```

NOTE that it is not the same with MySQL comment that is used to document your SQL statements or procedure which SQL parser ignores, such as

Line comments

```
SHOW FULL COLUMNS FROM tbl_pet; #this is a comment
```

```
SHOW FULL COLUMNS FROM tbl_pet; #this is a comment
```

Multi-line comments

```
MariaDB [db_BASTE_ActivityBasicSQL]> /* This is a
   /*> multi line
   /*> comment */
```

MySQL provides executable comments to support portability between different databases. These comments allow you to embed SQL code that will execute only in MySQL but not in other databases.

```
MariaDB [db_BASTE_ActivityBasicSQL]> SELECT 1 /*! +2 */;
+-------+
| 1  +2 |
+-------+
|     3 |
+-------+
```

```
MariaDB [db_BASTE_ActivityBasicSQL]> SELECT 1 /*! +5 */;
+-------+
| 1  +5 |
+-------+
|     6 |
+-------+
```

6. Alright then, let us proceed. I want you to create another table, **"Owner."** Your owner table should have the following fields:

NOTE: Wrong spellings are intended for a purpose. Kindly follow.

Fields	Datatype
Owner_ID	int(8)
Owner_Name	varchar(30)
Own_Address	Text
Owner_Ginger	char(6)

```
[db_BASTE_ActivityBasicSQL]> create table owner (owner_id int(8), owner_name varchar(30),
                             own_address text, owner_ginger char(6));
```

SQL/Output

Prepared by: ***Martzel P. Baste***

7. To apply the naming convention, rename the table ***owner to tbl_owner.***

SQL/Output

8. Add a new table **tbl_Buyer** with the following specifications:

Fields	Datatype
Buyer_ID	int(8)
Buyer _Name	varchar(30)
Buyer_Address	text
Buyer_Gender	char(6)

SQL/Output

9. **ADDING COLUMNS OR FIELDS**. The SQL **ALTER TABLE** statement is used to add, modify, or drop/delete columns in a table. It is also used to rename a table.

- **Adding one column**. To add a column\fields in a table, the SQL ALTER TABLE syntax is:

  ALTER TABLE table_name ADD column_name column_definition;

  Example:

```
ALTER TABLE tbl_pet add pet_breed varchar(30);
```

- **Adding two or more columns.** Let's look at the SQL **ALTER TABLE** example that adds more than one column\fields.

  Example:

```
ALTER TABLE tbl_pet ADD (pet_address varchar(50), pet_breed varchar(30));
```

- Now, complete adding fields to your table **tbl_pet** based on the following fields specifications:

Fields	Datatype
Pet_Breed	Varchar(30)
Pet_Price	Float(10,2)
Pet_contact	Varchar(11)
Pet_address	text

SQL/Output

- Show your data dictionary to see the changes you have made.

SQL/Output

10. Okay then. Let us try adding one new column or field for **tbl_owner: Owner_Contact.**

Describe and show the update of your table definition here:

SQL/Output

11. **RENAME THE COLUMN IN THE TABLE.** To rename a column in an existing table, the SQL ALTER TABLE syntax is:

ALTER TABLE table_name CHANGE COLUMN old_name new_name datatype;

Example:

```
ALTER TABLE tbl_owner CHANGE COLUMN own_address owner_address VARCHAR(100);
```

Rename the columns or fields of your **tbl_owner** based on the following requirement specifications**:**

Field name	New Field Name	Datatype
own_address	***owner_address***	Text
owner_ginger	***owner_gender***	char(6)

SQL/Output

12. **DROP COLUMN IN TABLE.** You found out that ***pet_contact and pet_address*** fields of **tbl_pet** are unnecessary since pets don't have these. Whatever the owner's address is, the address of the Pet too. The same applies to contact. So, you must remove the columns ***pet_contact and pet_address from tbl_pet***.

To drop a column in an existing table, the SQL ALTER TABLE syntax is:

ALTER TABLE table_name DROP COLUMN column_name;

Example:

```
ALTER TABLE tbl_pet DROP COLUMN pet_address;
```

Now, do as instructed above (item 11).

SQL/Output

13. **ADDING FIELDS AT A CERTAIN POSITION.**

If you want to add a single column after a specific field, then the command should be:

ALTER TABLE tbl_owner ADD *owner_weight* char(6) AFTER owner_id;

If you want to add a single column after a specific field, then the command should be:

ALTER TABLE tbl_owner ADD *owner_weight* char(6) FIRST;

Modify your **tbl_Owner** by adding the following columns or fields:

Fields	Datatype	Position
Owner_Status	int(8)	***First***
Owner_Email	varchar(30)	***After*** **owner_contact** field

SQL/Output

You can also modify the sequence/arrangements of the existing table fields using the following commands:

```
ALTER TABLE tbl_owner modify owner_email varchar (10) FIRST;
```

```
ALTER TABLE tbl_owner modify owner_status varchar (10) AFTER owner_gender;
```

SQL/Output

Or modify the datatype of an existing field or column of a table.

```
ALTER table tbl_owner MODIFY owner_address text;
```

SQL/Output

14. As a result of your invoked SQL commands previously, **owner_id** is at the 2nd field position. Did you know that most database programmers believe that all **IDs or Primary keys** should be on every table's first position or column? Hence, how do you want to make up your table in a way acceptable to every database programmer?

SQL/Output

15. Drop/Remove the table **"tbl_buyer"** by invoking the ***DROP TABLE tbl_buyer*** statement.

SQL/Output

16. You can use the following command to put a comment on your table's field.

```
ALTER TABLE tbl_owner MODIFY owner_status varchar(30)
   COMMENT 'Owner status if single, married,etc';
```

SQL/Output

NOTE: Comments are ignored by the interpreter/compiler, just like any Programming Language.

17. You can describe your table to see the changes you've made.

SQL/Output

18. Type the word "Meow" in the comment section upon submission.

19. Submit the file in a .PDF format as "***DB_LAB5_<Surname>***".

Insights
Write your takeaways in the blank provided.

Task	Total Points	Score
Preparatory Task • **Task 1** • **Task 2** • **Task 3** • **Task 4**	**25** **5** **5** **5** **10**	
Formative Tasks Screenshots	**95**	
Insights	**30**	
TOTAL	**100**	