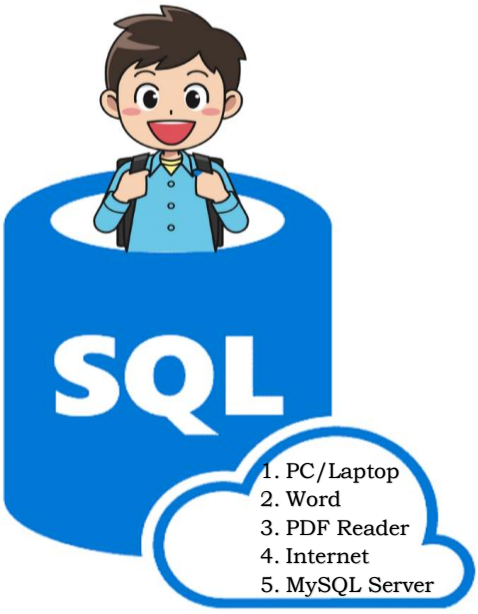
[Laboratory No. 6: Basic DML SQL 1]

Objectives

1. To store, search, retrieve, and remove records from a database using SQL-DML commands.

Materials



Background

DML- Data Manipulation Language (DML) statements are used for managing data within schema objects. DML deals with data manipulation, including most common SQL statements such as SELECT, INSERT, etc. DML allows to add/modify/delete data itself.

DML is used to manipulate the existing data in the database. Below are the commands:

1. **INSERT**
2. **SELECT**
3. **UPDATE**
4. **DELETE**

Instructions:

Create a database with a name following the format DB_<**last name>_activityBasicDML1**

Example: ***DB_BASTE_ActivityBasicDML1.***

READ

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

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

Preparatory Task

**** [DML-Records Manipulation] ****

INSERT STATEMENT stores single or multiple records into a table in MySQL.

SYNTAX

INSERT INTO table *(column1, column2, ...)*
VALUES *(expression1, expression2, ...), (expression1, expression2, ...),...;*

a. Create a ***tbl_pet*** with the following metadata.

Pet_id	int(12)
Pet_name	varhar(30)
Pet_age	tinyint(3)
Pet_gender	char(6)

```
CREATE table tbl_pet( pet_id int(12), pet_name varchar(3), pet_age tinyint(3), pet_gender char(6));
```

The above command can be written as:

```
MariaDB [DB_BASTE_ActivityBasicDML1]> CREATE table tbl_pet(
    -> pet_id int(12),
    -> pet_name varchar(3),
    -> pet_age tinyint(3),
    -> pet_gender char(6));
```

SQL/Output

b. Invoke the SQL statement below to insert a record into your table.

```
INSERT INTO tbl_pet VALUES (1,'Nymeria','female',5);
```

The above code will cause an error since the order of your values doesn't match the order of your table fields, particularly in pet_age of tinyint type and pet_gender as char type. Note that the SQL command above will be processed successfully if they are both characters.

Try this,

```
INSERT INTO tbl_pet VALUES (1,'female',5,'Nymeria');
```

But the above code is also unsuccessful because data is too long for column 'pet_gender' as it only stores up to 6 characters.

Now try this one,

```
INSERT INTO tbl_pet VALUES (1,'female',5,'Nymeri');
```

The above command will be successful, and the table will get its first record. However, the records are physically incorrect because pet_name and pet_gender do not match the required values.

Write the SQL command below to check

```
MariaDB [DB_BASTE_ActivityBasicDML1]> SELECT *from tbl_pet;
+--------+----------+---------+------------+
| pet_id | pet_name | pet_age | pet_gender |
+--------+----------+---------+------------+
|      1 | female   |       5 | Nymeri     |
+--------+----------+---------+------------+
```

Awesome! Now, do this

```
INSERT INTO tbl_pet VALUES (2,'Nymeria',5,'female');
```

SQL/Output

c. Using a SELECT command, you can display the records in the **tbl_pet** table. Invoke the following statement:

```
SELECT *from tbl_pet;
```

SQL/Output

That's it, easy! You can also show records based on a specific field.

```
SELECT pet_id, pet_name from tbl_pet;
```

SQL/Output

Observation

d. What if you want to insert two or more rows or records? Here's how.

```
[DB_BASTE_ActivityBasicDML1]> INSERT INTO tbl_pet VALUES(3,'Jj',15,'male'),(4,'Kelly',8,'female');
```

The above statement can be written as

```
MariaDB [DB_BASTE_ActivityBasicDML1]> INSERT INTO tbl_pet VALUES(3,'Jj',15,'male'),
    -> (4,'Kelly',8,'female');
```

SQL/Output

Or, if you want to store only specific values or fields in your table, you do this:

```
INSERT INTO tbl_pet(pet_id, pet_gender) VALUES(5,'male');
```

```
SELECT *from tbl_pet;
```

SQL/Output

Observation

e. Using your **tbl_pet** table, store these values as an additional record:

Pet_ID	Pet_Name	Pet_Age	Owner_Gender
1	Amenadiel	5	Female
4	Lucifer Morningstar	4	Male

SQL/Output

*Now, show the records of your **tbl_pet** table using the **SELECT** command.*

SQL/Output

Observation

NOTE: You will observe that you can store records having duplicate IDs even if they have different information. Hence, it is called data redundancy, and it must be avoided. For now, let us leave it as it is. We will have another activity intended for that purpose.

f. You just tried removing the **tbl_buyer** table from your previous one using the **DROP** command. This time I was hoping you could use the **TRUNCATE TABLE** command. To do it, see below:

TRUNCATE TABLE tbl_pet;

SQL/Output

Observation

NOTE: You will observe that all the records of your table tbl_pet are gone but NOT the tbl_pet table structure. Unlike when using the DROP command, the table and its records are all eradicated from the database. For your information, DROP and TRUNCATE commands belong to DDL SQL.

Let's do this! ***Formative Task***

1. Create a new table **tbl_owner** with the following fields, as shown below. *Use data types appropriately.*

Owner_ID	*Owner_Name*	*Owner_Address*	*Owner_Gender*	*Owner_Contact*

SQL/Output

2. Using your **tbl_owner** table, store these values: ***1," Cain", "Davao City", "female," 2229614.***

   Your table should become the following:

Owner_ID	*Owner_Name*	*Owner_Address*	*Owner_Gender*	*Owner_Contact*
1	***Cain***	***Davao City***	***Female***	***2229614***

SQL/Output

2. Show records of your **tbl_owner** table using a **SELECT** statement. Invoke the SQL statement below to accomplish the task:

   SELECT *from tbl_owner;

SQL/Output

3. Modify your created table ***tbl_pet*** earlier with newly added fields, as shown below. Use datatype appropriately.

Pet_ID	Pet_Name	Pet_Gender	Pet_Color	Pet_Age	Pet_Type	Pet_Breed	Pet_Price

SQL/Output

4. Using the **tbl_pet** table, store this information (sequence):

   1, "Anna", "Female", "Pinkish White", 12,'Dog', "Husky", 2300.45.

   (Show your records after)

   Your table should have become:

Pet_ID	Pet_Name	Pet_Gender	Pet_Color	Pet_Age	Pet_Type	Pet_Breed	Pet_Price
1	Anna	female	Pinkish White	12	Dog	Husky	8900.00

SQL/Output

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

5. Insert information for **tbl_pet** at least 5 of your specifications and preference.

SQL/Output

6. Insert the following information into your **tbl_pet**:
(***Look at the table below carefully for what columns or fields are used upon storing values.***)

Pet_id	Pet_name	Pet_gender	Pet_age
1001	Nymeria	female	5
1002	Luh Kay		
3	Ismael Kah Conti	female	16
4	***Anne ann***		***12***

Again, please observe that you can store pet information with the same Pet_id since it wasn't specified as the primary key. Hence, redundancy is possible.

SQL/Output

7. Show records of your **tbl_pet.** Display **pet_name, pet_gender, pet_type, and pet_price** only**.**

SQL/Output

8. Display all fields from the table **tbl_pet. [DDL].**

SQL/Output

9. Show the final and updated records of your **tbl_pet.**

SQL/Output

10. Insert information to your table **tbl_owner** at least 3 of your specifications and preference.

SQL/Output

11. Show records of your **tbl_owner.**

SQL/Output

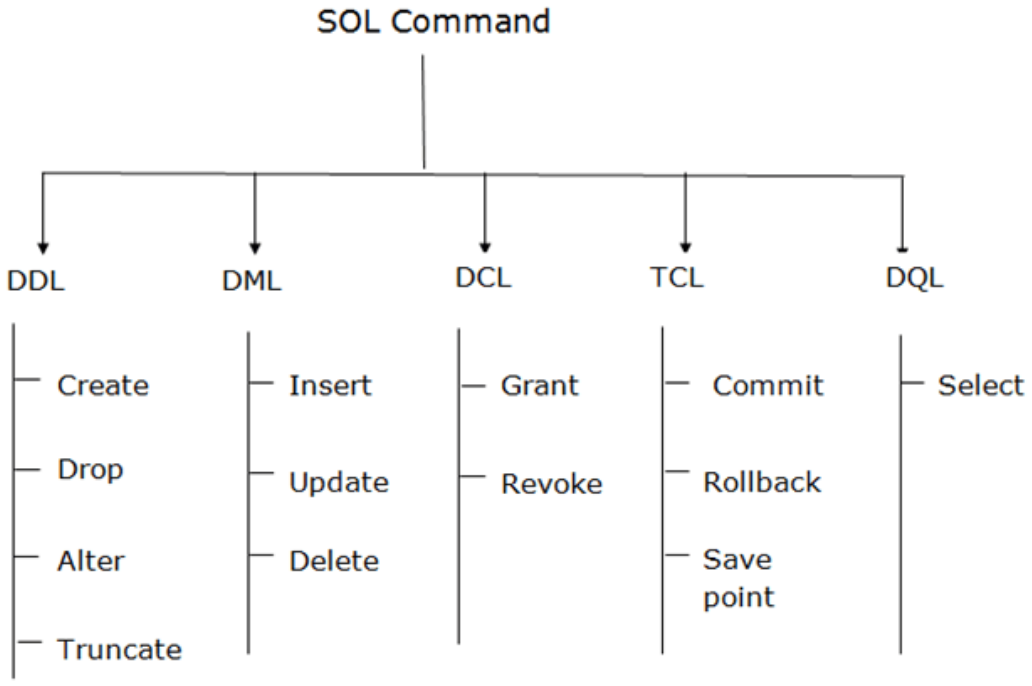
12. Display data dictionary of **tbl_owner and tbl_pet. [DDL].**

SQL/Output

13. Type "Hapee!" in the comment section upon submitting.

Congratulations! You have just made your first database application. To summarize, you have encountered some basic SQL statements. The first part was about creating a database, and tables, describing and specifying its structures. Some commands are ***CREATE, ALTER, DROP, RENAME, SHOW, AND DESCRIBE,*** which belong significantly to **DDL or Data Definition Language**.

On the other hand, we have **DML or Data Manipulation Language** which you are taught to store information in a table using **INSERT** and display this information using a **SELECT** statement. Some of the SQL commands under DML are **DELETE and UPDATE**. In other books, ***SELECT*** is under DQL or Data Query language. For reference, look at the figure below:



https://www.javatpoint.com/dbms-sql-command

Task	Total Points	Score
Preparatory Task • **Task a** • **Task b** • **Task c** • **Task d** • **Task e** • **Task f**	**80** **10** **10** **15** **15** **15** **15**	
Formative Tasks Screenshots • **Task 1** • **Task 2** • **Task 3** • **Task 4** • **Task 5** • **Task 6** • **Task 7** • **Task 8** • **Task 9** • **Task 10** • **Task 11** • **Task 12**	**120** **10** **10** **10** **10** **10** **10** **10** **10** **10** **10** **10** **10**	
TOTAL	**200**	