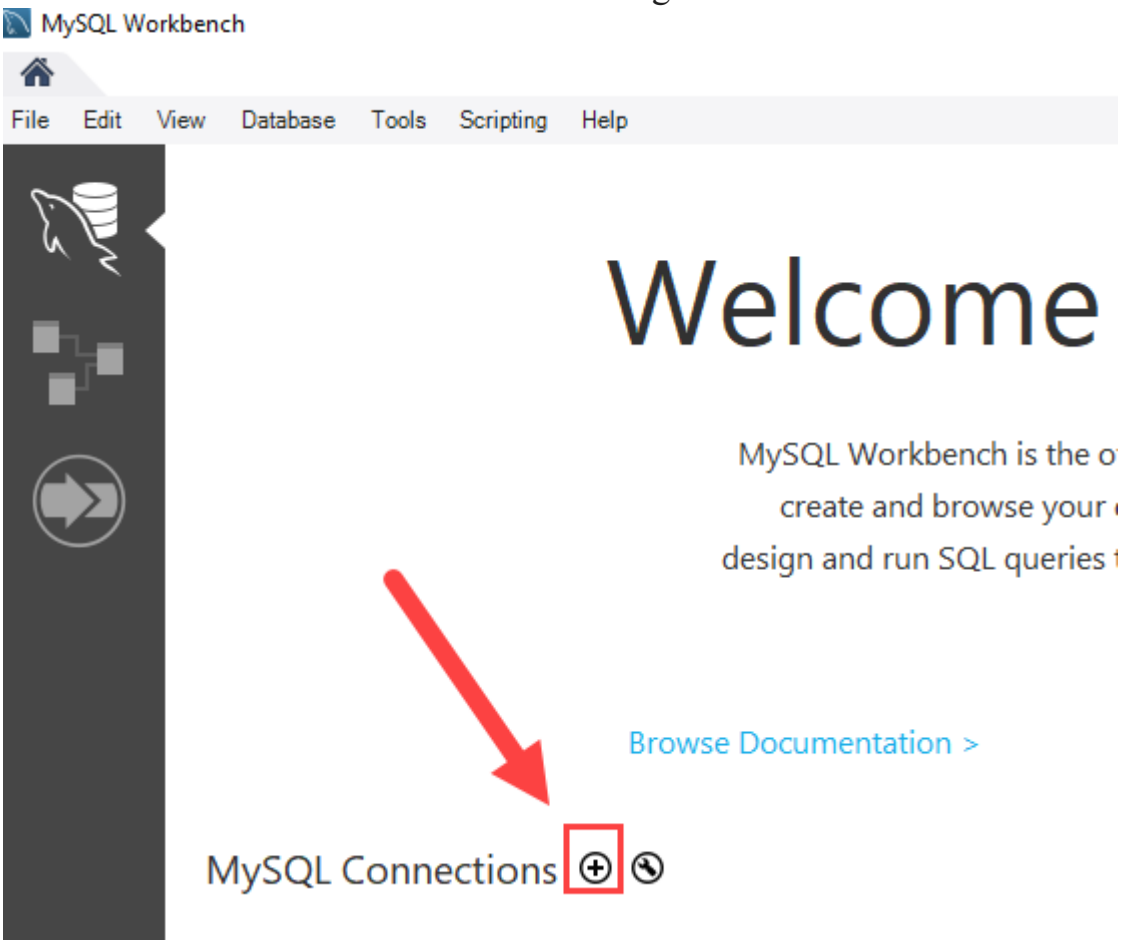
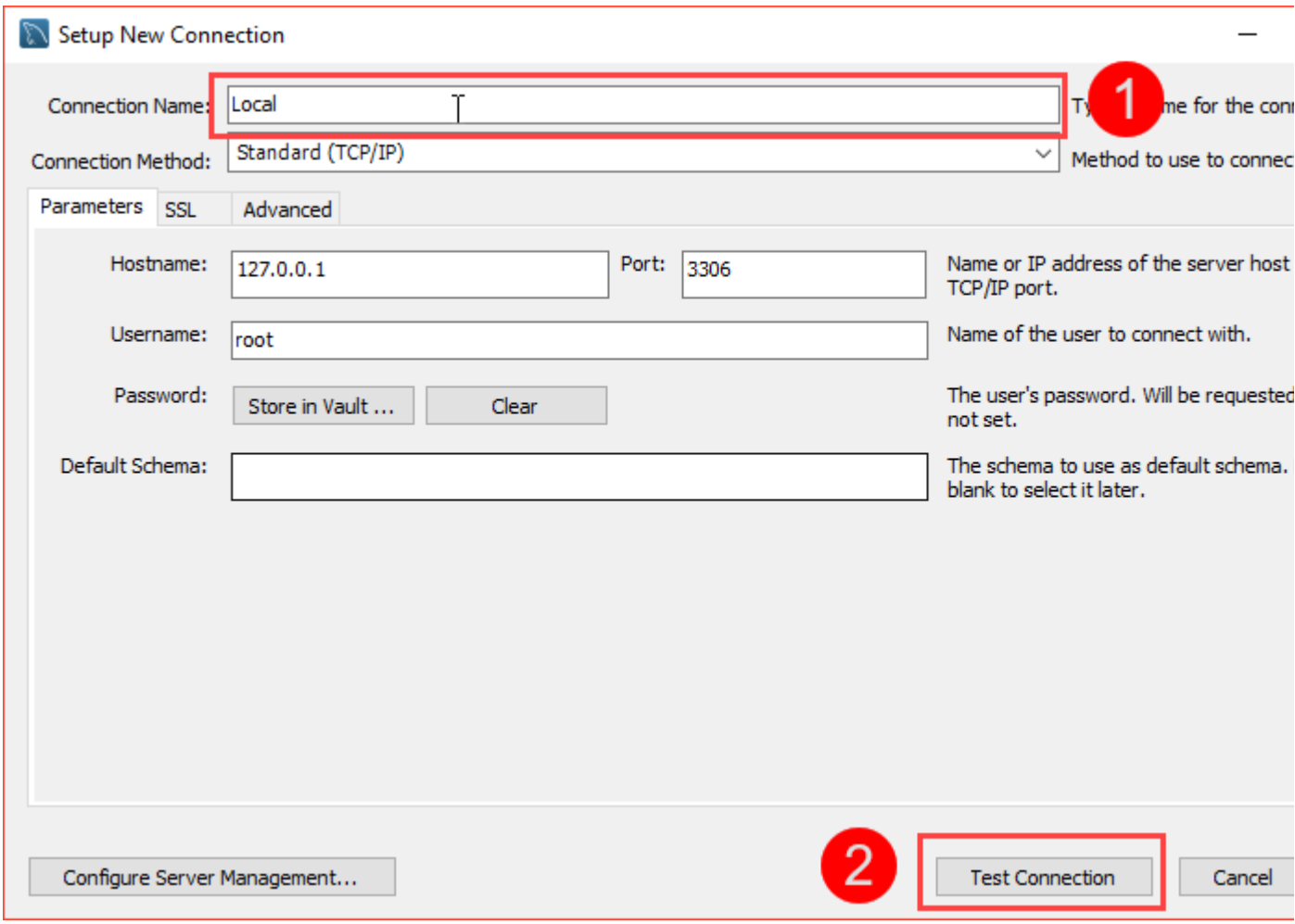
**Creating a new database using MySQL Workbench**

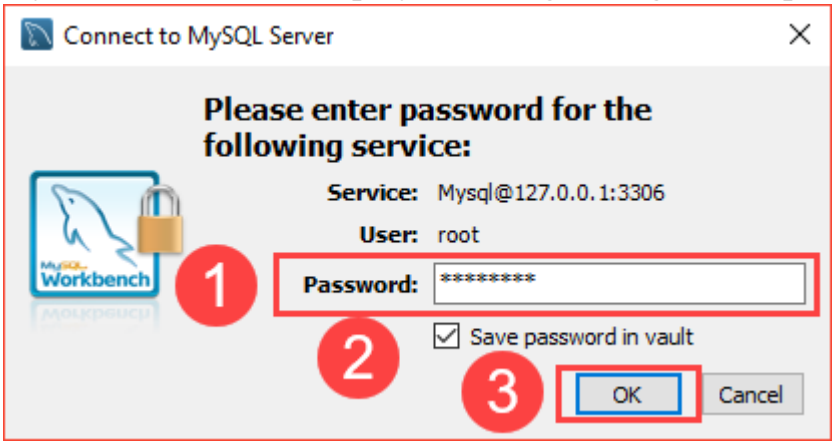
To create a new database using the MySQL Workbench, you follow these steps: First, launch the MySQL Workbench and click the **setup new connection** button as shown in the following screenshot:




Second, type the name for the connection and click the **Test Connection** button.

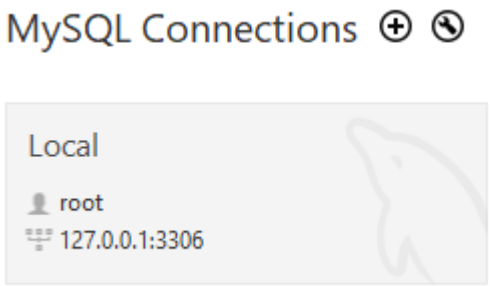



MySQL Workbench displays a dialog asking for the password of the root user:

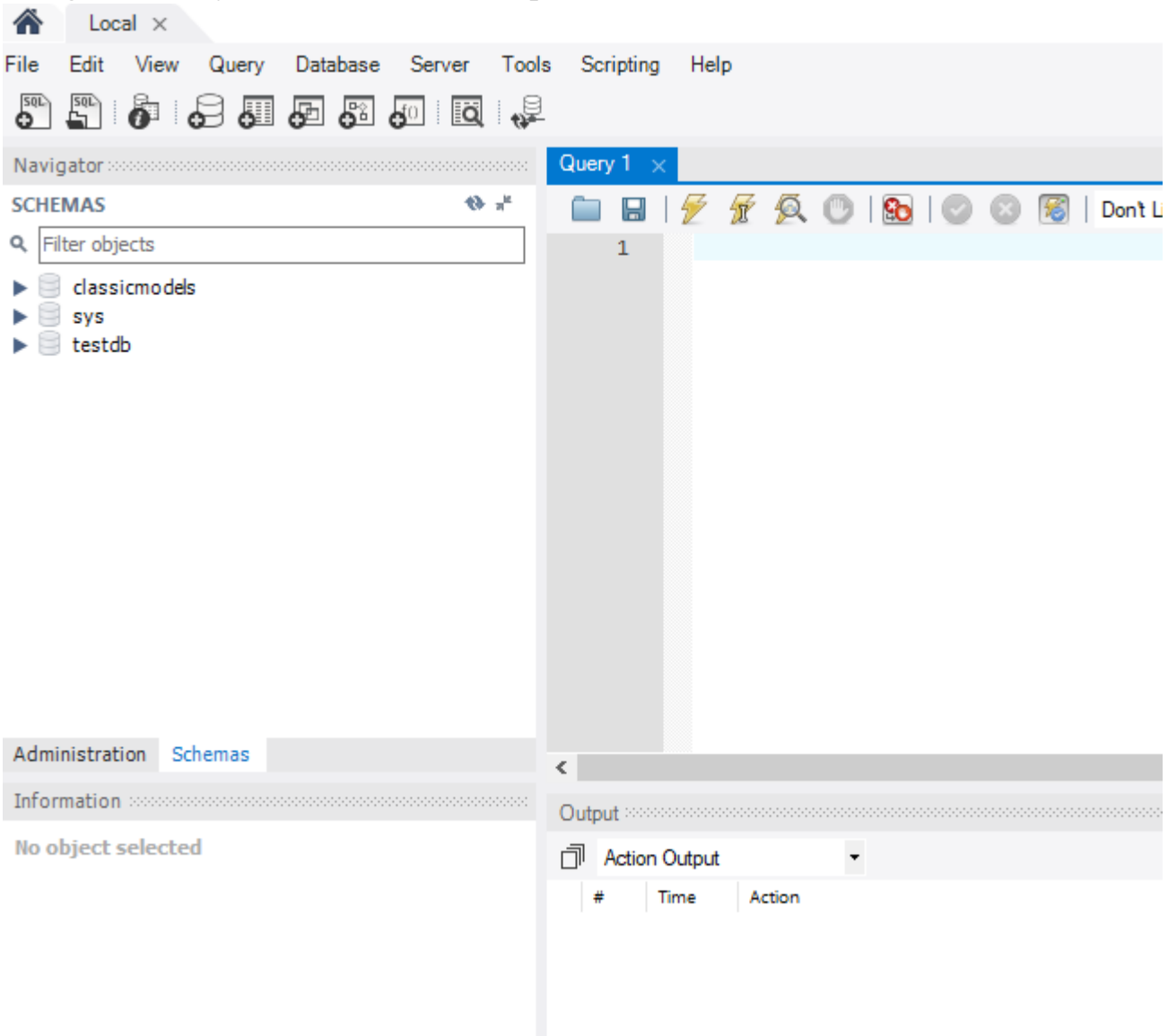



You need to (1) type the password for the root user, (2) check the **Save password in vault**, and (3) click **OK** button.

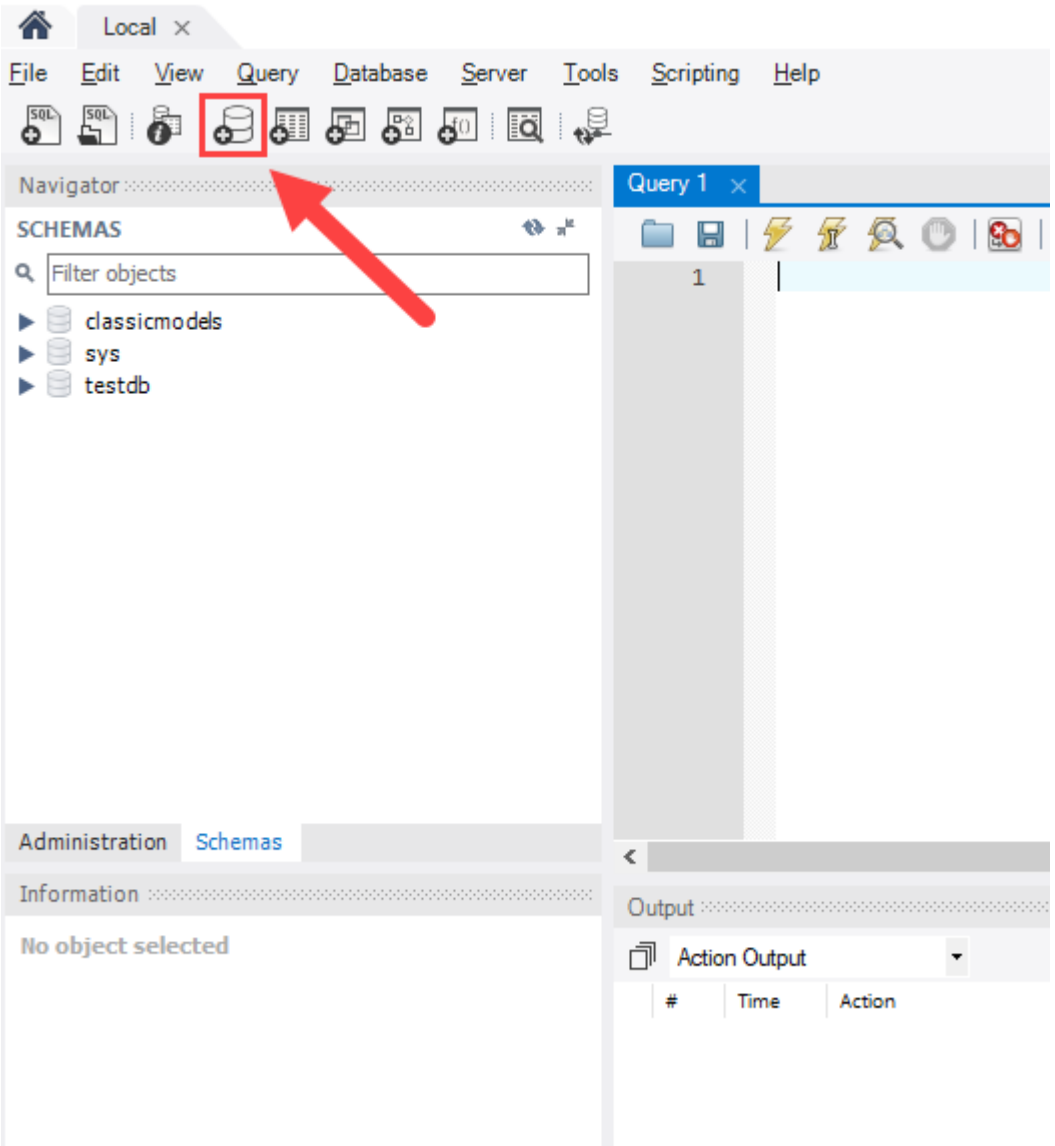
Third, double-click the connection name **Local** to connect to the MySQL Server.




MySQL Workbench opens the following window which consists of four parts: Navigator, Query, Information, and Output.

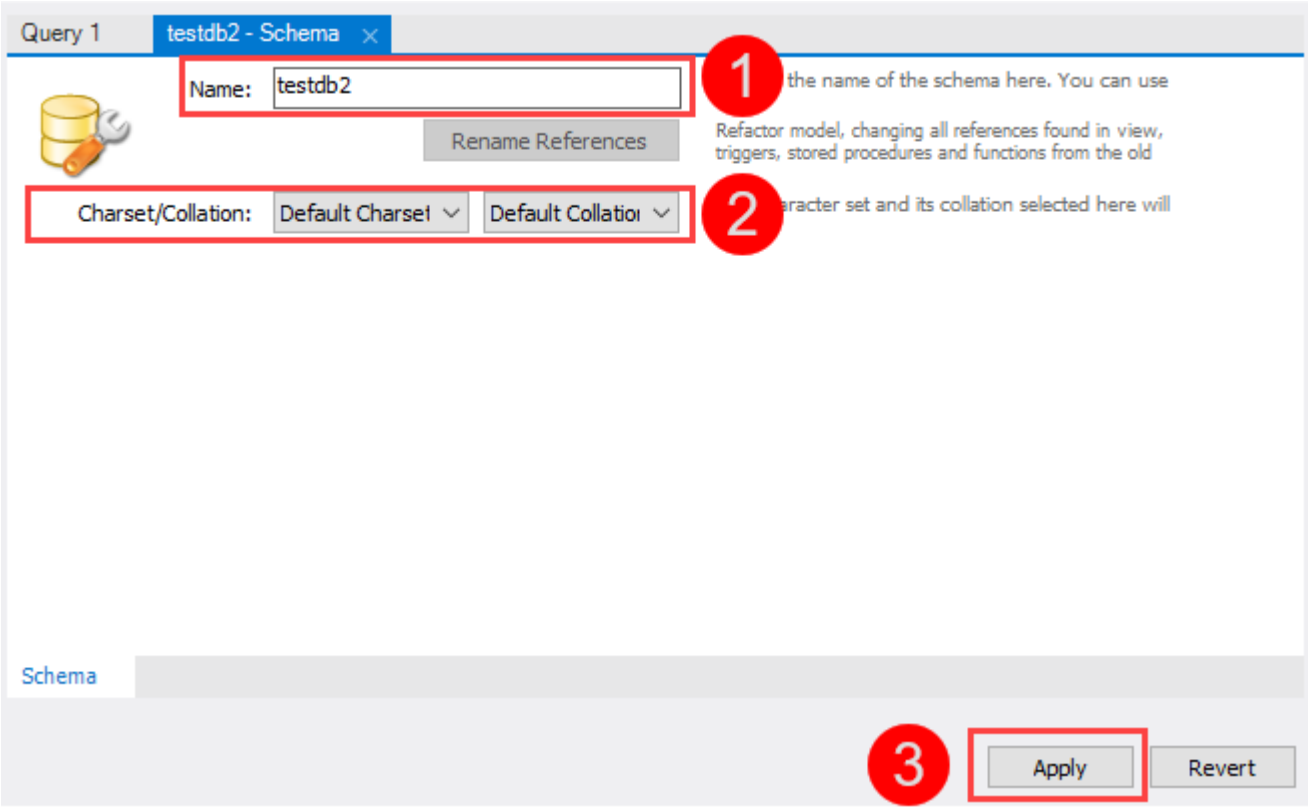



Fourth, click the **create a new schema in the connected server** button from the toolbar:

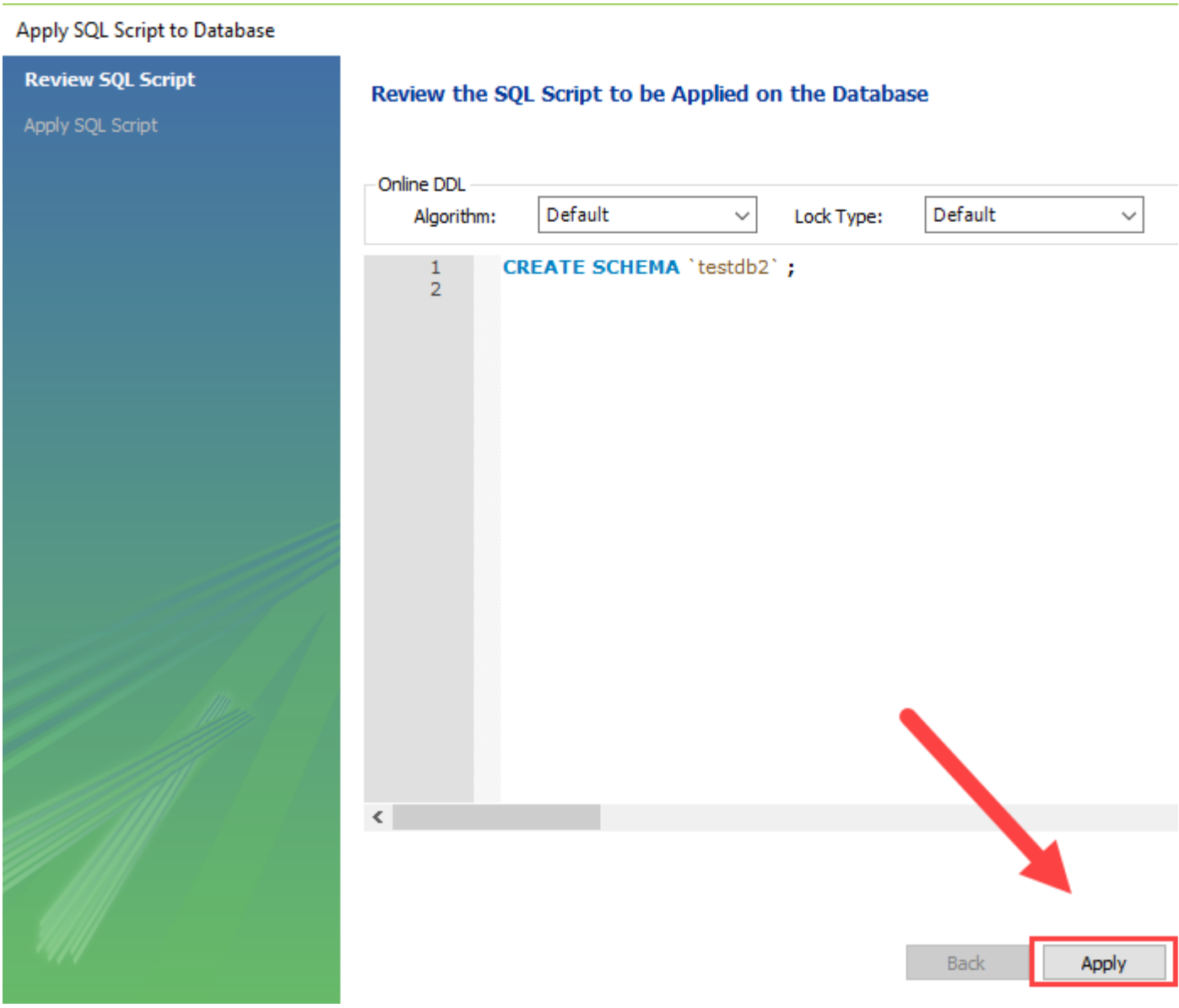



In MySQL, the schema is the synonym for the database. Creating a new schema also means creating a new database.

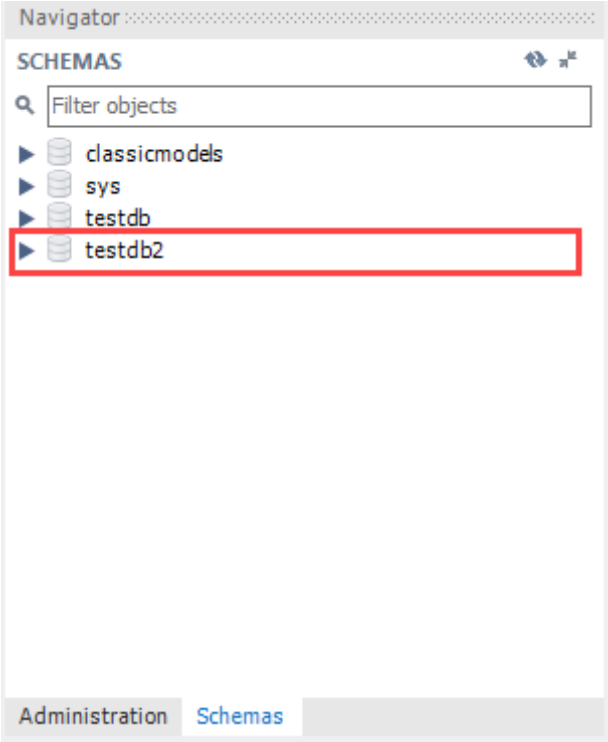
Fifth, the following window is open. You need to (1) enter the schema name, (2) change the character set and collation if necessary, and click the **Apply** button:



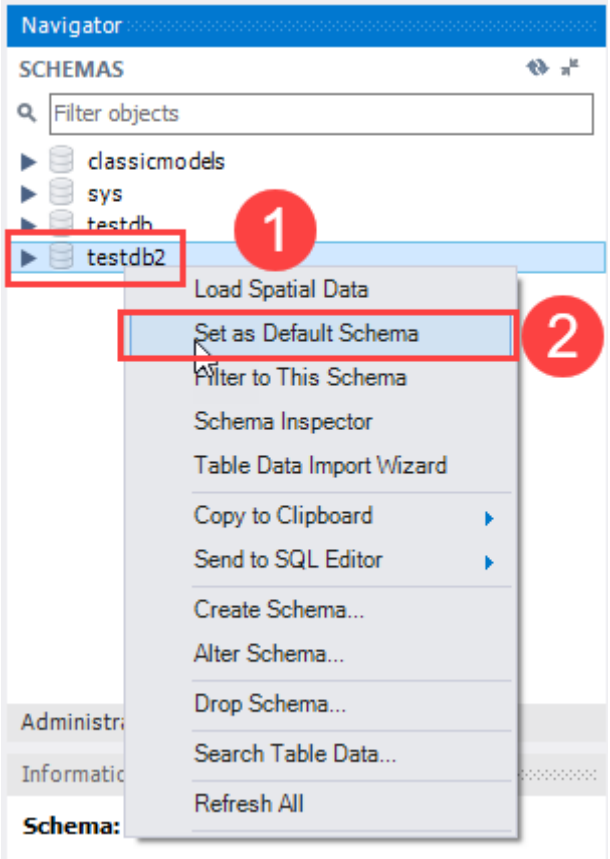

Sixth, MySQL Workbench opens the following window that displays the SQL script which will be executed. Note that the CREATE SCHEMA statement command has the same effect as the CREATE DATABASE statement.



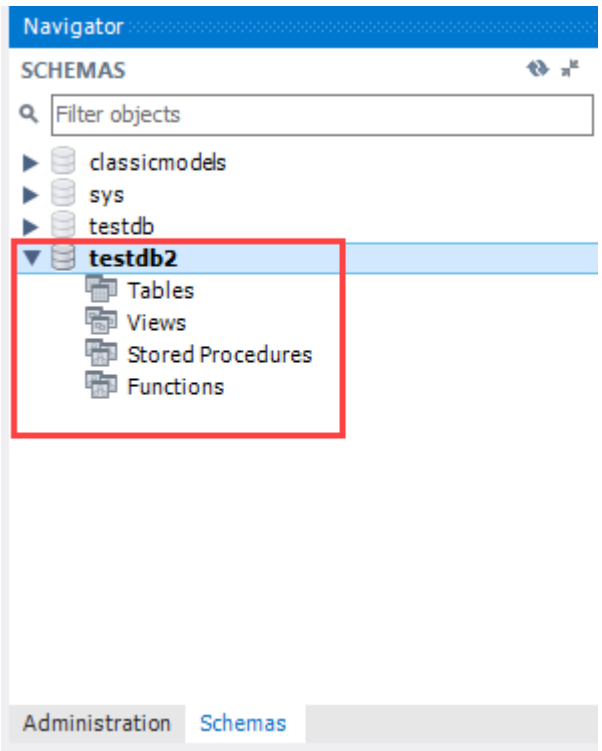

If everything is fine, you will see the new database created and showed in the **schemas** tab of the **Navigator** section.




Seventh, to select the testdb2 database, (1) right click the database name and (2) choose **Set as Default Schema** menu item:




The testdb2 node is open as shown in the following screenshot.




DDL

Data Definition Language (DDL) statements are used to define the database structure or schema. Data Definition Language describes how the data should exist 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. DDL are used to CREATE, ALTER, OR DROP the database objects (Table, Views, Users).

Data Definition Language (DDL) is used in different statements :

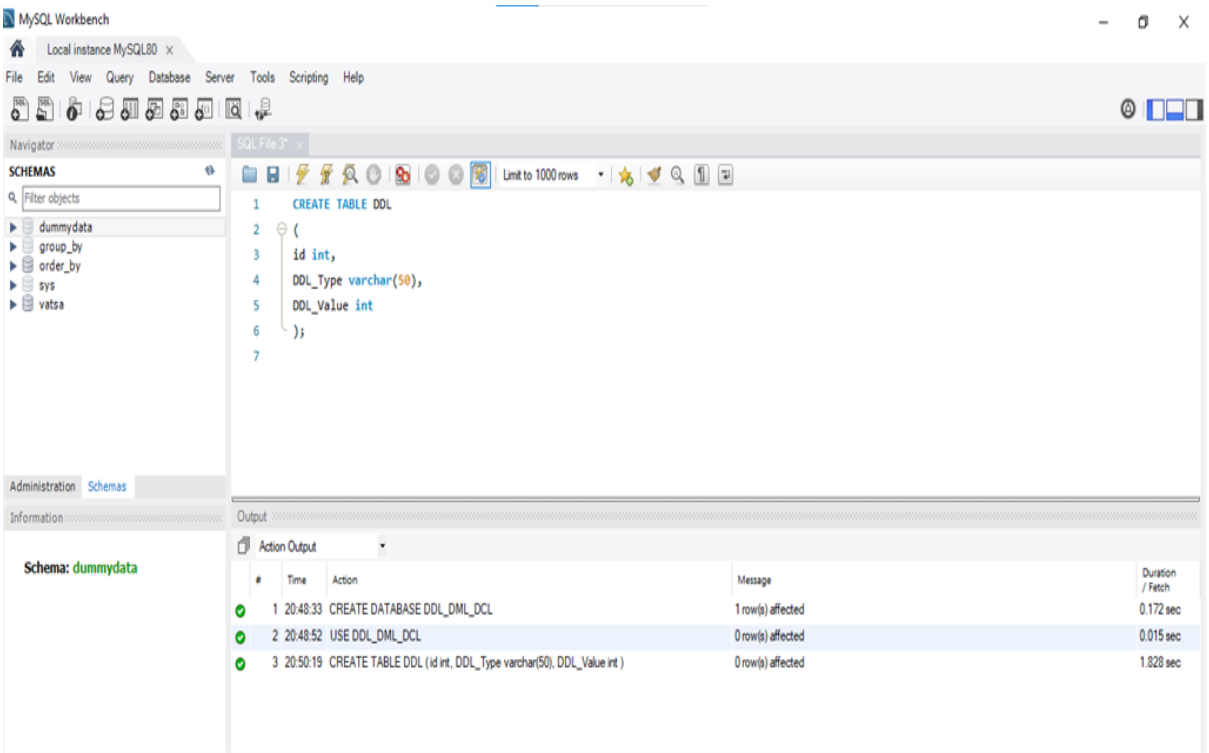
- CREATE - to create objects in the database
- ALTER - alters the structure of the database
- DROP - delete objects from the database
- TRUNCATE - remove all records from a table, including all spaces allocated for the records are removed
- COMMENT - add comments to the data dictionary
- RENAME - rename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

A) CREATE TABLE

**Syntax:**
CREATE TABLE table_name(
Col_name1 datatype(),
Col_name2 datatype(),…
Col_namen datatype(),
);

**Example:** Here, we are creating a sample table.

```
1. CREATE TABLE DDL
2. (
3.     id int,
4.     DDL_Type varchar(50),
5.     DDL_Value int
6. );
```



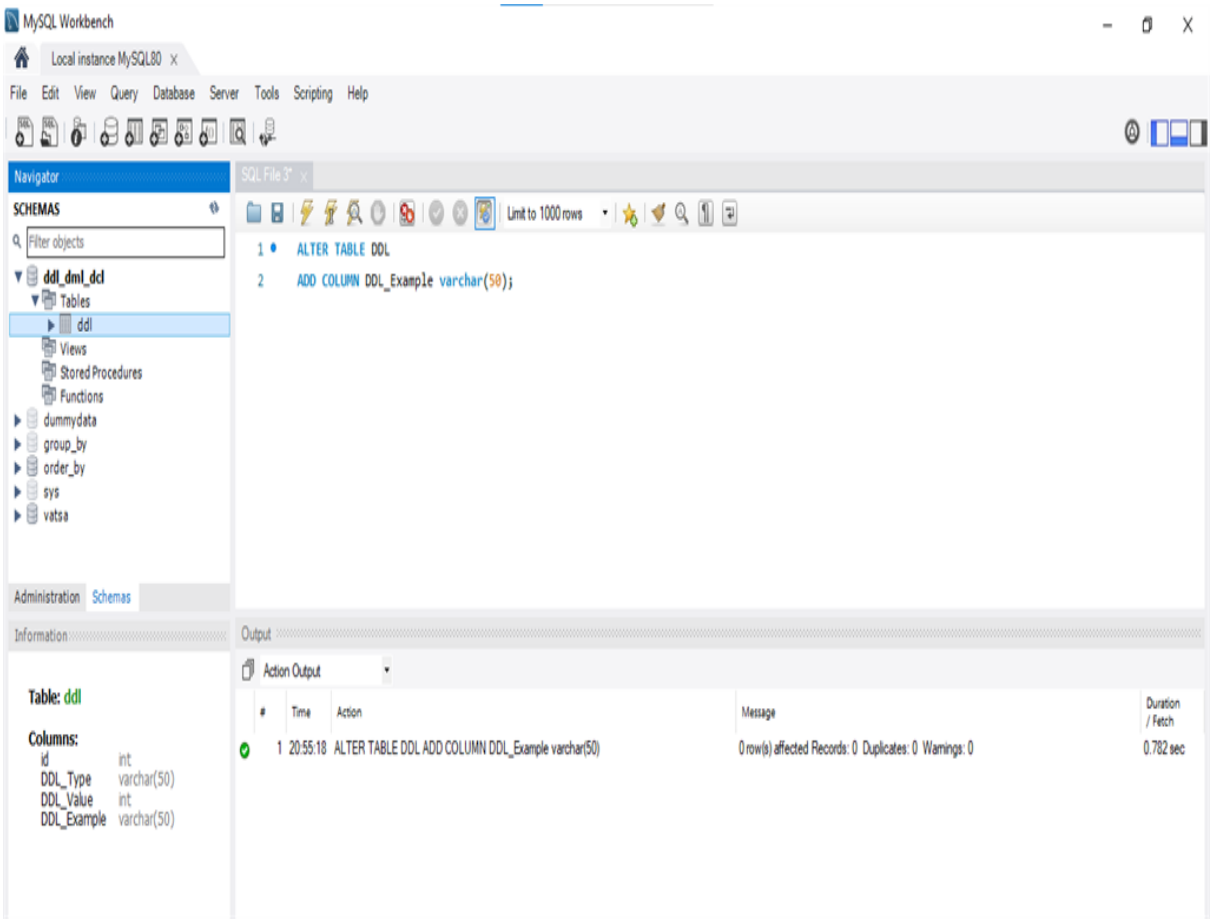

B) ALTER TABLE

1) ADD

**Syntax**:
ALTER TABLE table_name
ADD Col_name datatype()...;

**Example:** Here, we are adding a new column to the existing table.

```
1. ALTER TABLE DDL
2. ADD COLUMN DDL_Example varchar(50);
```

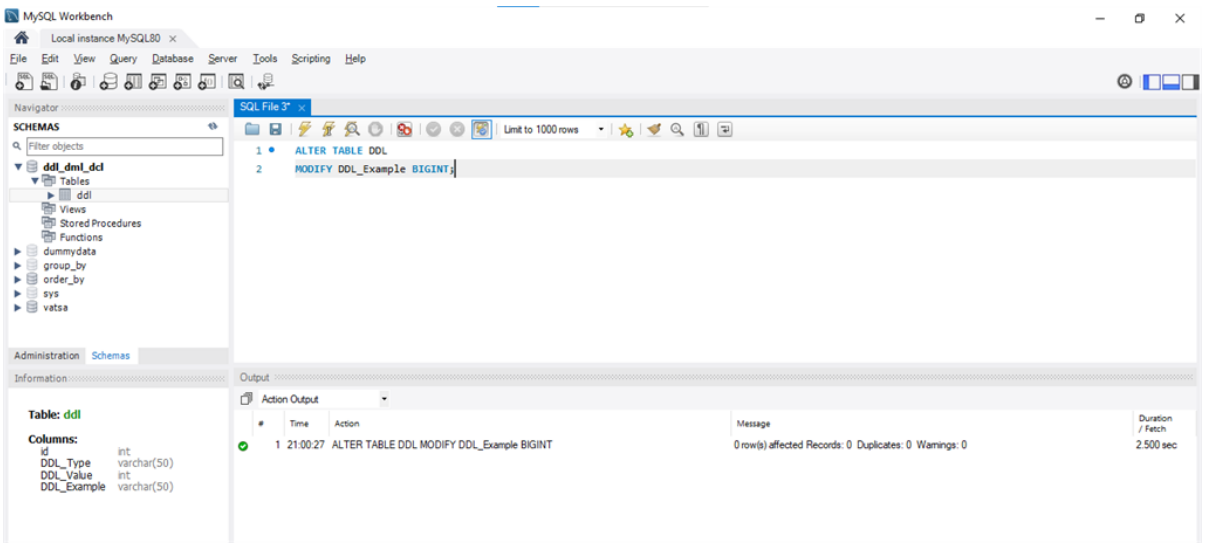



2) MODIFY

**Syntax:**
ALTER TABLE table_name
MODIFY (fieldname datatype()...);

**Example:** Modify a datatype in an existing table.

```
1. ALTER TABLE DDL
2. MODIFY DDL_Example BIGINT;
```

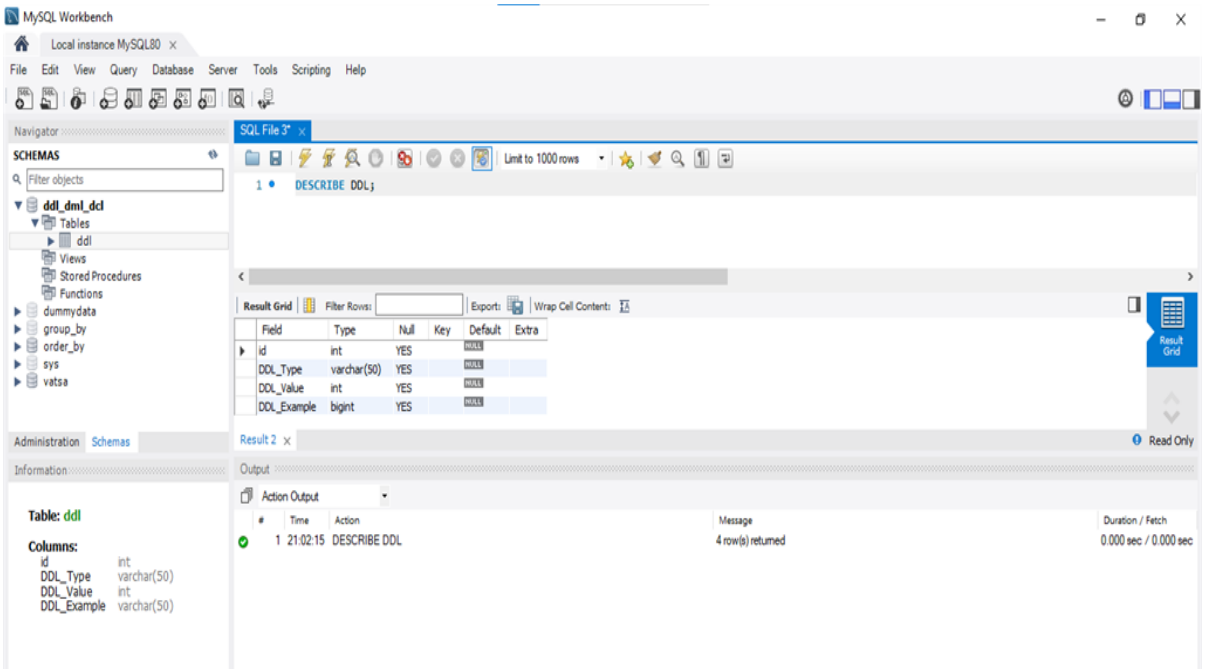



C) DESCRIBE TABLE

**Syntax:**
DESCRIBE TABLE NAME;

**Example:** This query is used to view the table.

1. DESCRIBE DDL;



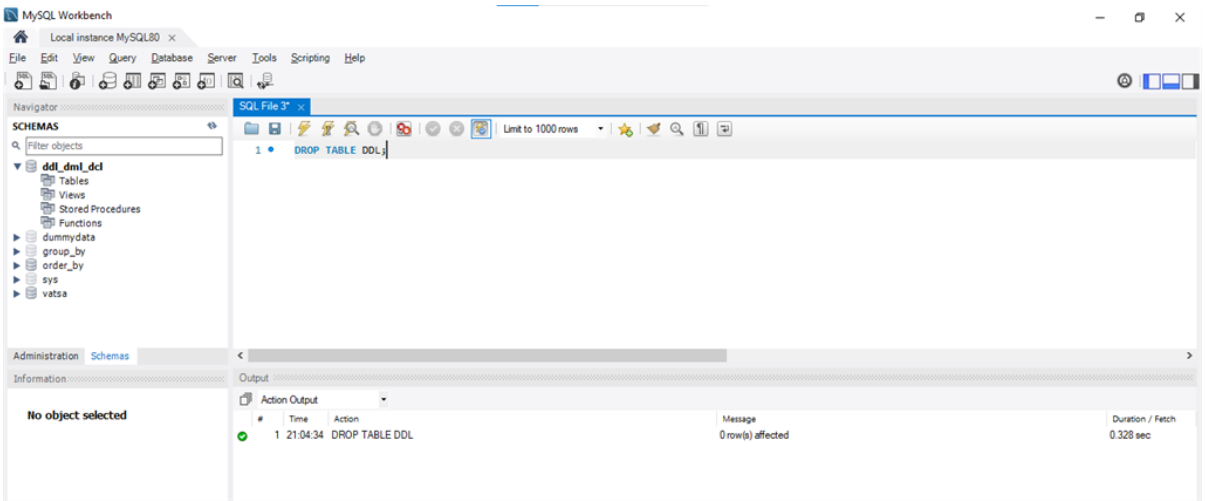

D) DROP TABLE

**Syntax:**

DROP Table name;

**Example:** Used to drop a table.

1. **DROP TABLE** DDL;




E) COMMENT

Add comments to the data dictionary

```
      # get subordinates of Diane

      OR
/*
    Get sales rep employees
    that reports to Anthony
*/
```

F) RENAME

Rename a table

**Syntax:**
**ALTER TABLE tablename**
RENAME  TO new table_name

Eg: ALTER TABLE STUDENT
      RENAME TO student;

Rename a column
Syntrax
**ALTER TABLE tablename**
RENAME  COLUMN oldcolumn_name TO new column_name;

Example
ALTER TABLE SPECIALISTDOCTOR
RENAME COLUMN DID TO DOCTORID;

F) TRUNCATE

It will remove all the tuples(CONTENTS ) from the table , but the table exists
Syntax:
TRUNCATE [TABLE] tbl_name

EXMAPLE

TRUNCATE TABLE SPECIALISTDOCTOR;

DML

Data Manipulation Language (DML) statements are used for managing data within schema objects DML deals with data manipulation, and therefore includes most common SQL statements such as SELECT, INSERT, etc. DML allows adding / modifying / deleting data itself.

DML is used to manipulate the existing data in the database objects (insert, select, update, delete).

**DML Commands**

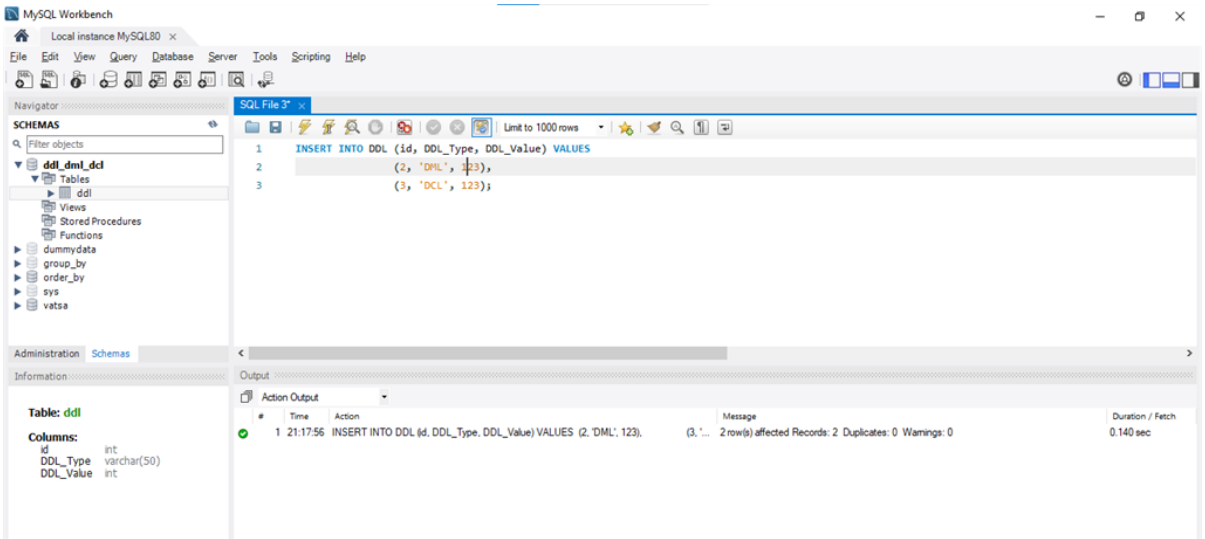
1.INSERT
2.SELECT
3.UPDATE
4.DELETE

1) INSERT

**Syntax:**
INSERT INTO Table_Name VALUES();

**Example:** Here, we are going to insert some values.

```
INSERT INTO DDL (id, DDL_Type, DDL_Value) VALUES
(2, 'DML', 123),
(3, 'DCL', 123);
```

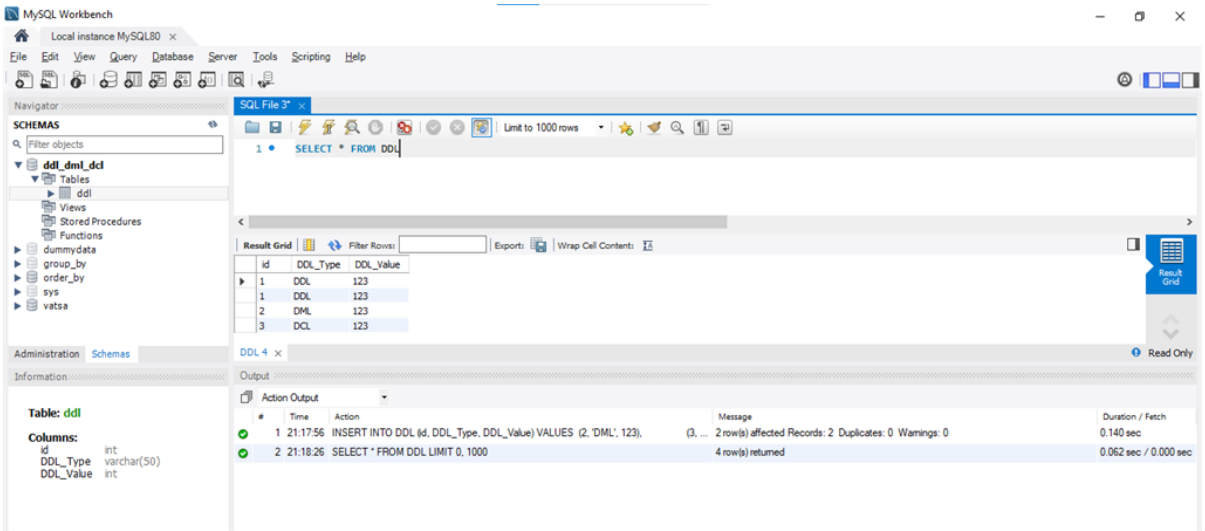



2) SELECT

**Syntax:**
SELECT * FROM <table_name>

**Example:** select query is used to fetch the data from tables.

```
SELECT * FROM DDL
```



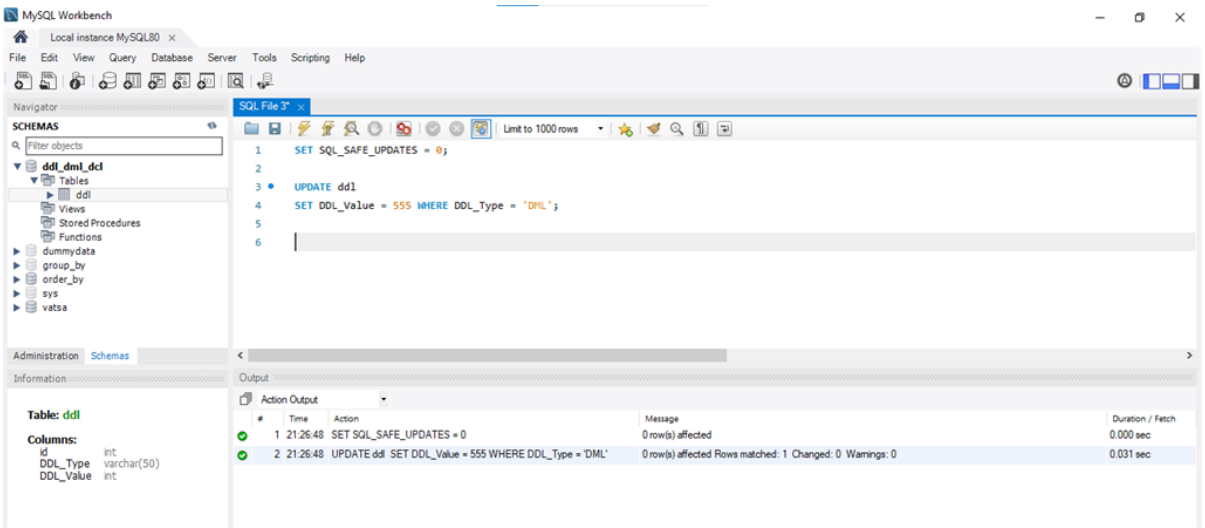

3) UPDATE

**Syntax:**

UPDATE <table name> set to(calculation);

**Example:** Update command is used to update any value from any table.

```
1. UPDATE ddl
2. SET DDL_Value = 555 WHERE DDL_Type = 'DML';
```

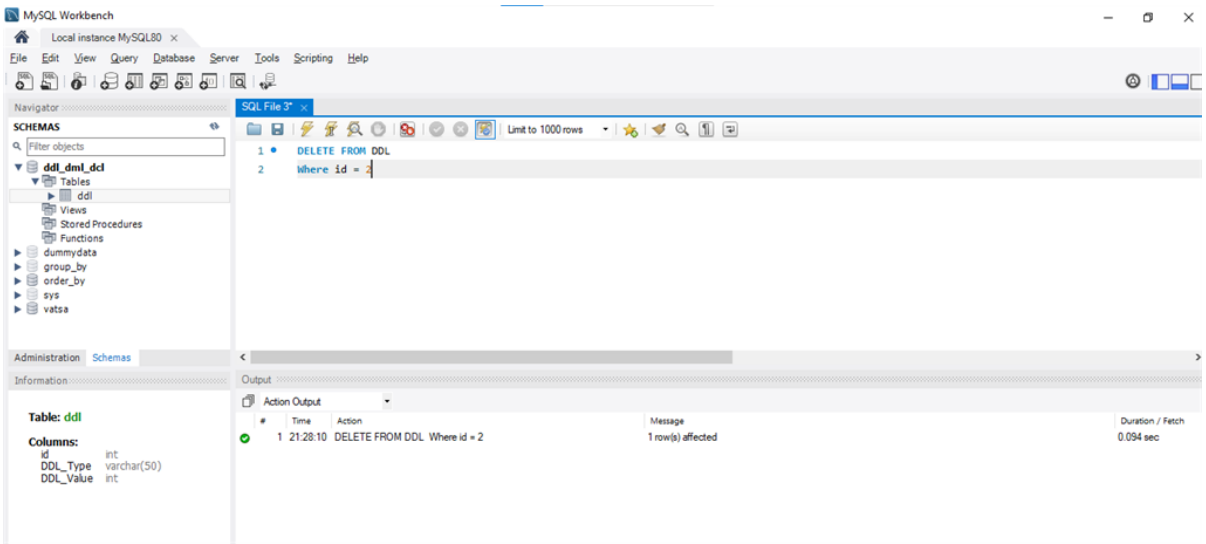



4) DELETE

**Syntax:**
DELETE FROM <table_name>

**Example:** Delete query is used to delete a row from a table.

```
1. DELETE FROM DDL
2. Where id = 2
```



DCL

DCL is the abstract of Data Control Language. Data Control Language includes commands such as GRANT, and is concerned with rights, permissions, and other controls of the database system. DCL is used to grant/revoke permissions on databases and their contents. DCL is simple, but MySQL permissions are a bit complex. DCL is about security. DCL is used to control the database transaction. DCL statements allow you to control who has access to a specific object in your database.
1. GRANT
2. REVOKE

GRANT

It provides the user's access privileges to the database. The MySQL database offers both the administrator and user a great extent of the control options. The administration side of the process includes the possibility for the administrators to control certain user privileges over the MySQL server by restricting their access to an entire database or usage limiting permissions for a specific table. It creates an entry in the security system that allows a user in the current database to work with data in the current database or execute specific statements.

**Syntax :**
Statement permissions:
GRANT { ALL | statement [ ,...n ] }
TO security_account [ ,...n ]

Normally, a database administrator first uses CREATE USER to create an account, then GRANT to define its privileges and characteristics.

**For example:**

```
1. CREATE USER vatsa@'localhost' IDENTIFIED BY 'mypass';
2. GRANT ALL ON MY_TABLE TO vatsa@'localhost';
3. GRANT SELECT ON Users TO vatsa@'localhost';
```

REVOKE

The REVOKE statement enables system administrators and to revoke (back permission) the privileges from MySQL accounts.

**Syntax:**

REVOKE
priv_type [(column_list)]
[, priv_type [(column_list)]] ...
ON [object_type] priv_level
FROM user [, user] ...
REVOKE ALL PRIVILEGES, GRANT OPTION
FROM user [, user] ...

**For example:**

```
1. REVOKE INSERT ON *.* FROM 'vatsa'@'localhost';
```