LAB# 3

MySQL CREATE DATABASE

MySQL implements a database as a directory that contains all files

which correspond to tables in the database.

To create a new database in MySQL, you use the CREATE DATABASE

statement with the following syntax:

```
CREATE DATABASE [IF NOT EXISTS] database_name

[CARA
CTER
SET
charse
t_nam
e]
[COLL
ATE
collati
on_na
me]
```

to display the existing database in the server to make sure that you are not creating a new database that already exists, you use the SHOW DATABASES command as follows:

```
MySQL> SHOW DATABASES;
```

Then, issue the CREATE DATABASE command with the database e.g., testdb and press Enter:

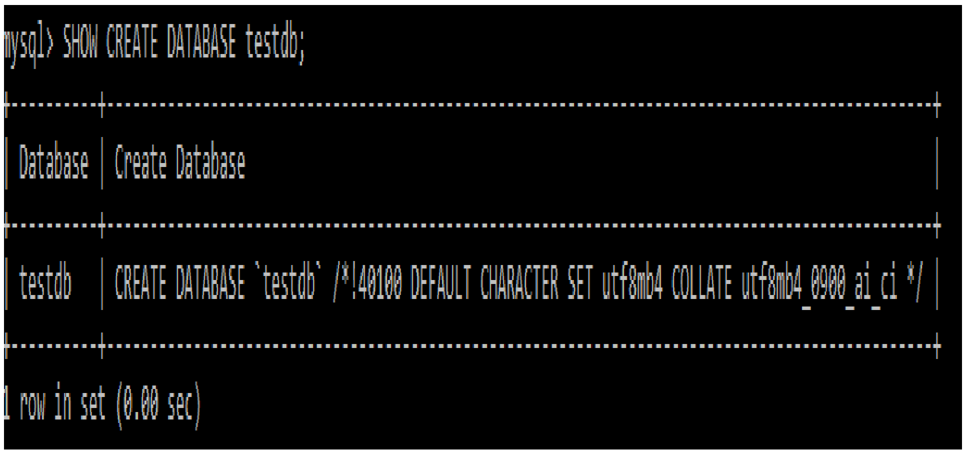
```
MySQL> CREATE DATABASE testdB;

Query OK, 1 row affected (0.12 sec)
```

After that, if you want to review the created database, you can use the SHOW CREATE

DATABASE command:

MySQL> SHOW CREATE DATABASE testdb;



```
mysql> SHOW CREATE DATABASE testdb;
+---------+----------------------------------------------------------------------------------------------------+
| Database | Create Database                                                                                  |
+---------+----------------------------------------------------------------------------------------------------+
| testdb  | CREATE DATABASE `testdb` /*!40100 DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci */ |
+---------+----------------------------------------------------------------------------------------------------+
1 row in set (0.00 sec)
```

MySQL returns the database name and the character set and collation of the database.

Finally, to access the newly created database, you use the USE database command as follows:MySQL> USE testdb;
Database changed

How to Create Table in MySQL using SQL create command

Tables can be created using CREATE TABLE statement and it actually has the following syntax.

CREATE TABLE [IF NOT EXISTS] `TableName` (`fieldname` dataType [optional parameters]) ENGINE = storage Engine;

HERE

- "CREATE TABLE" is the one responsible for the creation of the table in the database.

- "[IF NOT EXISTS]" is optional and only create the table if no matching

  table name is found.

- "`fieldName`" is the name of the field and "data Type" defines the nature of the data to be stored in the field.
- "[optional parameters]" additional information about a field such as

" AUTO_INCREMENT" , NOT NULL etc.

Now, we will create the following table in the TUTORIALS database.

Example
Here is an example, which will create tutorials_tbl root@host# mysql -u root -p

```
Enter password:*******

mysql> use TUTORIALS;

Database changed

mysql> CREATE TABLE tutorials_tbl(

   -> tutorial_id INT NOT NULL AUTO_INCREMENT,

   -> tutorial_title VARCHAR(100) NOT NULL,

   -> tutorial_author VARCHAR(40) NOT NULL,

   -> submission_date DATE,

   -> PRIMARY KEY ( tutorial_id )

   -> );

Query OK, 0 rows affected (0.16 sec)

mysql>
```

Here, a few items need explanation −

Field Attribute NOT NULL is being used because we do not want this field to be NULL. So, if a user will try to create a record with a NULL value, then MySQL will raise an error. Field Attribute AUTO_INCREMENT tells MySQL to go ahead and add the next available number

to the id field. Keyword PRIMARY KEY is used to define a column as a primary key. You canuse multiple columns separated by a comma to define a primary key.

DATA TYPES

Data types define the nature of the data that can be stored in a particular column of a table

MySQL has 3 main categories of data types namely
Numeric, Text, Date/time.

MySQL - Insert Query

To insert data into a MySQL table, you would need to use the

SQL **INSERT INTO** command.

Syntax

Here is a generic SQL syntax of INSERT INTO command to insert data into

the MySQL table

```
INSERT INTO table_name ( field1, field2,...fieldN )
   VALUES
   ( value1, value2,...valueN );
```

To insert string data types, it is required to keep all the values

into double or single quotes. For example "value".

Example

The following example will create 3 records into tutorials_tbl table –

```
mysql> INSERT INTO tutorials_tbl
```

```
->(tutorial_title, tutorial_author, submission_date)

    ->VALUES

    ->("Learn PHP", "John Poul", NOW());

Query OK, 1 row affected (0.01 sec)


mysql> INSERT INTO tutorials_tbl

    ->(tutorial_title, tutorial_author, submission_date)

    ->VALUES

    ->("Learn MySQL", "Abdul S", NOW());

Query OK, 1 row affected (0.01 sec)


mysql> INSERT INTO tutorials_tbl

    ->(tutorial_title, tutorial_author, submission_date)

    ->VALUES

    ->("JAVA Tutorial", "Sanjay", '2007-05-06');

Query OK, 1 row affected (0.01 sec
```

MySQL DELETE Query: How to Delete Row from a Table

MySQL DELETE command is used to delete rows that are no longer required from the database tables.

The Delete query in MySQL can delete more than one row from a table in a single query. This proves to be advantages when removing large numbers of rows from a database table.

Once a Delete row in MySQL row has been deleted, it cannot be recovered. It is therefore strongly recommended to make database backups before deleting any data from the database. This can allow you to restore the database and view the data later on should it be required.

How to Delete a row in MySQL

To delete a row in MySQL, the DELETE FROM statement is used: DELETE

FROM `table_name` [WHERE condition];

DELETE FROM `table_name` tells MySQL server to remove rows from the table

.. [WHERE condition] is optional and is used to put a filter that restricts the

number of rows affected by the MySQL DELETE row query.

If the WHERE clause is not used in the MySQL DELETE query, then all the rows in

a given table will be deleted.

Introduction to MySQL UPDATE statement

The UPDATE statement updates data in a table. It allows you to change the values in one or more columns of a single row or multiple rows.

The following illustrates the basic syntax of the UPDATE statement:

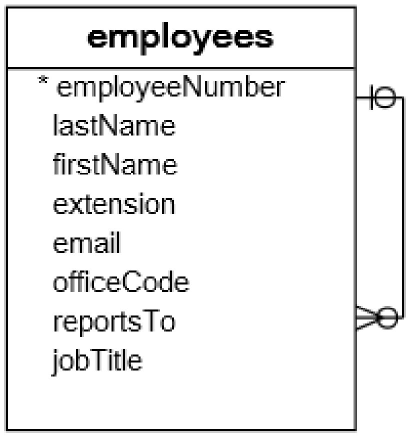
```
  UPDATE [LOW_PRIORITY] [IGNORE] table_nameSET

        column_name1 = expr1,

        column_name2 = expr2,.. [WHEREcondition];
```

Using MySQL UPDATE to modify values in a single column example

See the following employees table from the sample database.



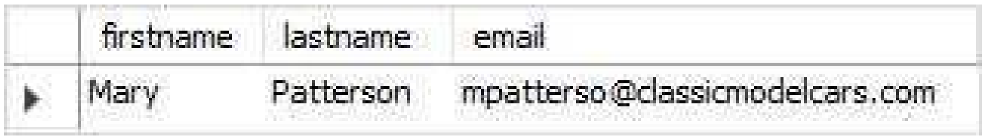
In this example, we will update the email of Mary Patterson to the new email mary.patterso@classicmodelcars.com.

First, find Mary’s email from the employees table using the following SELECT statement:

```
SELECTfirstname, lastname, emailFROMemployees

WHERE

    employeeNumber = 1056;
```



	firstname	lastname	email
▶	Mary	Patterson	mpatterso@classicmodelcars.com

Second, update the email address of Mary to the new email mary.patterson@classicmodelcars.com :UPDATE employees SET

```
    email = 'mary.patterson@classicmodelcars.com' WHERE
    employeeNumber = 1056;
```

MySQL issued the number of rows affected:

```
1 row(s) affected
```