NCTU ECE

Introduction to MySQL with QT

Advanced Object Oriented Programming

Alphar Juan

2019. 11. 20/21

Introduction to Database

Database Management System (DBMS) and SQL

INTRODUCTION TO DATABASE

- A database is an organized collection of data, generally stored and accessed electronically from a computer system. Where databases ar e more complex they are often developed using formal design and modeling techniques.
- A relational database organizes data in tables. A table has rows (or records) and columns (or fields). Tables are related based on common columns to eliminate data redundancy and ensure data integrity.
- The database management system (DBMS) is the software that interacts with end users, applications, and the database itself to capture and analyze the data. The DBMS software additionally encompasses the core facilities provided to administer the database. The sum total of the database, the DBMS and the associated applications can be referred to as a "database system". Often the term "database" is also used to loosely refer to any of the DBMS, the database system or an application associated with the database.
- Popular Relationship Database Management System (RDBMS) includes the commercial Oracle, IBM DB2, Microsoft SQL Server and Access, SAP SyBase Teradata; and the open-source MySQL, PostgreSQL, Embedded Apache Derby (Java DB), mSQL (mini-SQL), SQLite and Apache OpenOffice's Base.
- A relational database system organizes data in the following hierarchy: A relational database system contains many *databases*. A database comprise s *tables*. A table have *rows* (or *records*) and *columns* (or *fields*).

Code/User End **DBMS** **Database**

INTRODUCTION TO SQL

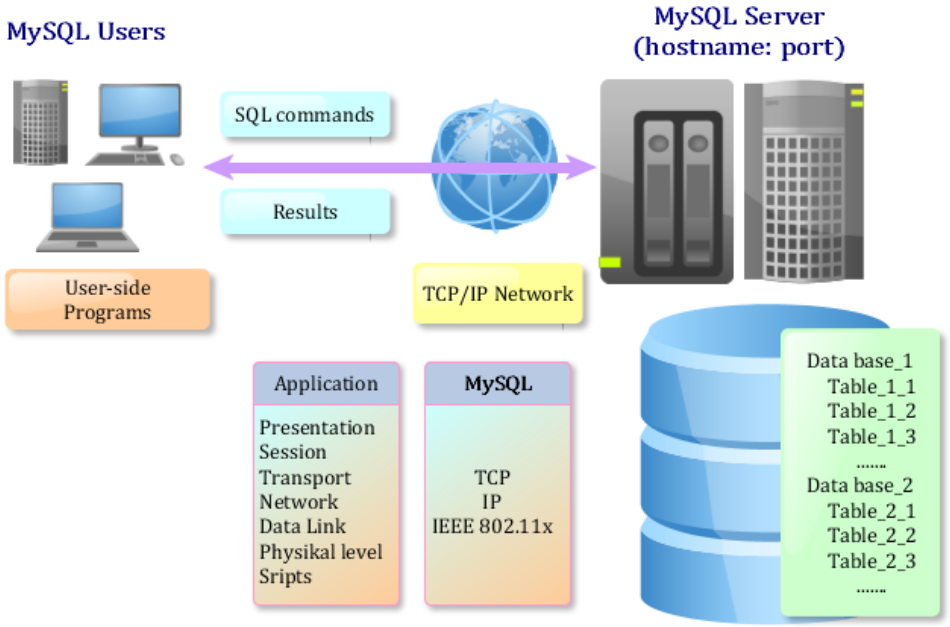
- A high-level programming language, called Structure Query Language (SQL), is designed for interacting with the relational databases. SQL defines a set of commands, such as SELECT, INSERT, UPDATE, DELETE, CREATE TABLE, DROP TABLE, and etc.

- Edgar F. Codd of IBM proposed the Relational Database Model in 1970. SQL, one of the earlier programming language, was subsequently developed by Donald D. Chamberlin and Raymond F. Boyce at IBM in the early 1970s. Oracle, subsequently, took it to a new height.

- ANSI (American National Standard Institute) established the first SQL standard in 1986 (SQL-86 or SQL-87) - adopted by ISO/IEC as "ISO/IEC 9075" - followed in 1989 (SQL-89), 1992 (SQL-92 or SQL2), 1999 (SQL-99 or SQL3), 2003 (SQL:2003), 2006 (SQL:2006), 2011 (SQL:2011) and 2016 (SQL:2016). However, most of the database vendors have their own directs, e.g., PL/SQL (Oracle), Transact-SQL (Microsoft, SAP), PL/pgSQL (PostgreSQL).

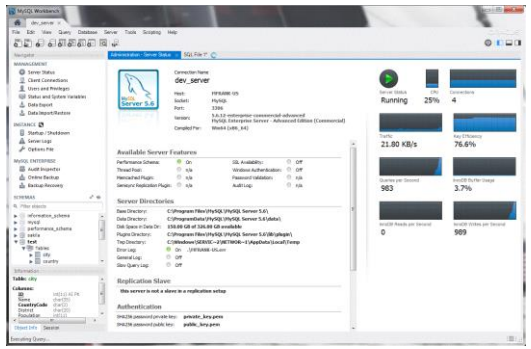
INTRODUCTION TO MySQL RDBMS

- SQL is a *programming language* for interacting with relational databases. On the other hand, MySQL is a *software system* - a Relational Database Management System.
- MySQL is one of the most used, *industrial-strength*, *open-source* and *free* Relational Database Management System (RDBMS). MySQL was developed by Michael "Monty" Widenius and David Axmark in 1995. It was owned by a Swedish company called MySQL AB, which was bought over by Sun Microsystems in 2008. Sun Microsystems was acquired by Oracle in 2010.
- MySQL is successful, not only because it is free and open-source (there are many free and open-source databases, such as PostgreSQL, Apache Derby (Java DB), mSQL (mini SQL), SQLite and Apache OpenOffice's Base), but also for its speed, ease of use, reliability, performance, connectivity (full networking support), portability (run on most OSes, such as Unix, Windows, Mac), security (SSL support), small size, and rich features. MySQL supports all features expected in a high-performance relational database, such as transactions, foreign key, replication, subqueries, stored procedures, views and triggers.
- MySQL is often deployed in a LAMP (Linux-Apache-MySQL-PHP), WAMP (Windows-Apache-MySQL-PHP), or MAMP (MacOS-Apache-MySQL-PHP) environment. All components in LAMP is free and open-source, inclusive of the Operating System.
- The mother site for MySQL is https://www.mysql.com. The ultimate reference for MySQL is the "MySQL Reference Manual", available at https://dev.mysql.com/doc. The reference manual is huge - the PDF has over 3700 pages!!!

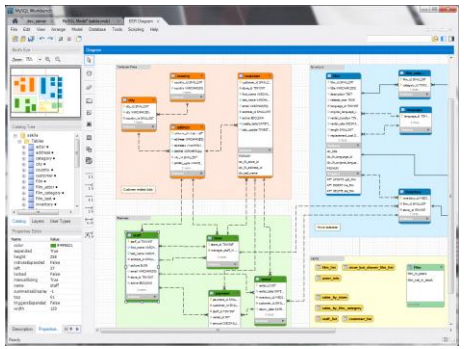


INTRODUCTION TO MySQL WORKBENCH

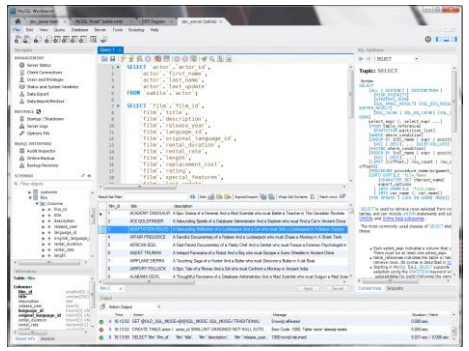
MySQL Workbench is a unified visual tool for database architects, developers, and DBAs. MySQL Workbench provides data modeling, SQL development, and comprehensive administration tools for server configuration, user administration, backup, and much more. MySQL Workbench is available on Windows, Linux and Mac OS X.



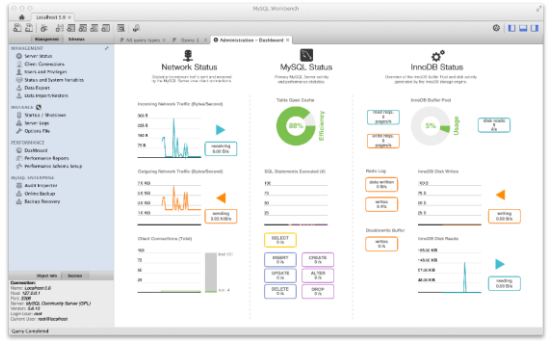
MySQL Workbench Home



Visual Database Design



SQL Editor



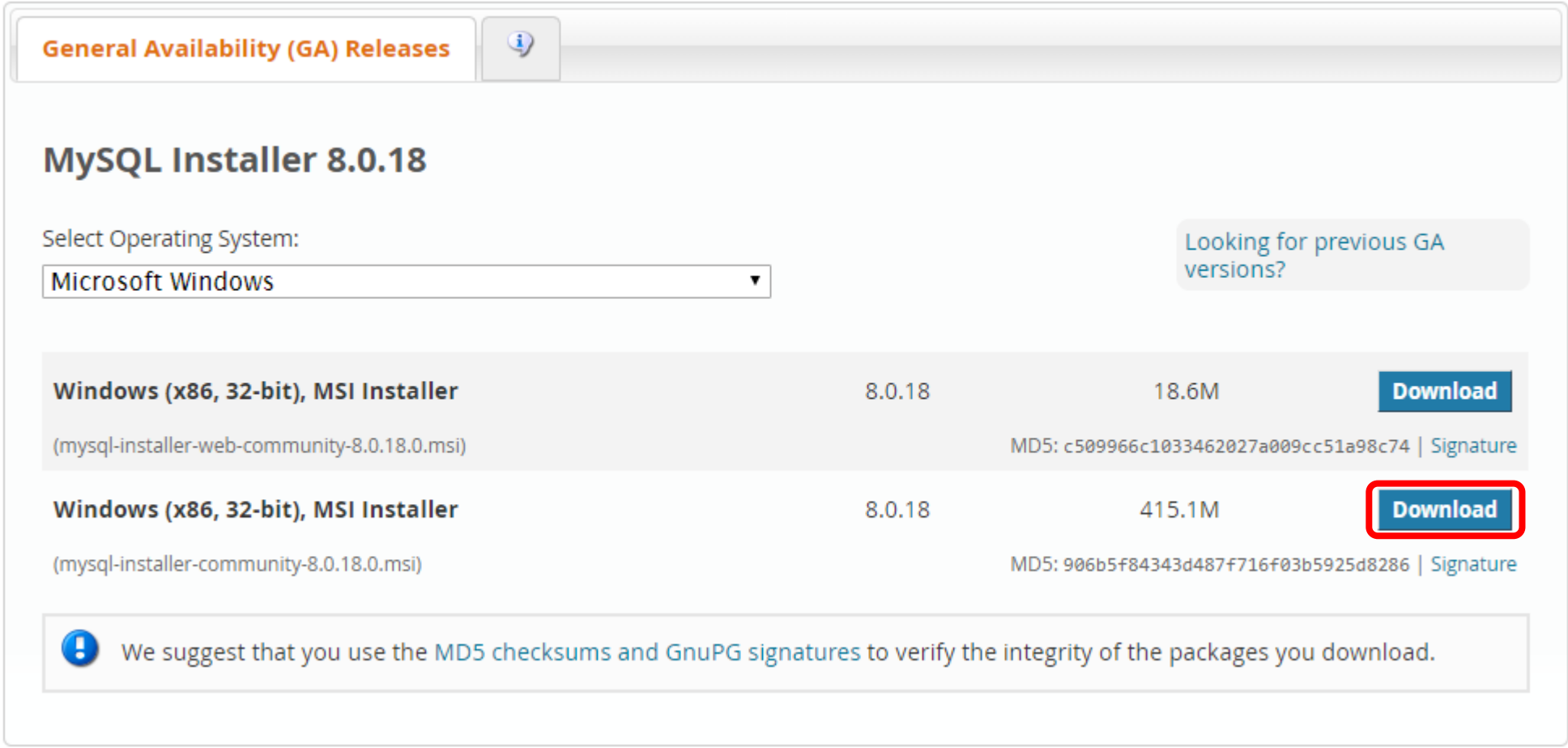
Performance Dashboard

Installation

MySQL Server and MySQL WorkBench

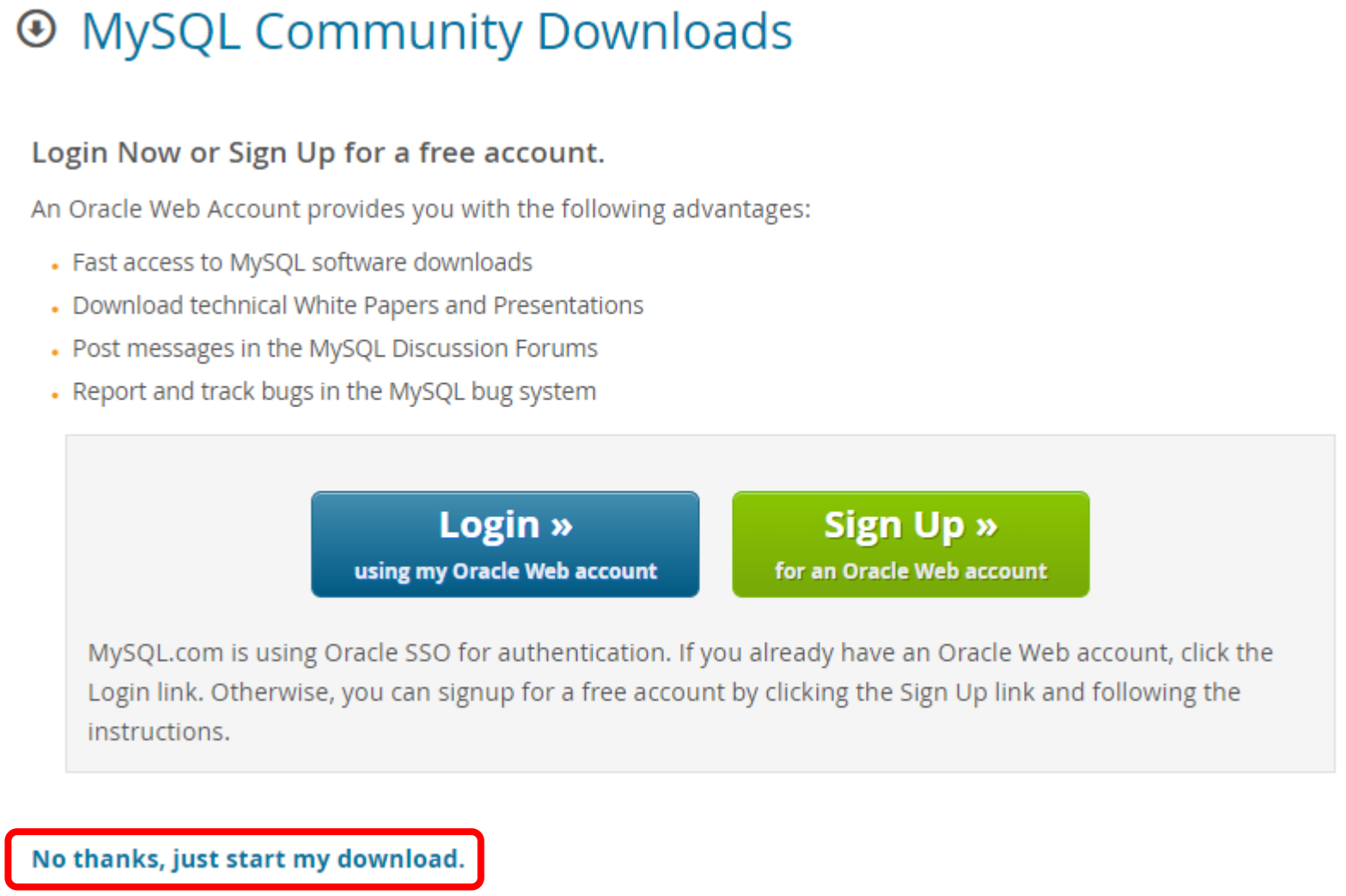
INSTALLATION

Go to official website to download complete version: https://dev.mysql.com/downloads/installer/



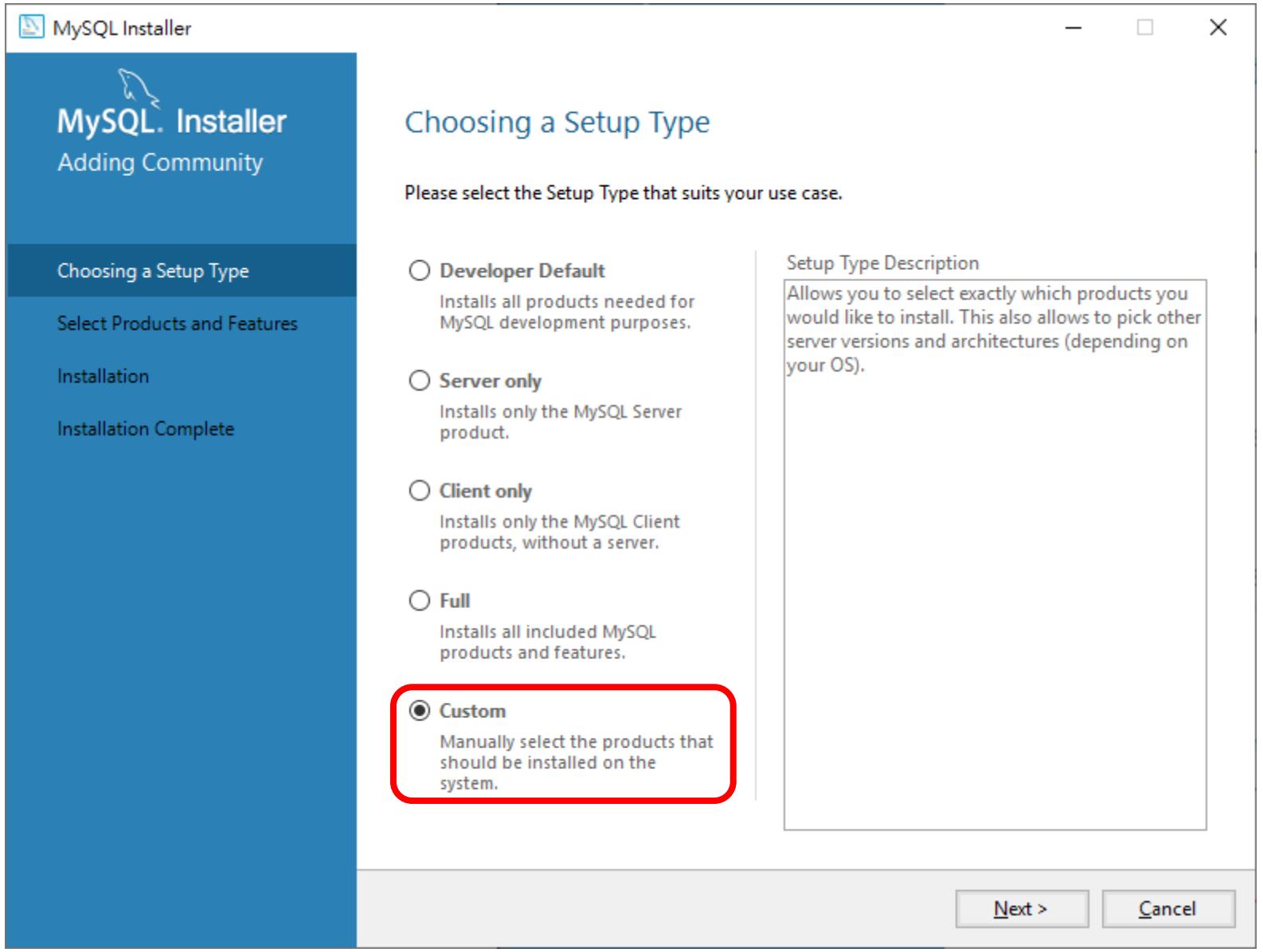
INSTALLATION

Skip login and sign up, select “No thanks, just start my download.”



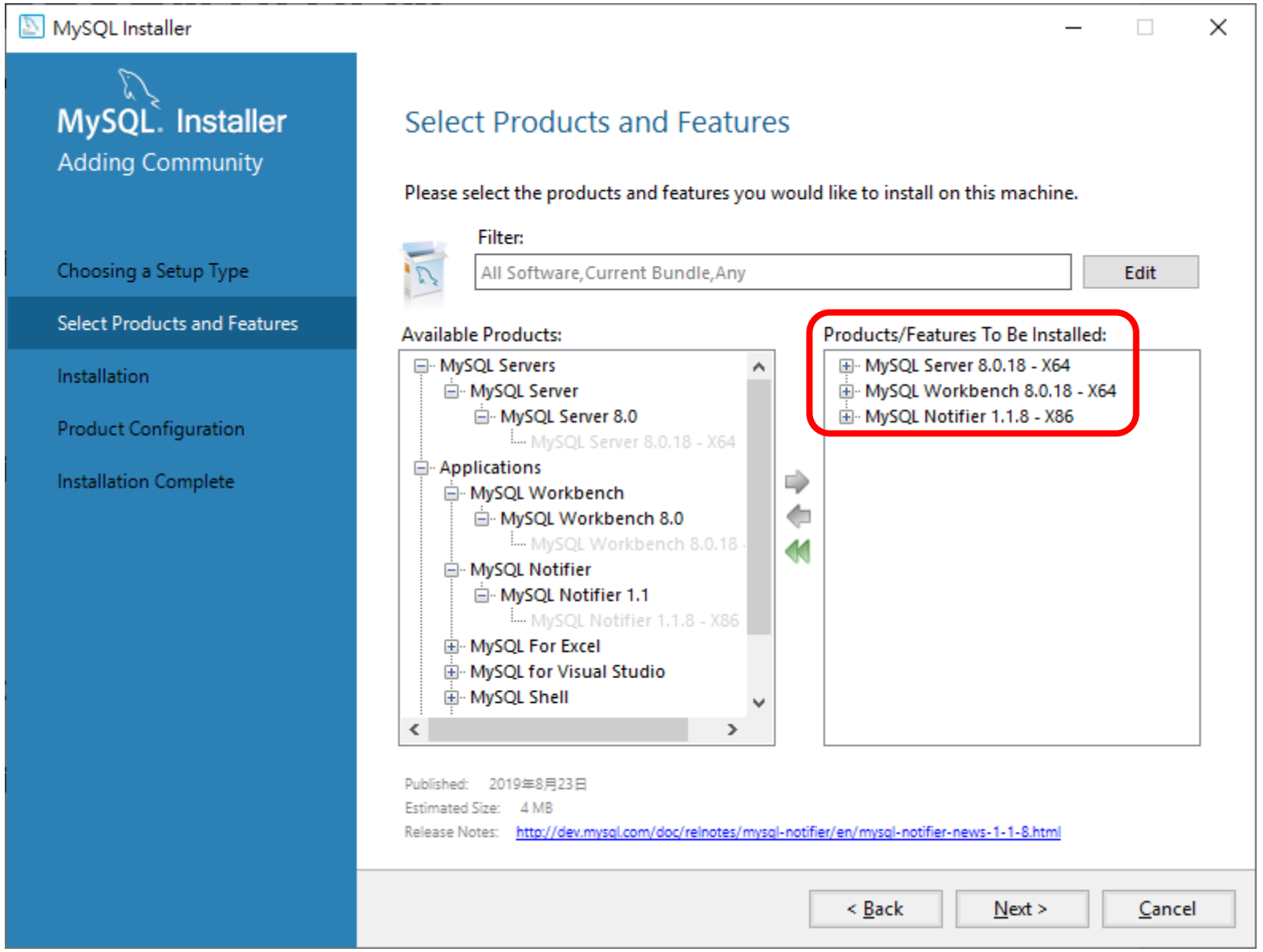
INSTALLATION

Select “Custom”.



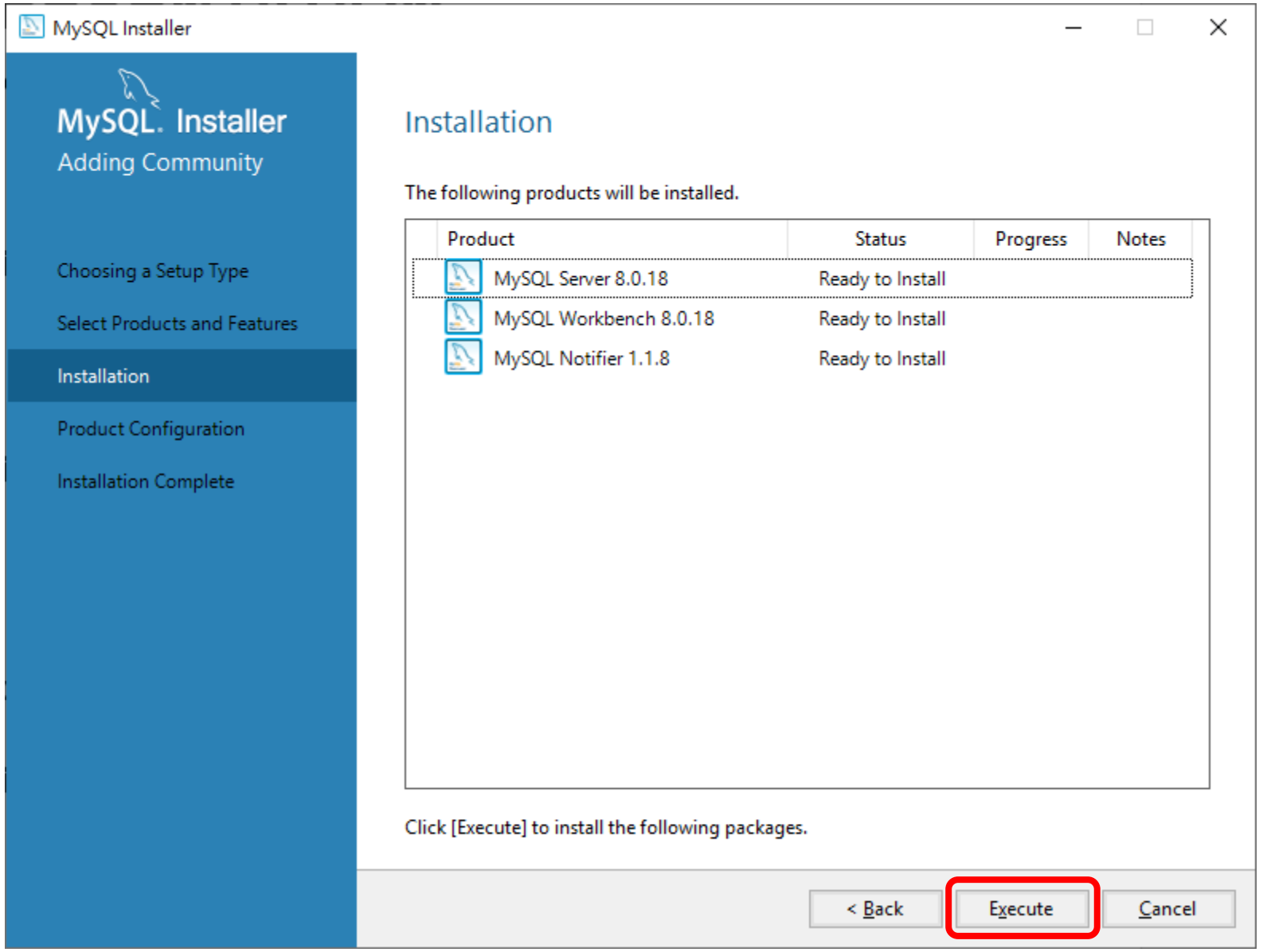
INSTALLATION

Select “MySQL Server”, “MySQL Workbench”, and “MySQL Notifier”.



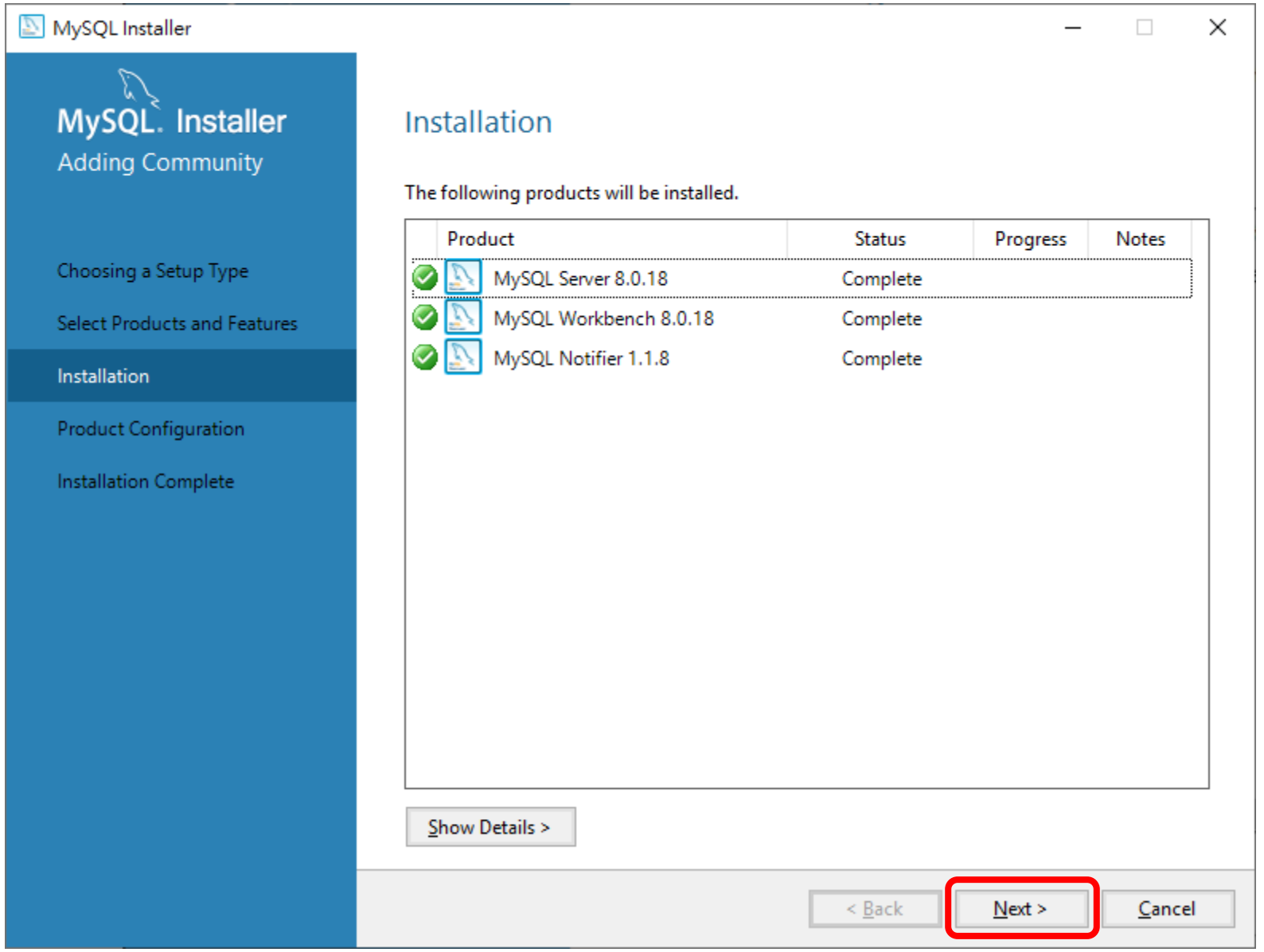
INSTALLATION

Select “Execute” ton install the softwares.



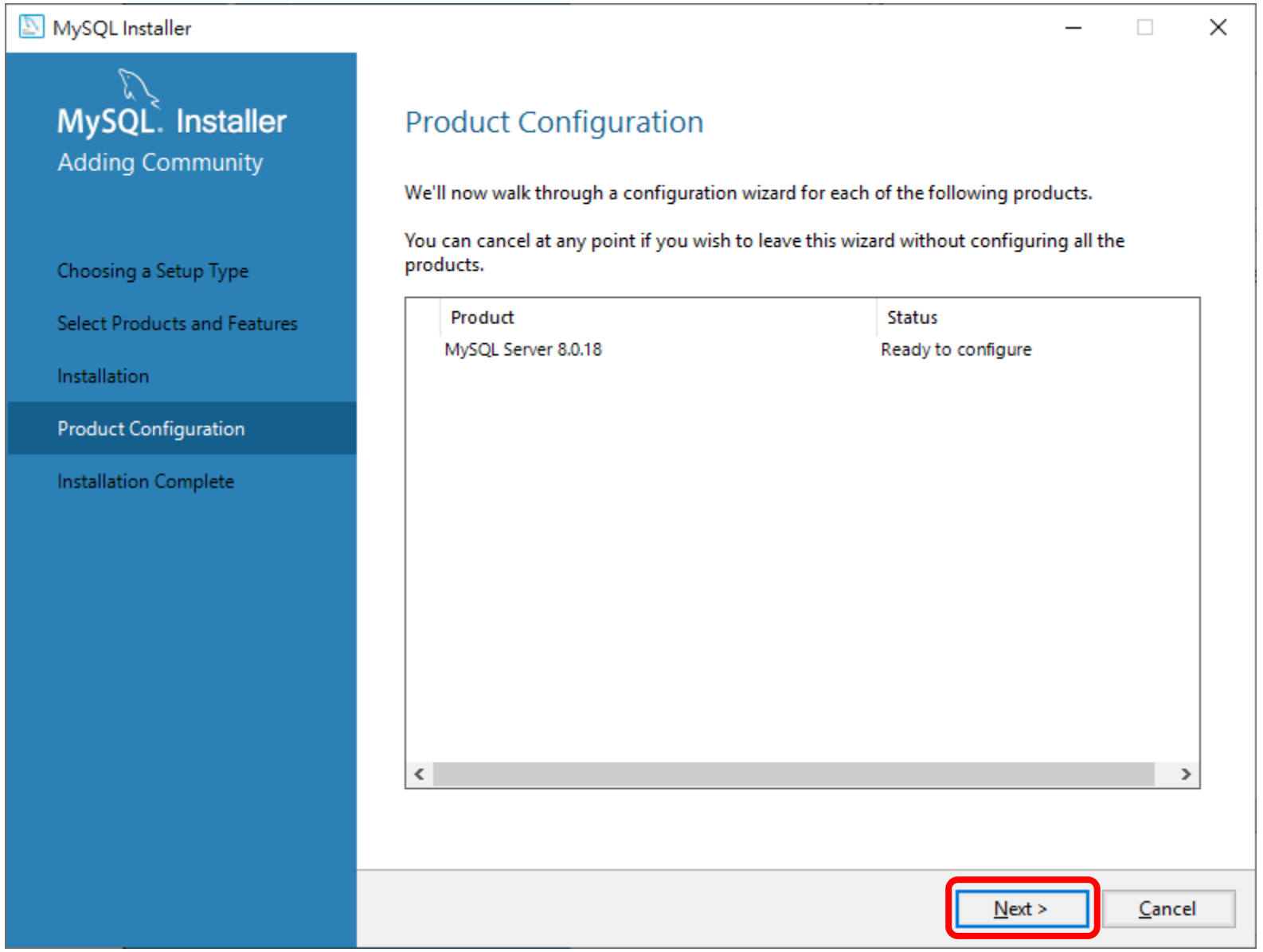
INSTALLATION

Select “Next”.



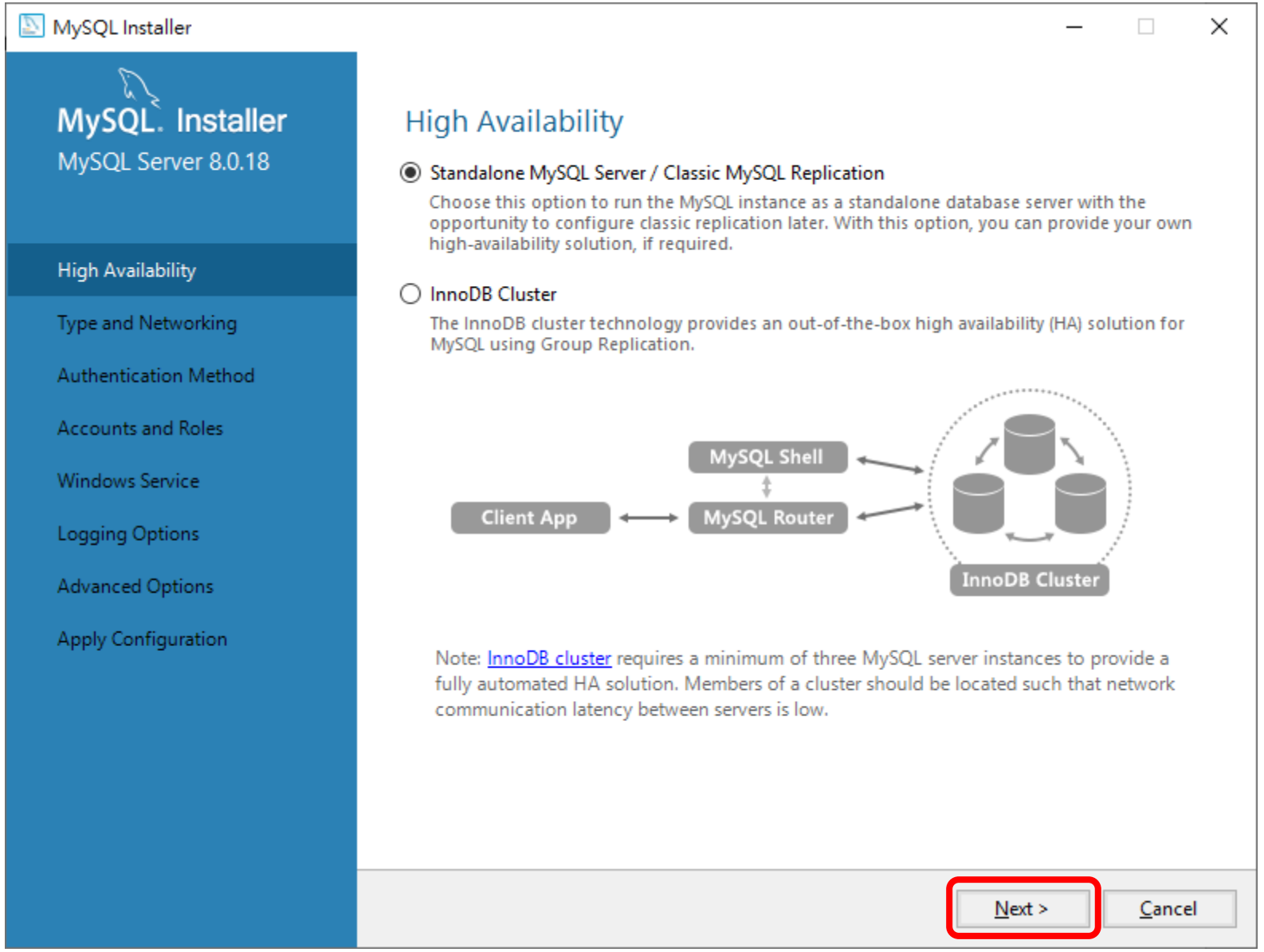
INSTALLATION

Select “Next”.



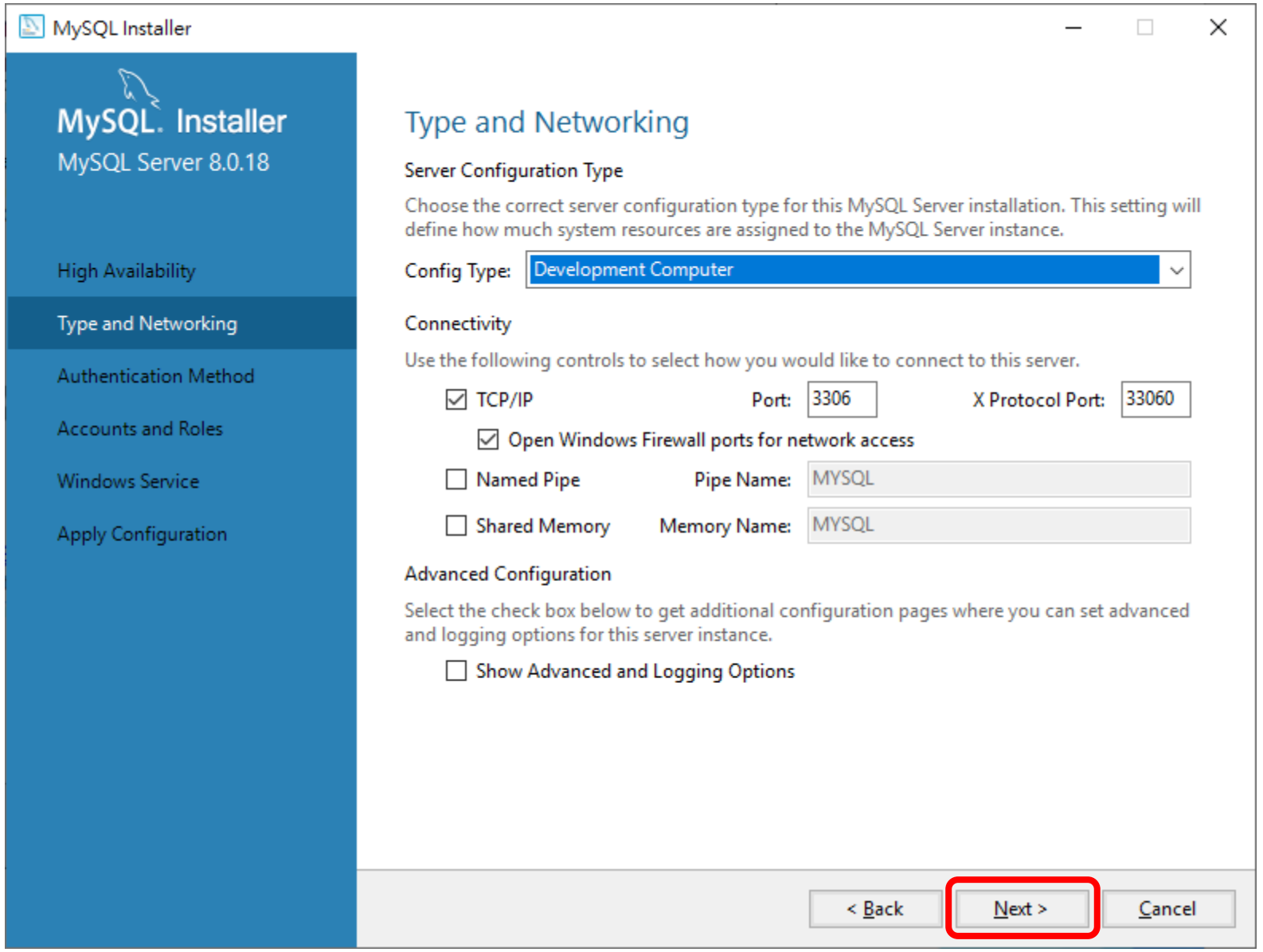
INSTALLATION

Select “Next”.



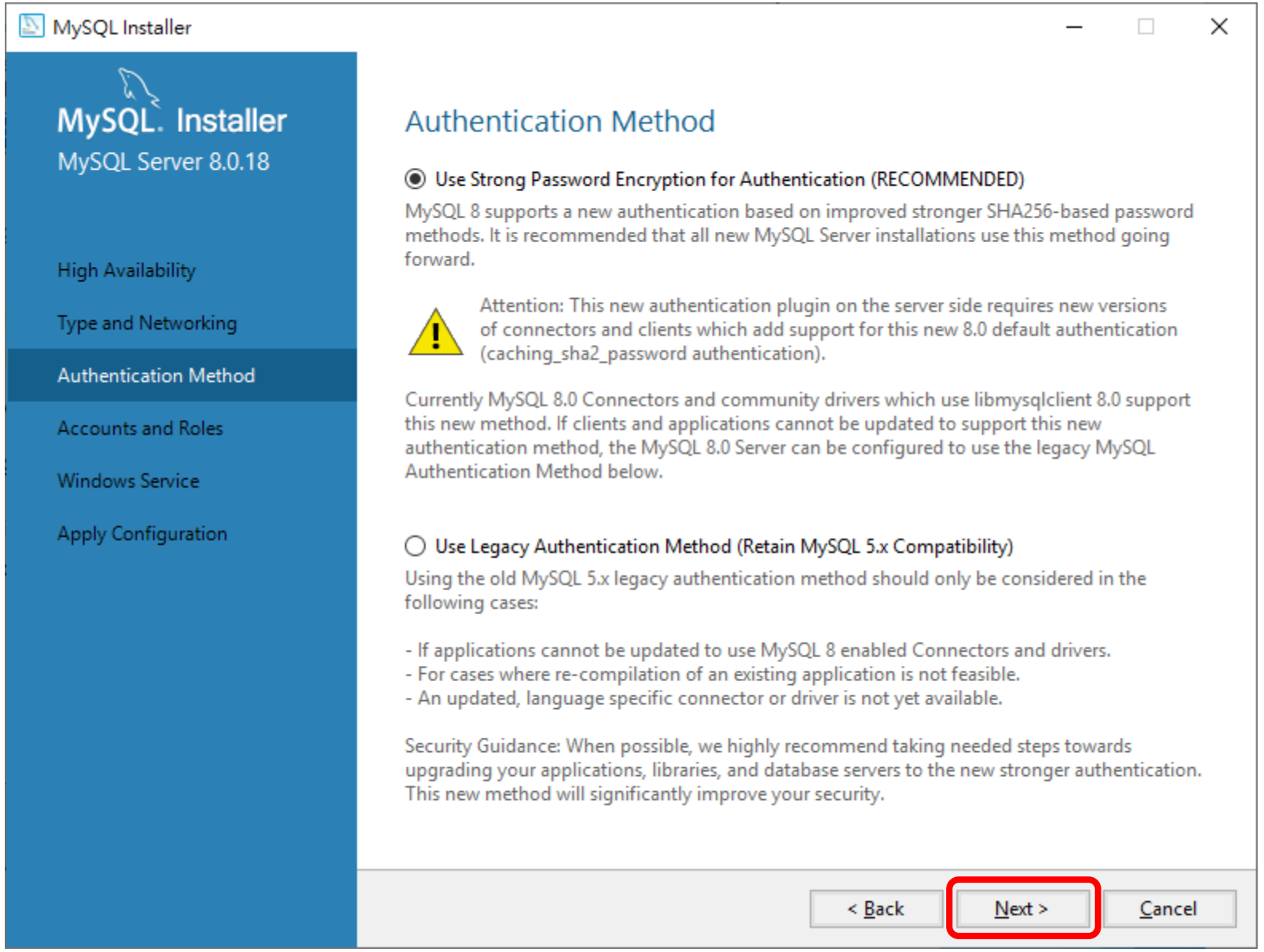
INSTALLATION

Select “Next”.



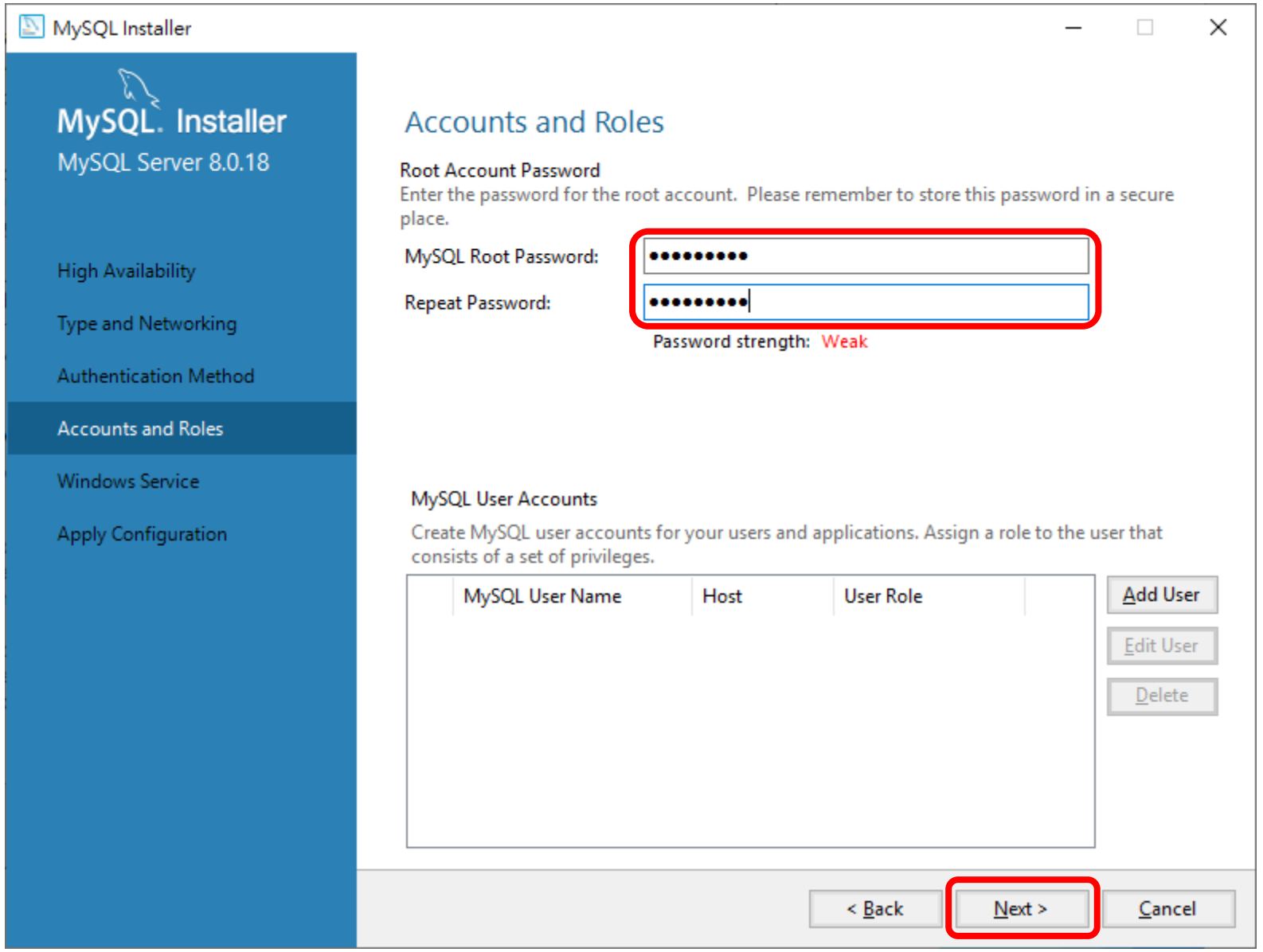
INSTALLATION

Select “Next”.



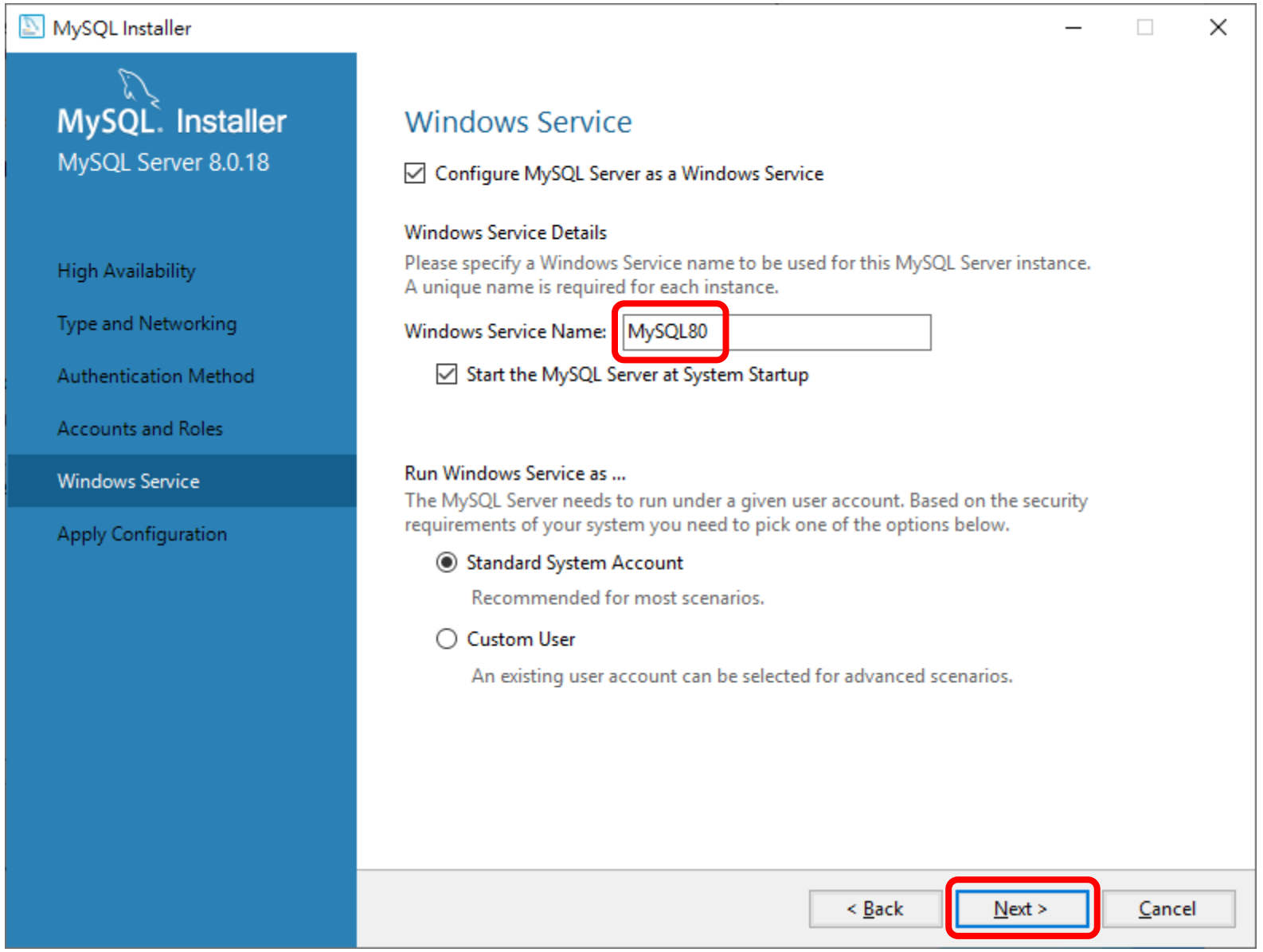
INSTALLATION

Set root password (remember this password, ex: 1234567) and select “Next”.



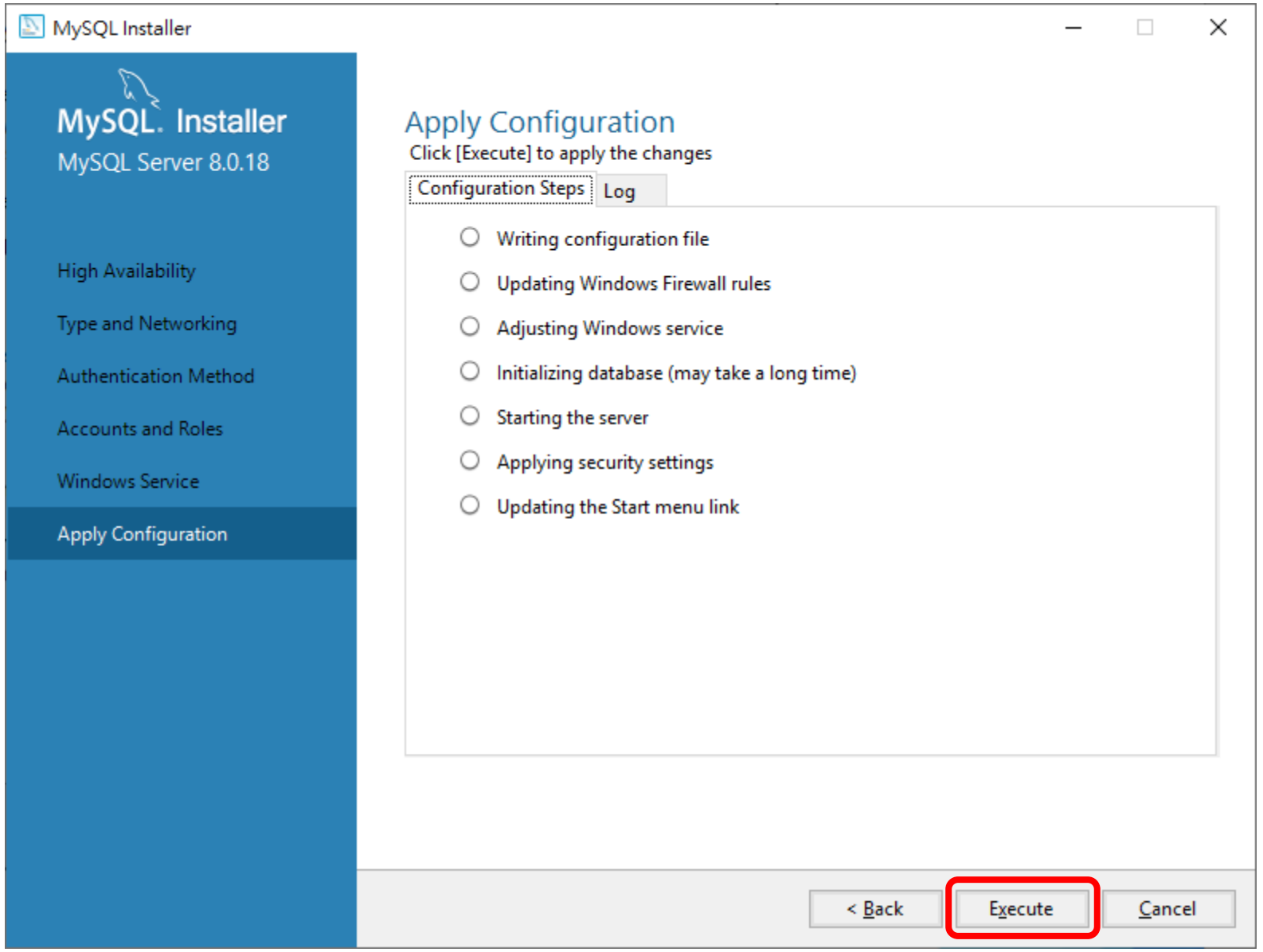
INSTALLATION

Remember the “Windows Service Name” and select “Next”.



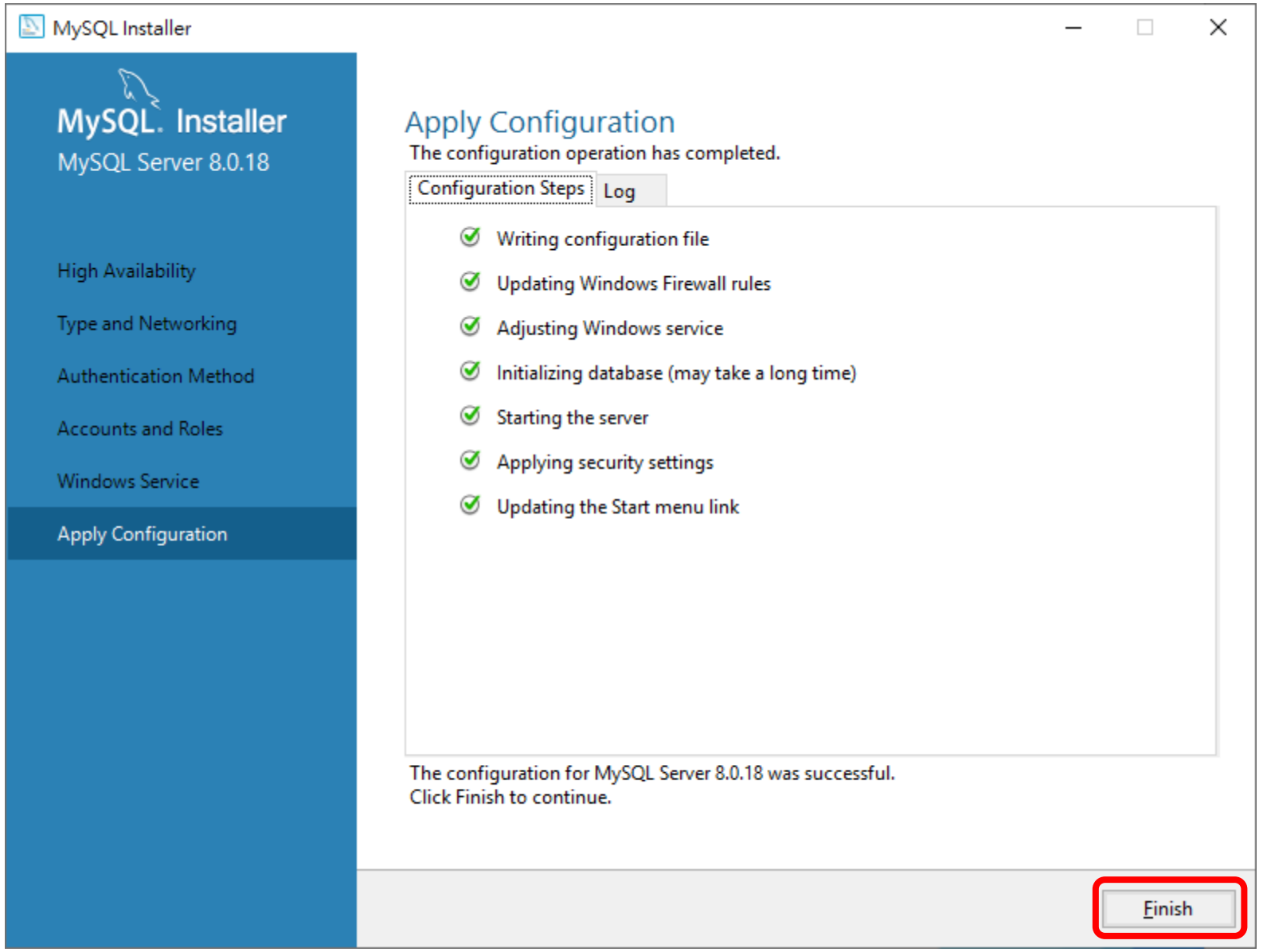
INSTALLATION

Select “Next”.



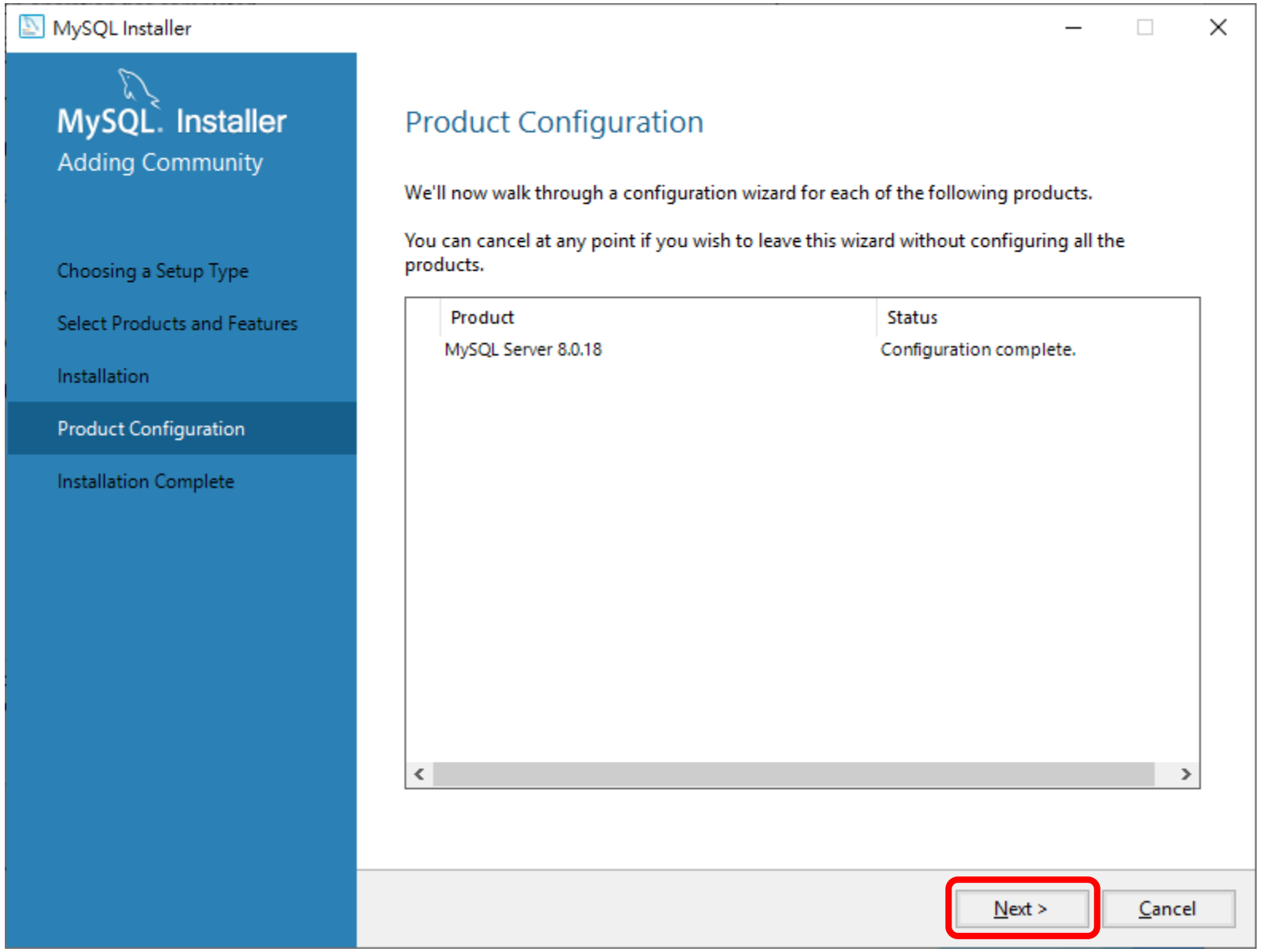
INSTALLATION

Complete the installation and select “Finish”.



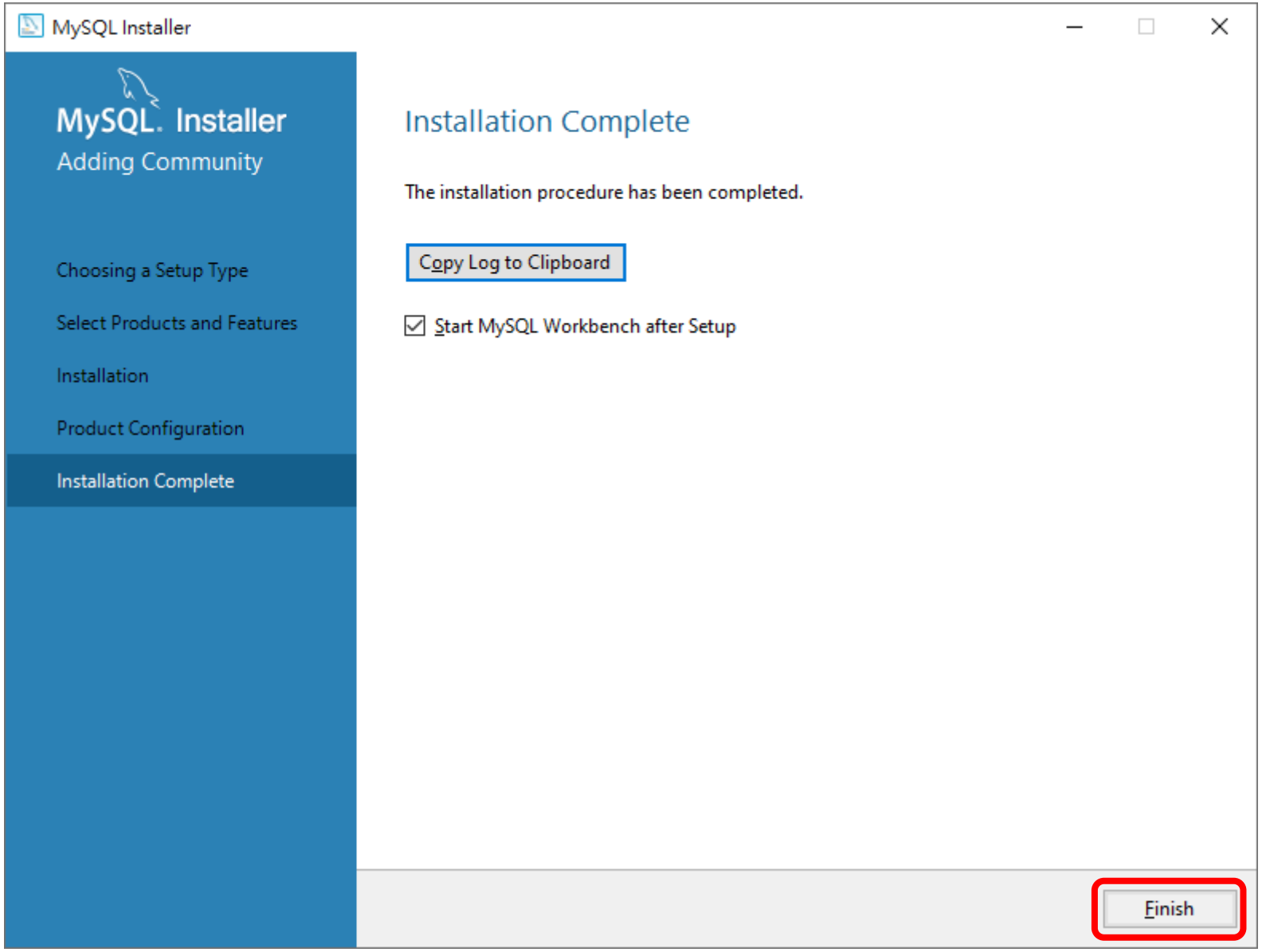
INSTALLATION

Complete the installation and select “Finish”.



INSTALLATION

Complete the installation and select “Finish”.

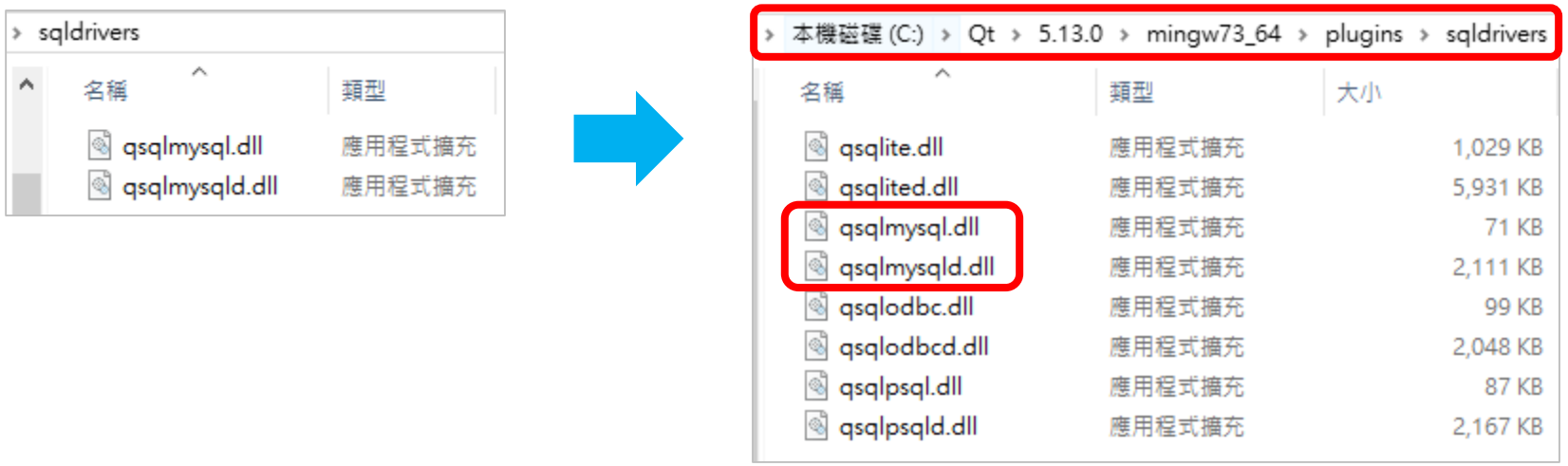


Drivers Setup

SQL Database Drivers

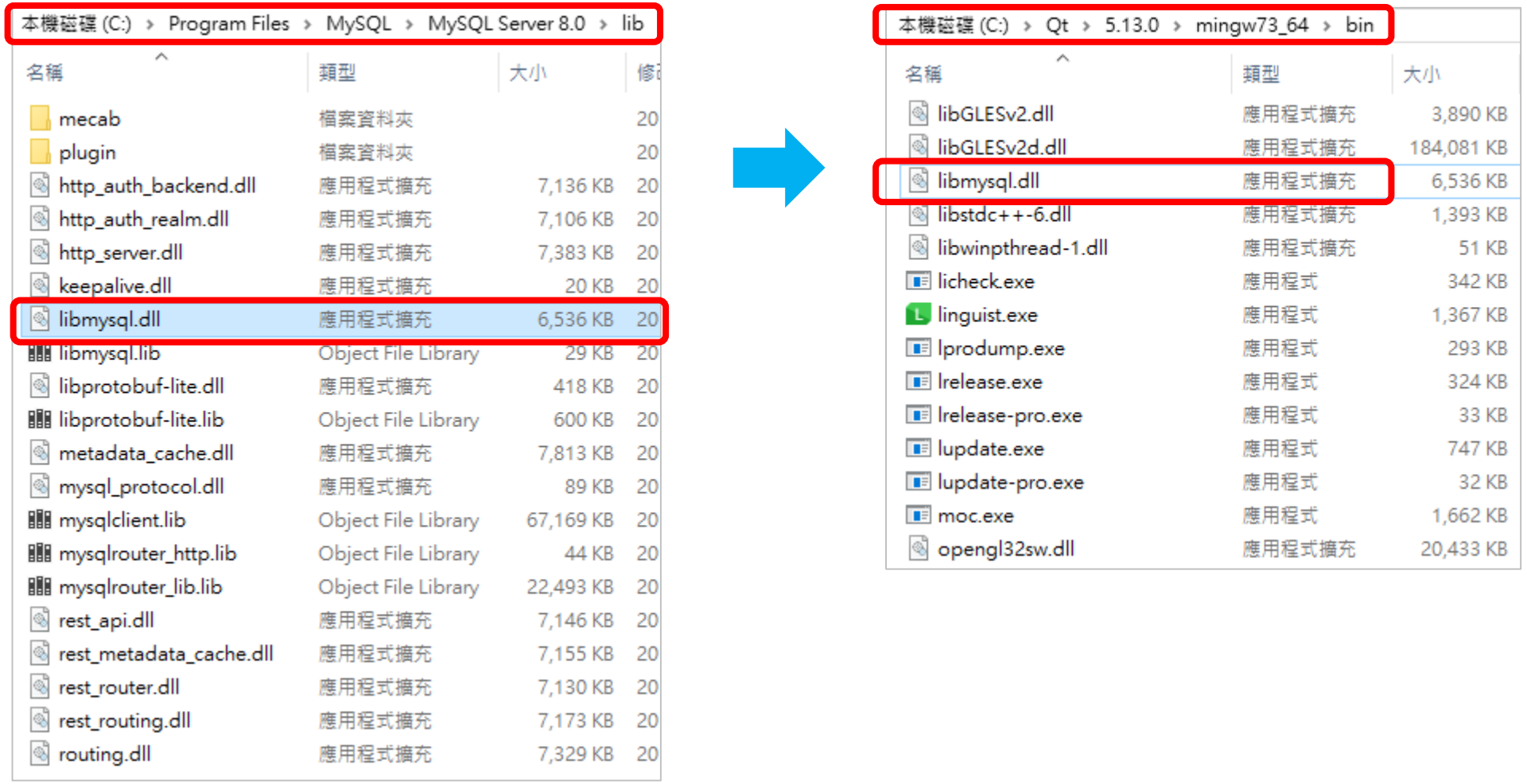
DRIVERS SETUP

Copy “qsqlmysql.dll” and “qsqlmysqld.dll” which are provided by TA to QT plugins folder, for example, “C:\Qt\5.13.0\mingw73_64\plugins\sqldrivers\”



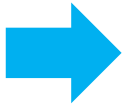
DRIVERS SETUP

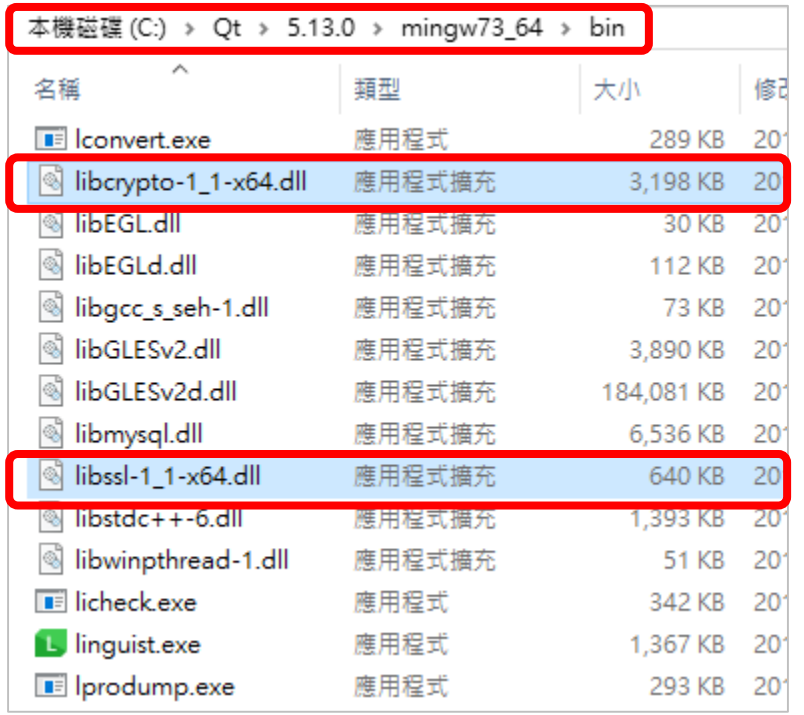
Copy “libmysql.dll” of MySQL to folder “bin” of QT, for example,
copy “C:\Program Files\MySQL\MySQL Server 8.0\lib\ libmysql.dll” to “C:\Qt\5.13.0\mingw73_64\bin\”



DRIVERS SETUP

Copy “libcrypto-1_1-x64.dll” and “libssl-1_1-x64.dll” of MySQL to folder “bin” of QT, for example,
copy “C:\Program Files\MySQL\MySQL Server 8.0\bin\libcrypto-1_1-x64.dll” and “C:\Program Files\MySQL\
MySQL Server 8.0\bin\libssl-1_1-x64.dll” to “C:\Qt\5.13.0\mingw73_64\bin\”



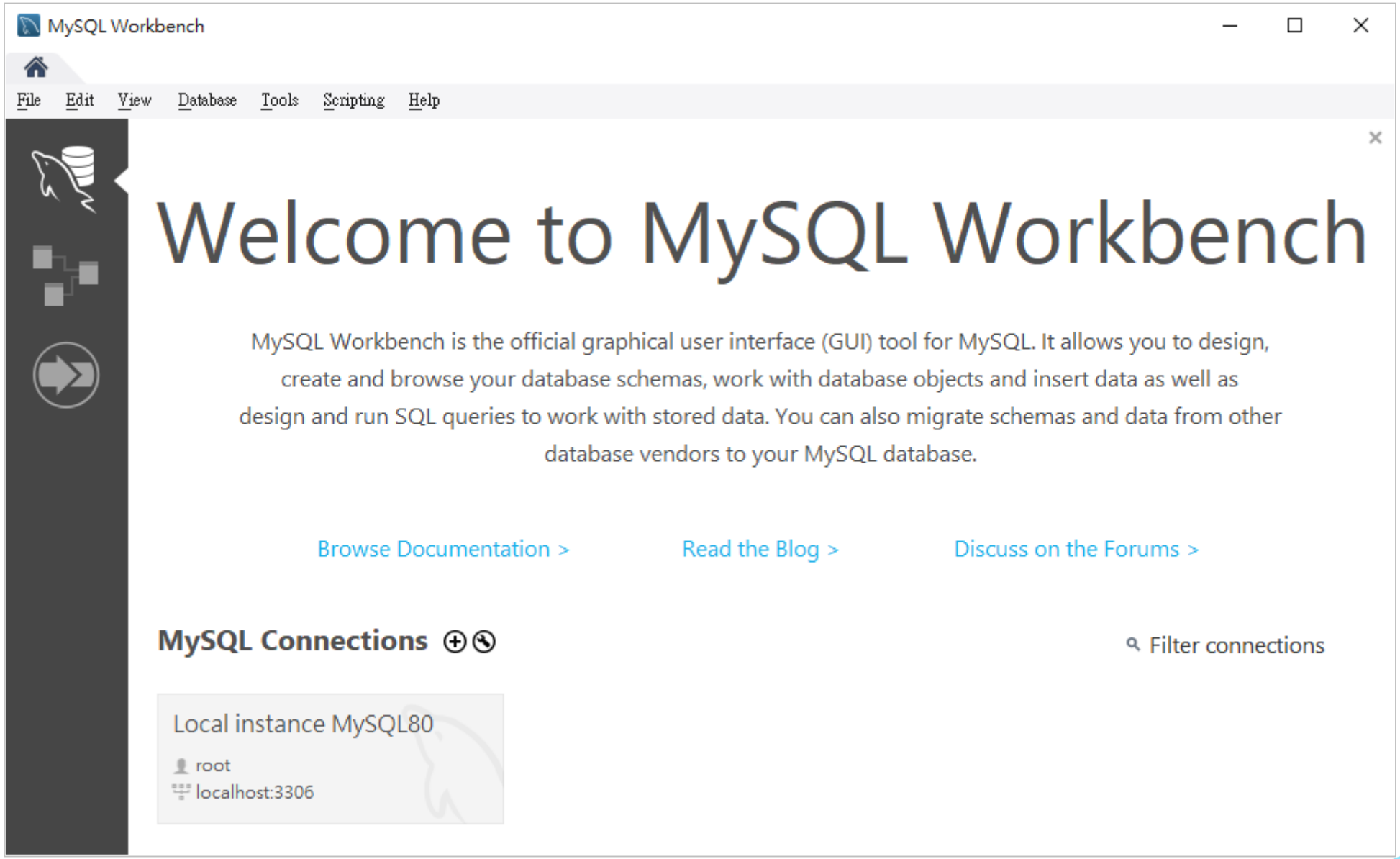


MySQL WorkBench

Environment of SQL Development

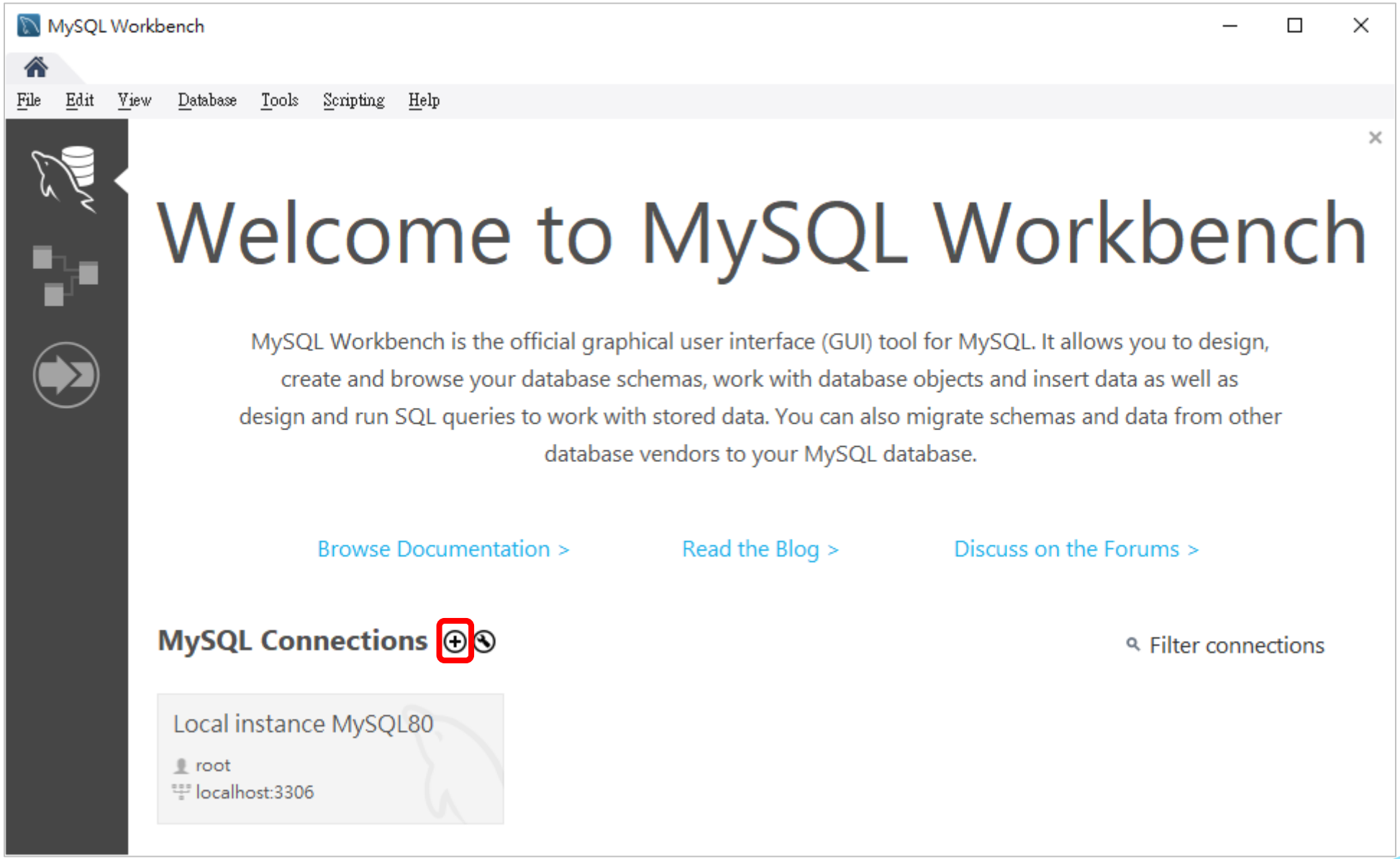
MySQL WorkBench

Execute “MySQL Workbench” firstly.



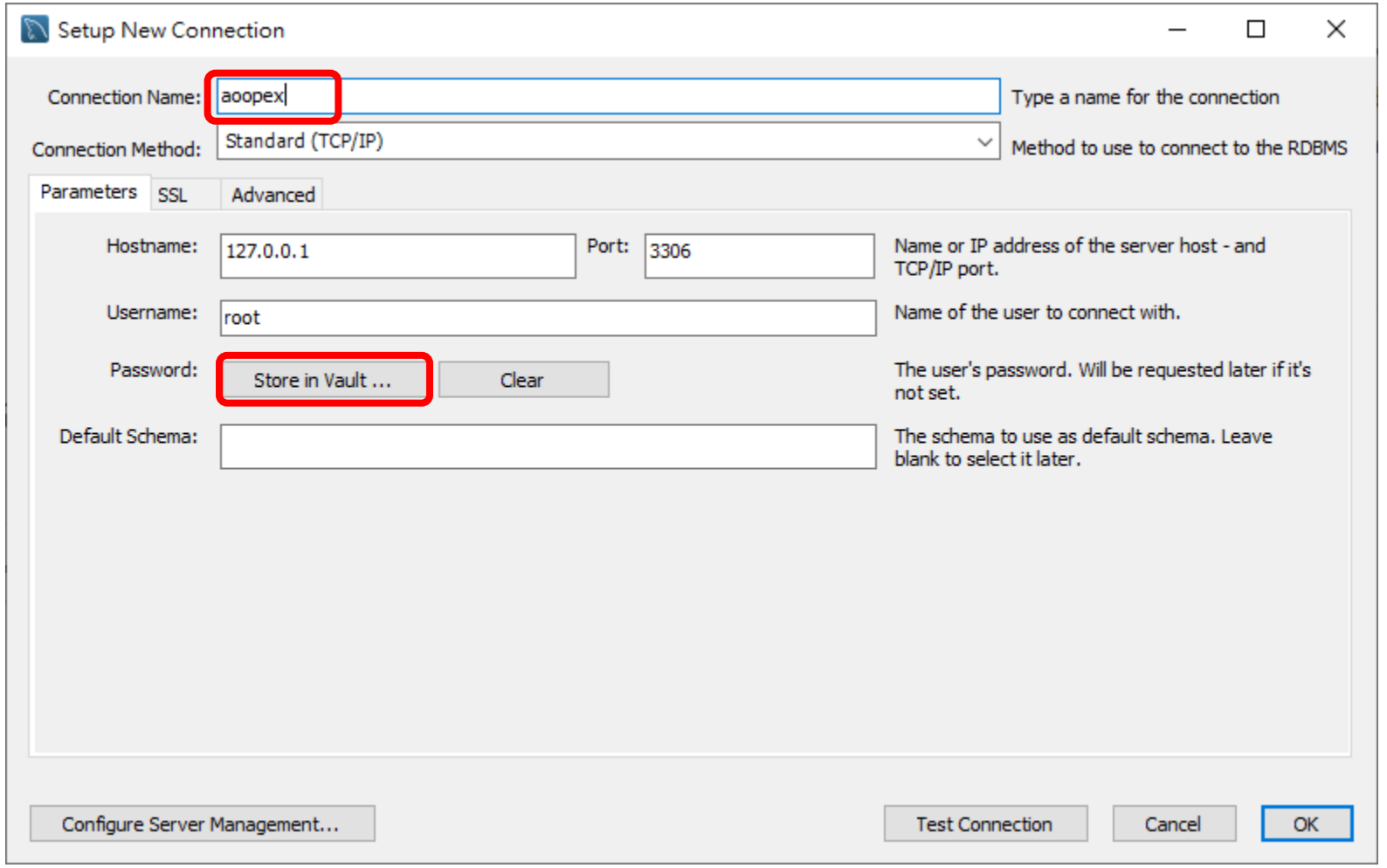
STRUCTURE QUERY LANGUAGE (SQL)

Click “+” button.



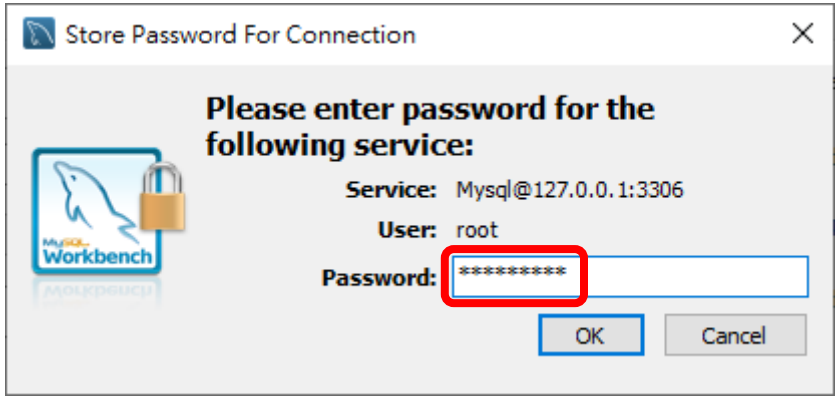
STRUCTURE QUERY LANGUAGE (SQL)

Key in “Connection Name” and then click “Store in Vault …”.



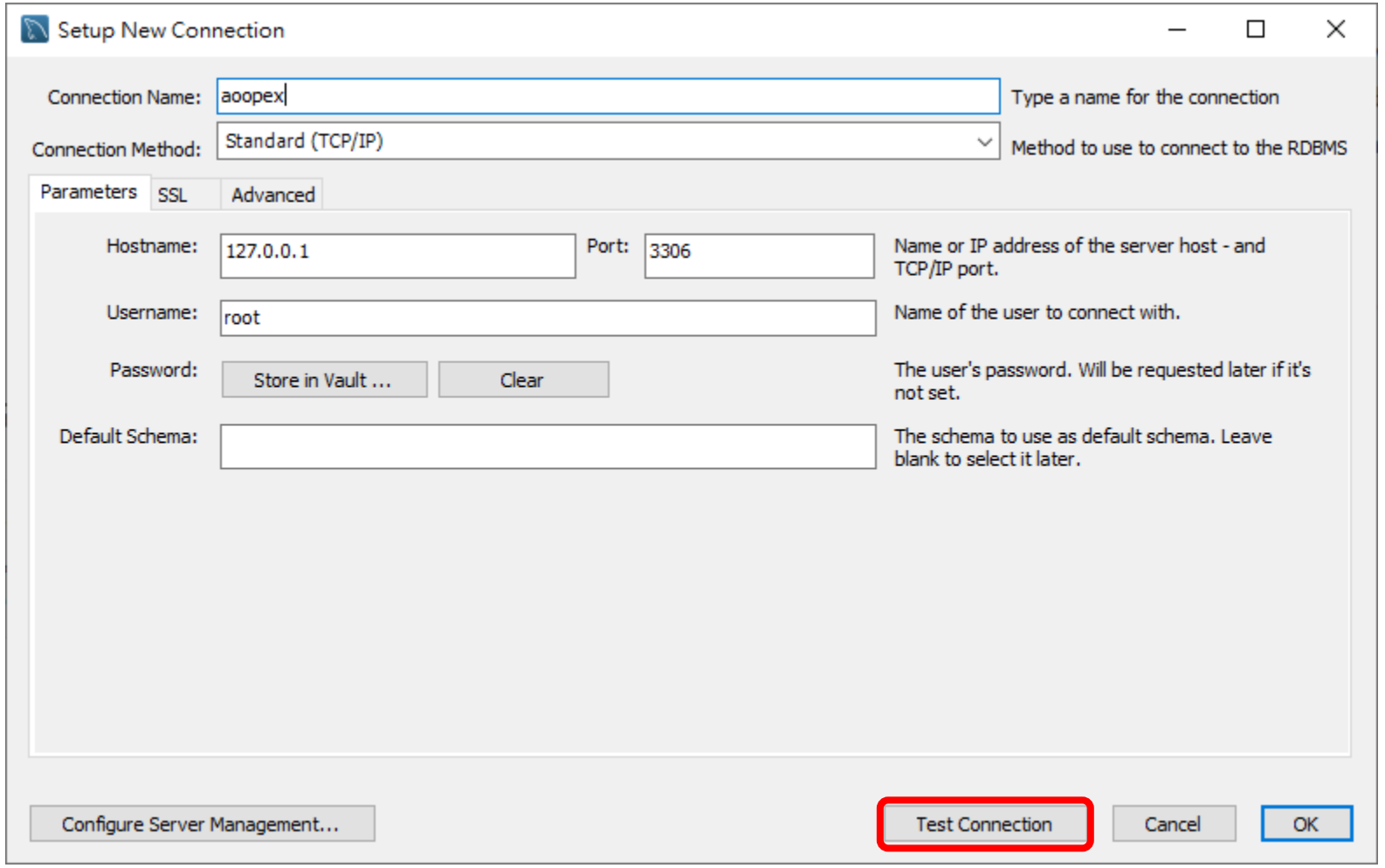
STRUCTURE QUERY LANGUAGE (SQL)

Key in “Password”, ex: 123456789.



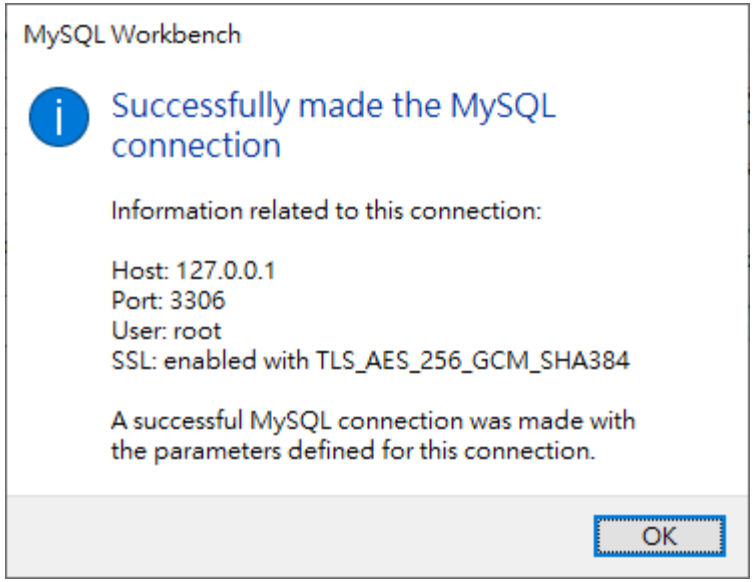
STRUCTURE QUERY LANGUAGE (SQL)

You can click “Test Connection” to check if it can make the MySQL connection successfully.



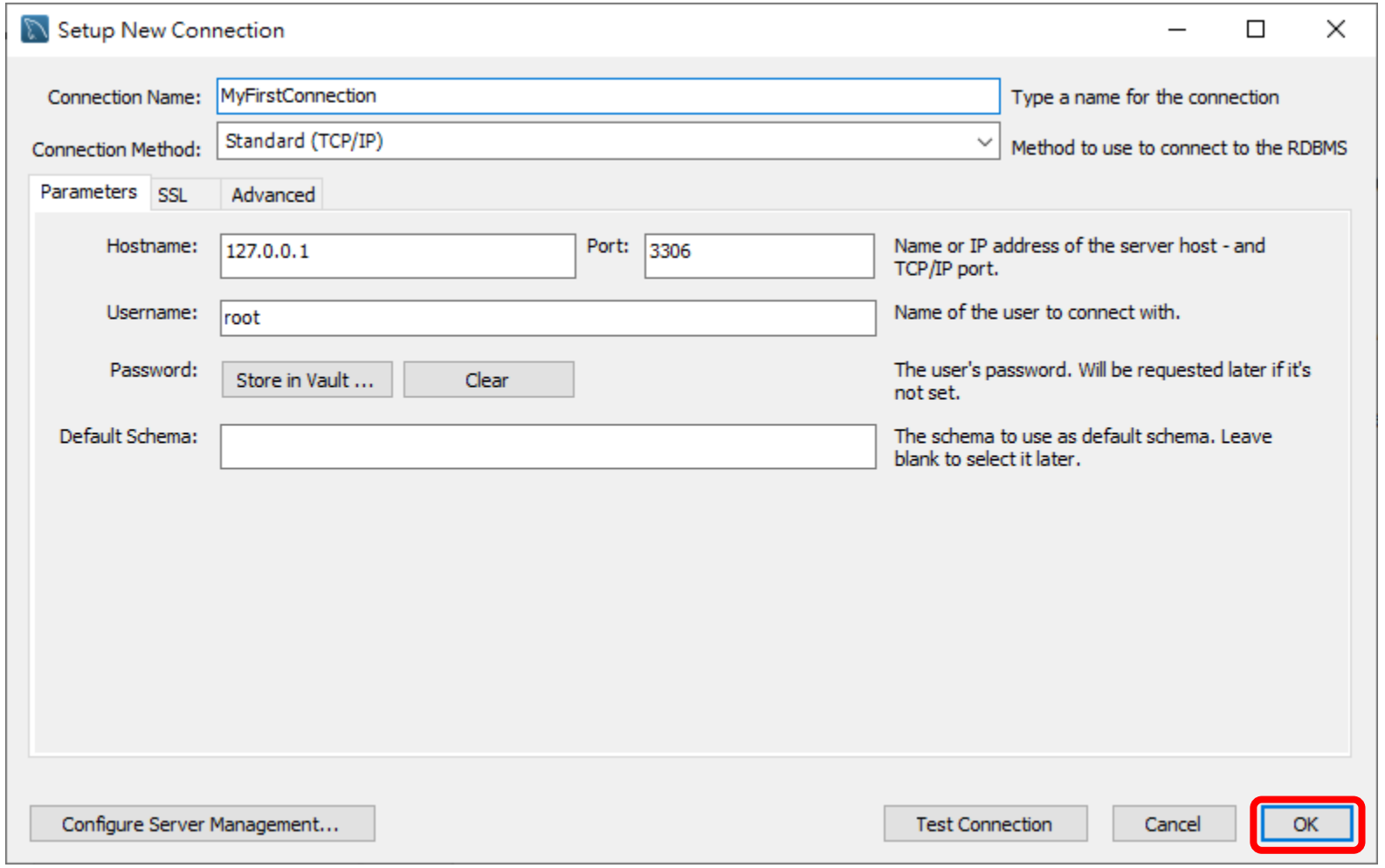
STRUCTURE QUERY LANGUAGE (SQL)

If connection is successful, the following result should be shown.



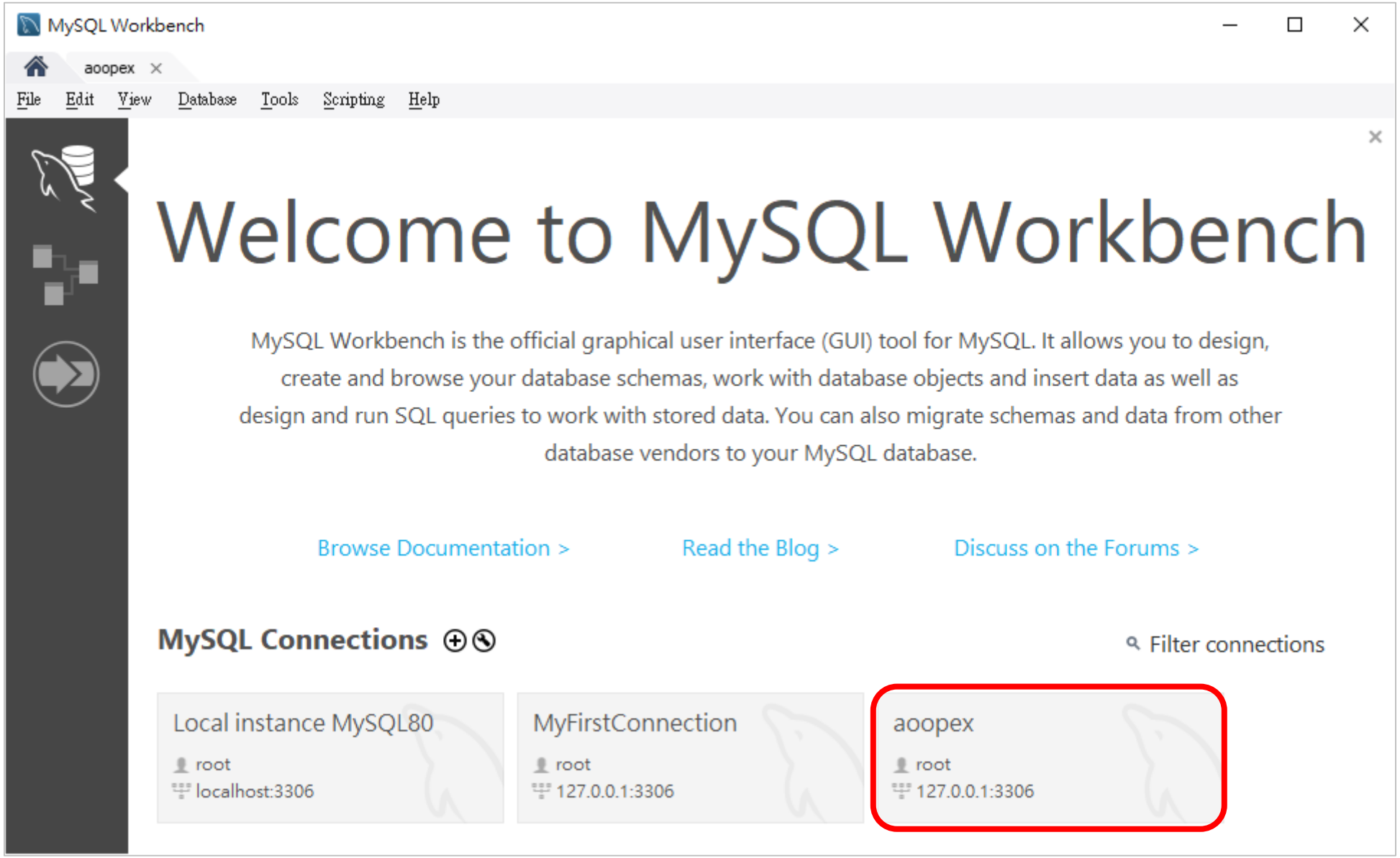
STRUCTURE QUERY LANGUAGE (SQL)

Click “OK”.



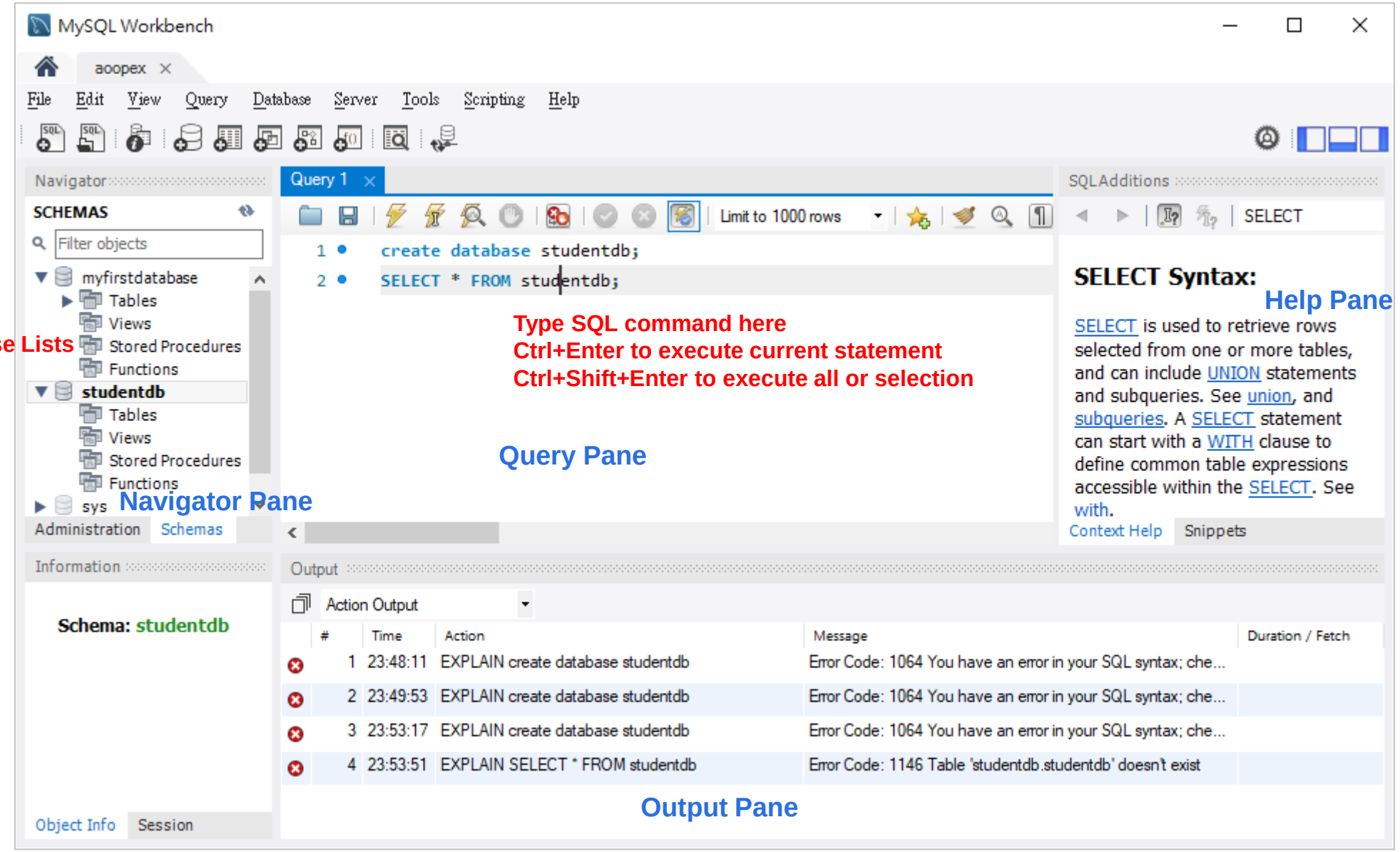
STRUCTURE QUERY LANGUAGE (SQL)

A new connection will be shown as below, and then click it.



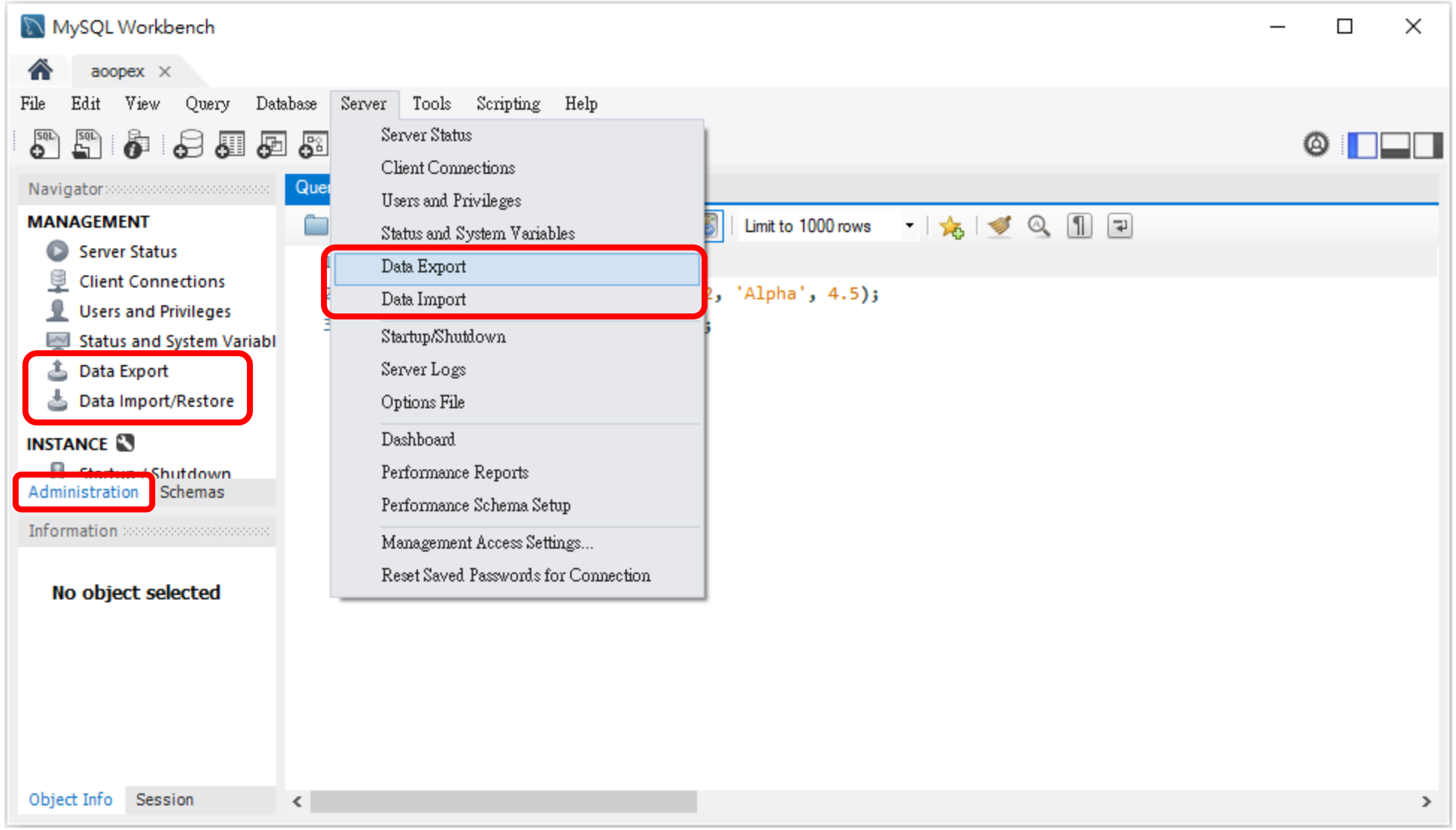
Structure Query Language (SQL)

Execute SQL command.



Structure Query Language (SQL)

Export/Import database.



Structure Query Language (SQL)

Frequently-used MySQL Commands Part I

STRUCTURE QUERY LANGUAGE (SQL) Part I

Basic Knowledge

- **Case Sensitivity**: SQL keywords, names (identifiers), strings may or may not be case-sensitive, depending on the implementation.
  - In MySQL, the *keywords* are NOT case-sensitive. For clarity, we show the keywords in UPPERCASE in this lecture.
  - For programmers, it is BEST to treat the *names (identifiers)* and *strings* as case-sensitive. (In MySQL, column-names are always case insensitive; but table-names are case-sensitive in Unix, but case-insensitive in Windows (confused!!). Case-sensitivity in string comparison depends on the *collating sequence* used (?!).)

- **String**: SQL strings are enclosed in single quotes. But most implementations (such as MySQL) accept both single and double quotes.

- **Syntax**: single line comment --content or #content
  multiple lines comment /* content */

STRUCTURE QUERY LANGUAGE (SQL) Part I

Data type – Numeric types

Data Type	Explanation
tinyint	A very small integer. The signed range for this numeric data type is -128 to 127, while the unsigned range is 0 to 255.
smallint	A small integer. The signed range for this numeric type is -32768 to 32767, while the unsigned range is 0 to 65535.
mediumint	A medium-sized integer. The signed range for this numeric data type is -8388608 to 8388607, while the unsigned range is 0 to 16777215.
int or integer	A normal-sized integer. The signed range for this numeric data type is -2147483648 to 2147483647, while the unsigned range is 0 to 4294967295.
bigint	A large integer. The signed range for this numeric data type is -9223372036854775808 to 9223372036854775807, while the unsigned range is 0 to 18446744073709551615.
float	A small (single-precision) floating-point number.
double, double precision, or real	A normal sized (double-precision) floating-point number.
dec, decimal, fixed, or numeric	A packed fixed-point number. The display length of entries for this data type is defined when the column is created, and every entry adheres to that length.
bool or boolean	A Boolean is a data type that only has two possible values, usually either true or false.
bit	A bit value type for which you can specify the number of bits per value, from 1 to 64.

STRUCTURE QUERY LANGUAGE (SQL) Part I

Data type – Date and time types

Data Type	Explanation
date	A date, represented as YYYY-MM-DD.
datetime	A timestamp showing the date and time, displayed as YYYY-MM-DD HH:MM:SS.
timestamp	A timestamp indicating the amount of time since the Unix epoch (00:00:00 on January 1, 1970).
time	A time of day, displayed as HH:MM:SS.
year	A year expressed in either a 2 or 4 digit format, with 4 digits being the default.

STRUCTURE QUERY LANGUAGE (SQL) Part I

Data type – String types

Data Type	Explanation
char	A fixed-length string; entries of this type are padded on the right with spaces to meet the specified length when stored.
varchar	A string of variable length.
binary	Similar to the char type, but a binary byte string of a specified length rather than a nonbinary character string.
varbinary	Similar to the varchar type, but a binary byte string of a variable length rather than a nonbinary character string.
blob	A binary string with a maximum length of 65535 (2^16 - 1) bytes of data.
tinyblob	A blob column with a maximum length of 255 (2^8 - 1) bytes of data.
mediumblob	A blob column with a maximum length of 16777215 (2^24 - 1) bytes of data.
longblob	A blob column with a maximum length of 4294967295 (2^32 - 1) bytes of data.
text	A string with a maximum length of 65535 (2^16 - 1) characters.
tinytext	A text column with a maximum length of 255 (2^8 - 1) characters.
mediumtext	A text column with a maximum length of 16777215 (2^24 - 1) characters.
longtext	A text column with a maximum length of 4294967295 (2^32 - 1) characters.
enum	An enumeration, which is a string object that takes a single value from a list of values that are declared when the table is created.
set	Similar to an enumeration, a string object that can have zero or more values, each of which must be chosen from a list of allowed values that are specified when the table is created.

STRUCTURE QUERY LANGUAGE (SQL) Part I

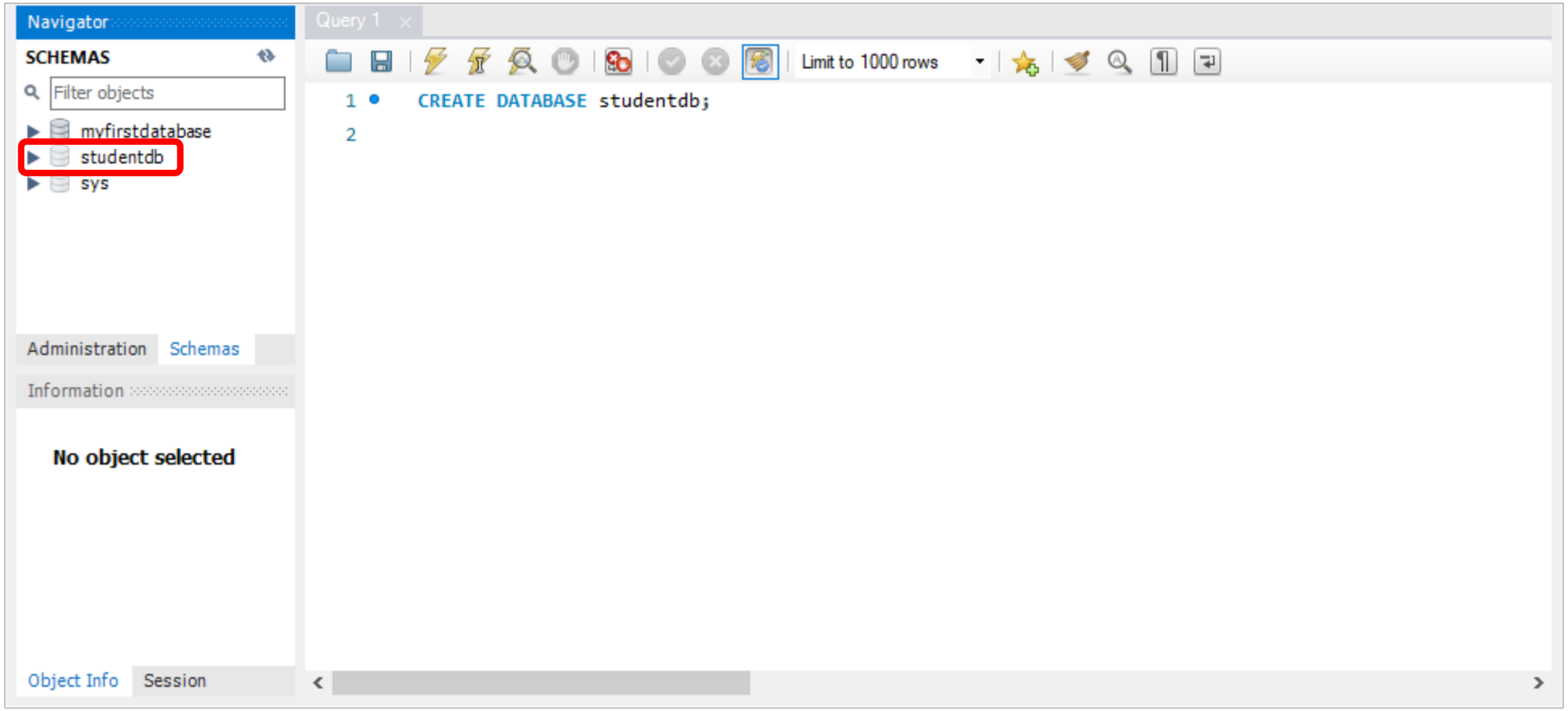
Frequently-used MySQL Commands (MySQL commands are NOT case sensitive).

```
-- General
; -- Sends command to server for processing (or \g)
\c -- Cancels (aborts) the current command

-- Database-level
DROP DATABASE databaseName; -- Deletes the database
DROP DATABASE IF EXISTS databaseName; -- Deletes only if it exists
CREATE DATABASE databaseName; -- Creates a new database
CREATE DATABASE IF NOT EXISTS databaseName; -- Creates only if it does not exists
SHOW DATABASES; -- Shows all databases in this server

-- Set default database.
-- Otherwise you need to use the fully-qualified name, in the form of "databaseName.tableName", to refer to a table.
USE databaseName;

-- Table-level
DROP TABLE tableName;
DROP TABLE IF EXISTS tableName;
CREATE TABLE tableName (column1Definition, column2Definition, ...);
CREATE TABLE IF NOT EXISTS tableName (column1Definition, column2Definition, ...);
SHOW TABLES; -- Shows all the tables in the default database
DESCRIBE tableName; -- Describes the columns for the table
DESC tableName; -- Same as above

-- Record-level (CURD - create, update, read, delete)
INSERT INTO tableName VALUES (column1Value, column2Value,...);
INSERT INTO tableName (column1Name, ..., columnNName) VALUES (column1Value, ..., columnNValue);
DELETE FROM tableName WHERE criteria;
UPDATE tableName SET columnName = expression WHERE criteria;
SELECT column1Name, column2Name, ... FROM tableName WHERE criteria ORDER BY columnAName ASC|DESC, columnBName ASC|DESC, ...;

-- Running a script of MySQL statements
SOURCE full-Path-Filename
```

STRUCTURE QUERY LANGUAGE (SQL) Part I

Create a new database.

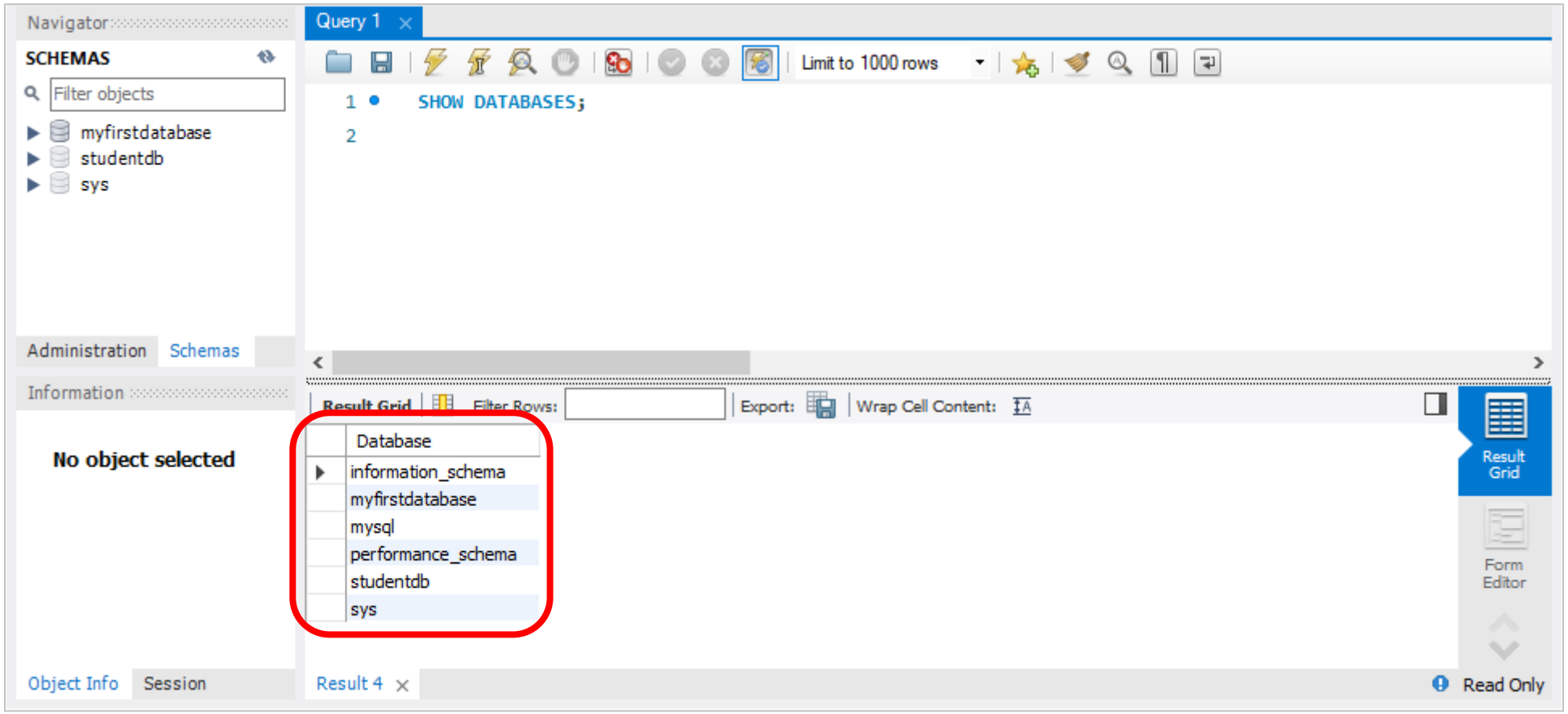
```
-- SYNTAX
CREATE DATABASE databaseName;
CREATE DATABASE IF NOT EXISTS databaseName;

-- EXAMPLES
CREATE DATABASE studentdb;
    -- Create a database named studentdb.
```



STRUCTURE QUERY LANGUAGE (SQL) Part I

Show all databases.

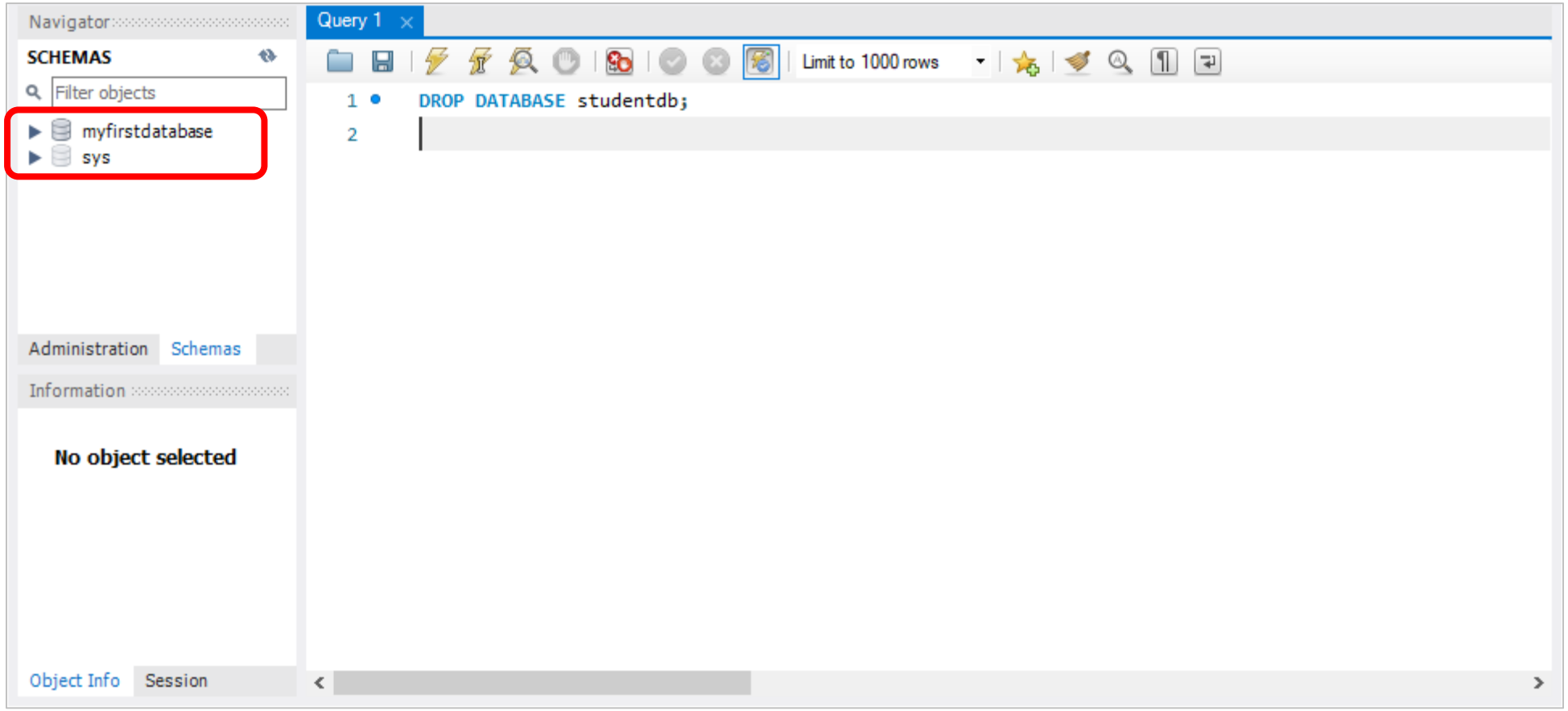
```
-- SYNTAX
SHOW DATABASES;

-- EXAMPLES
SHOW DATABASES;
    -- Show all databases in this server.
```



STRUCTURE QUERY LANGUAGE (SQL) Part I

Delete the database.

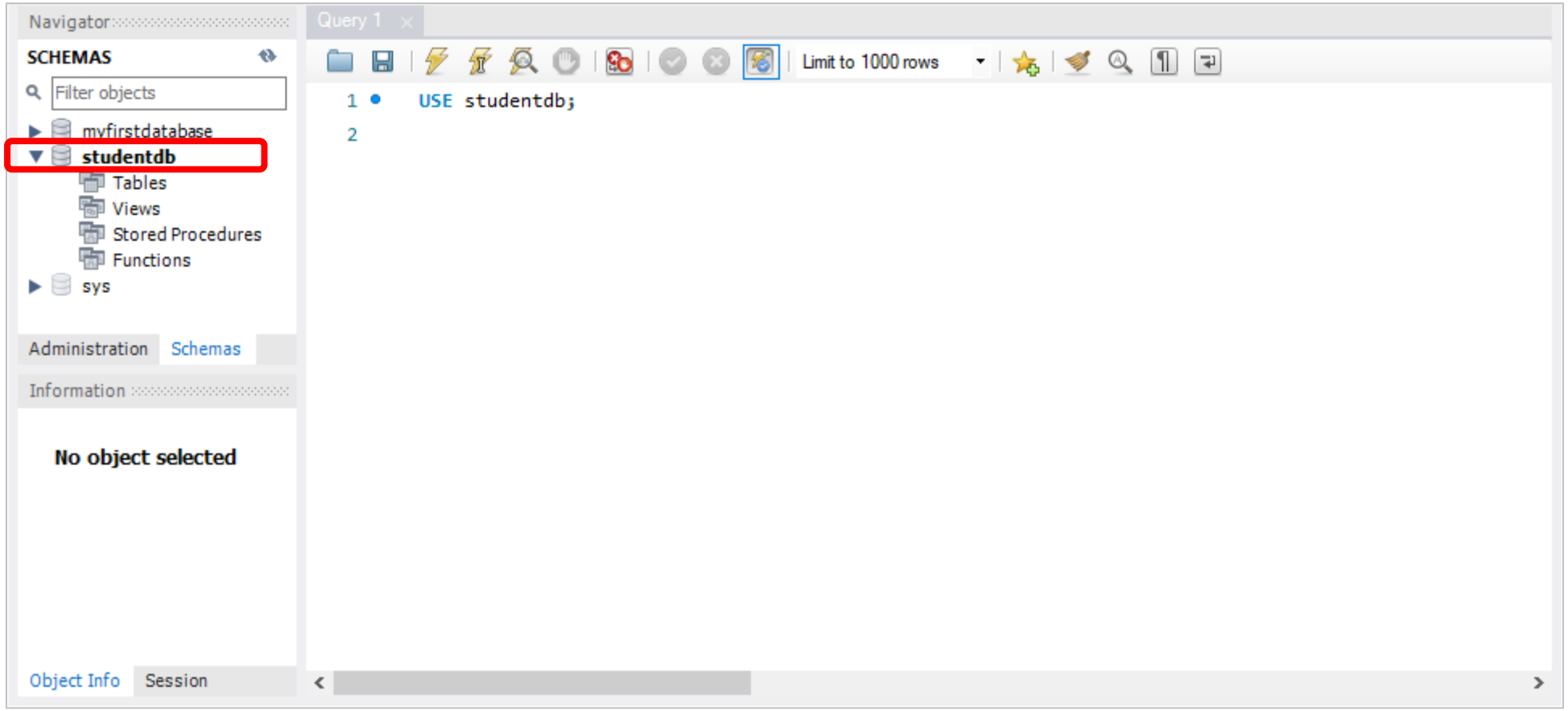
```
-- SYNTAX
DROP DATABASE databaseName;
DROP DATABASE IF EXISTS databaseName;

-- EXAMPLES
DROP DATABASE studentdb;
    -- Delete a database named studentdb.
```



STRUCTURE QUERY LANGUAGE (SQL) Part I

Set default database.

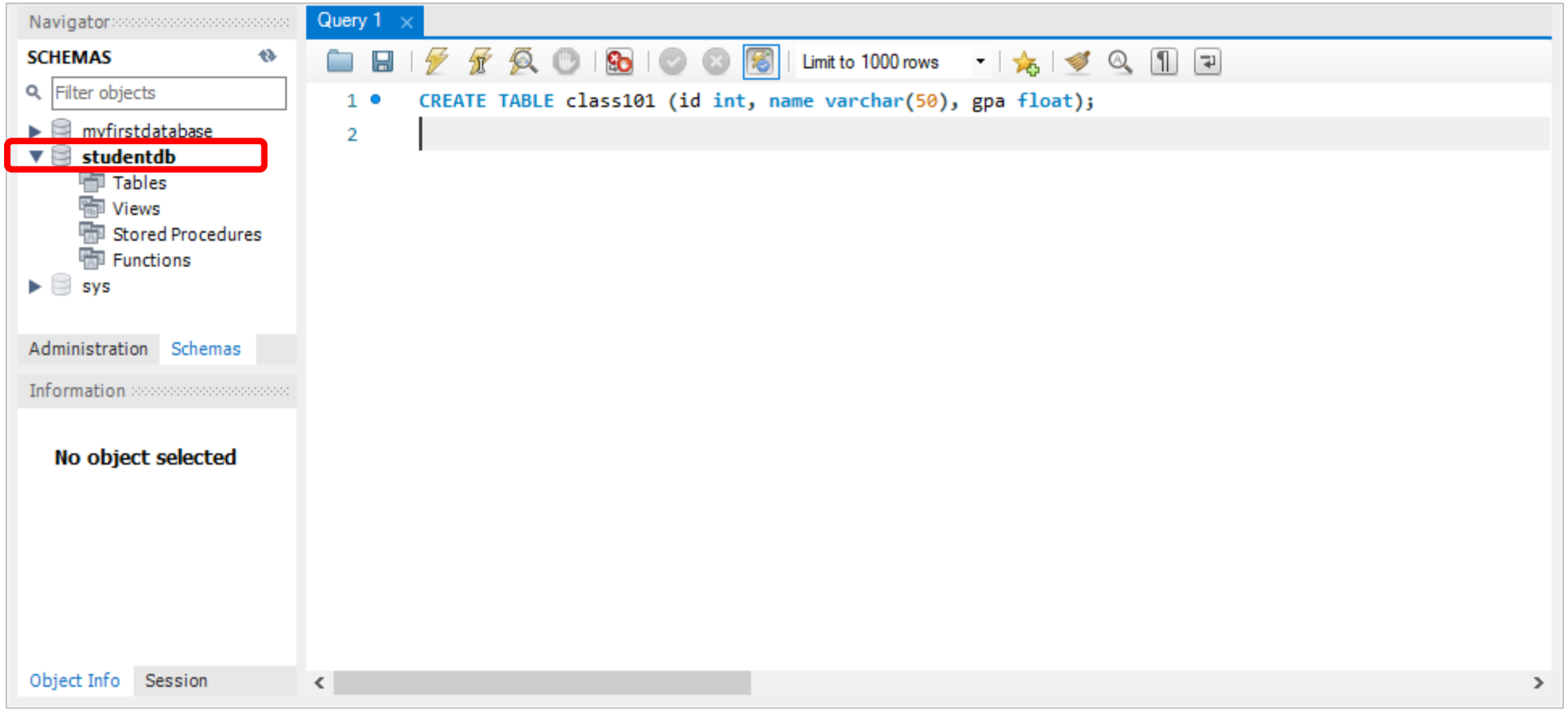
```
-- SYNTAX
USE databaseName;

-- EXAMPLES
USE studentdb;
    -- Use 'studentdb' database as the default (current) database
    -- You can refer to tables in the default database by the 'tablename' alone,
    -- instead of 'databasename.tablename' for non-default database.
```



STRUCTURE QUERY LANGUAGE (SQL) Part I

Create a new table.

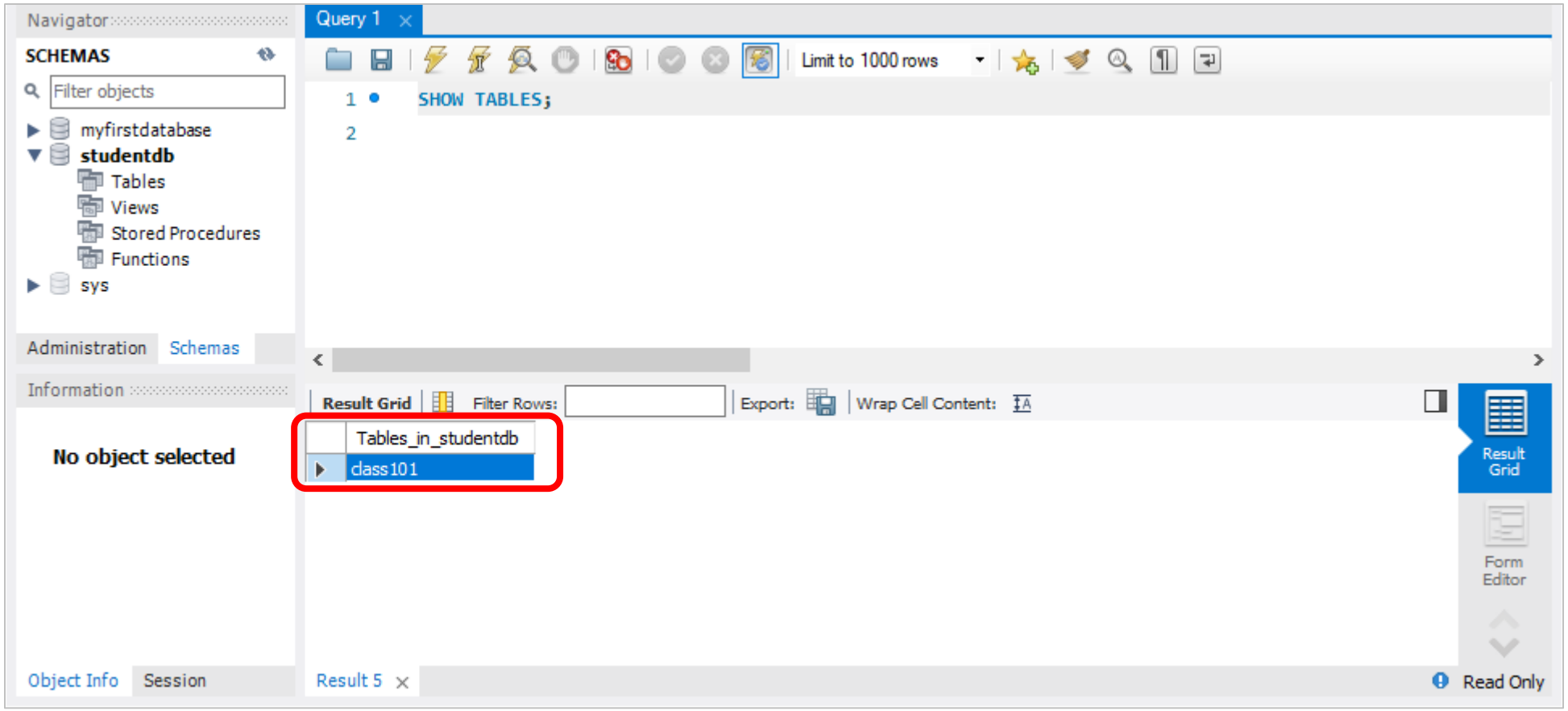
```
-- SYNTAX
CREATE TABLE tableName (column1Definition, column2Definition, ...);
CREATE TABLE IF NOT EXISTS tableName (column1Definition, column2Definition, ...);

-- EXAMPLES
CREATE TABLE class101 (id int, name varchar(50), gpa float);
    -- Create a new table called 'class101' in the default database 'studentdb',
    -- with 3 columns of the specified types
```



STRUCTURE QUERY LANGUAGE (SQL) Part I

List all tables.

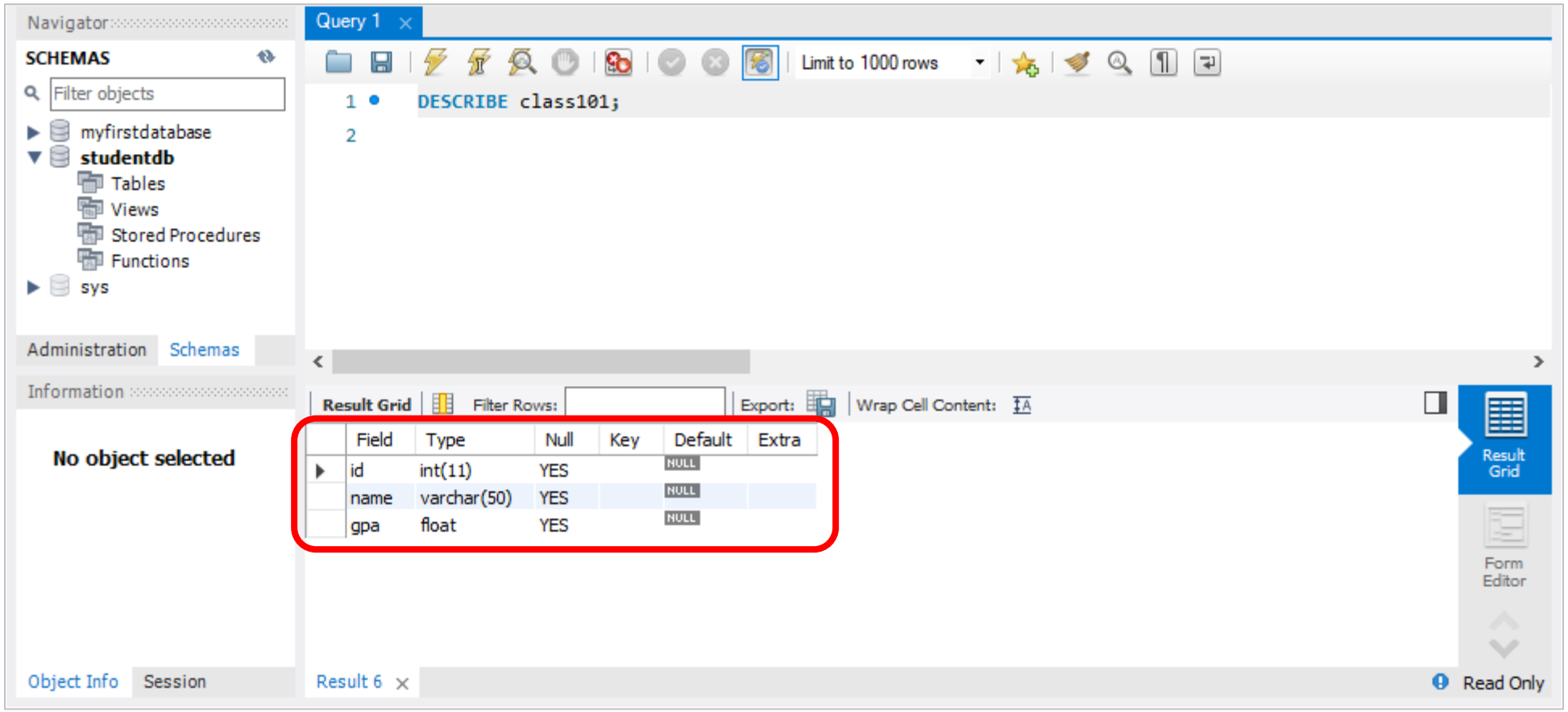
```
-- SYNTAX
SHOW TABLES;

-- EXAMPLES
SHOW TABLES;
    -- List all the tables in the default database 'studentdb'
```



STRUCTURE QUERY LANGUAGE (SQL) Part I

Describes the columns for the table.

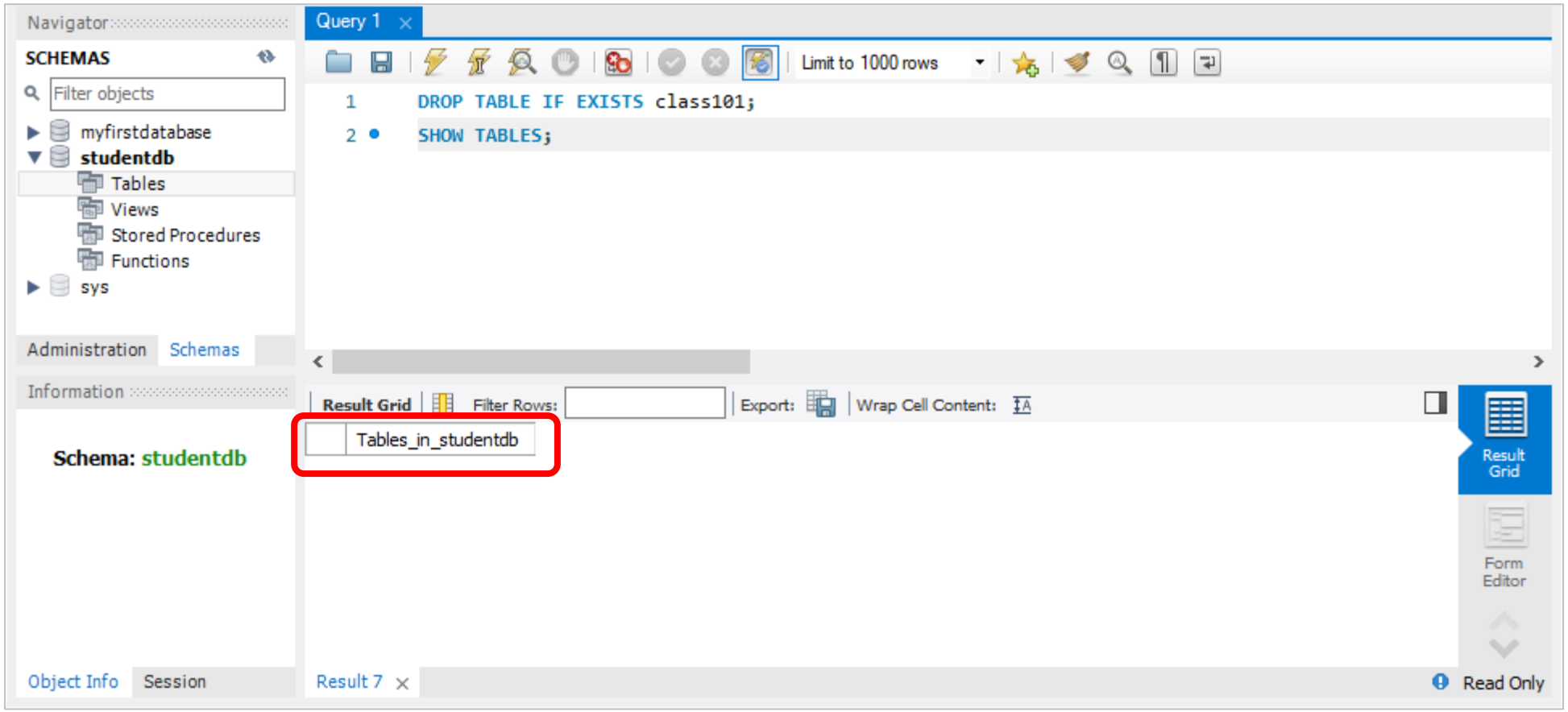
```
-- SYNTAX
DESCRIBE tableName;
DESC tableName;

-- EXAMPLES
DESCRIBE class101;
    -- Describe the 'class101' table (List its columns' definition)
```



STRUCTURE QUERY LANGUAGE (SQL) Part I

Remove the table.

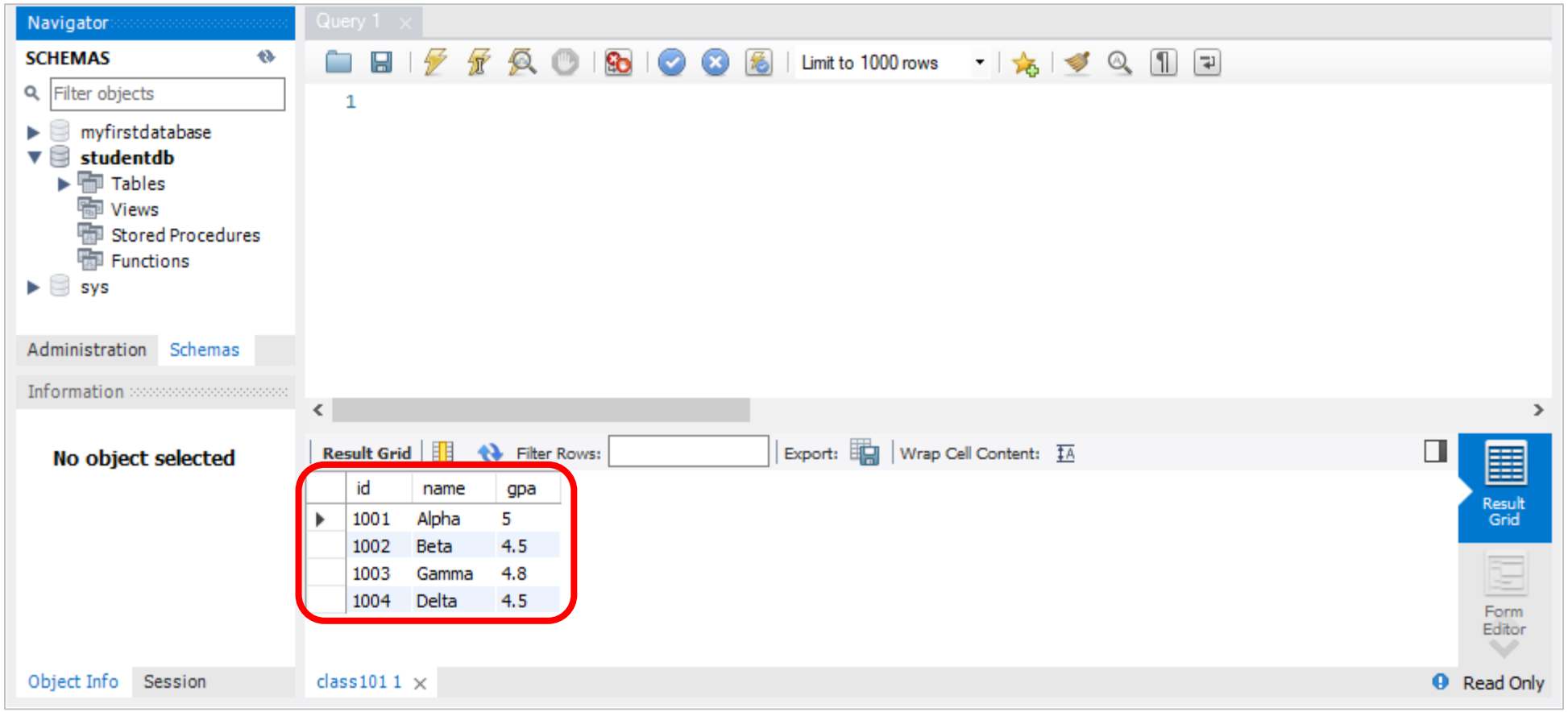
```
-- SYNTAX
DROP TABLE tableName;
DROP TABLE IF EXISTS tableName;

-- EXAMPLES
DROP TABLE IF EXISTS class101;
    -- Remove the table 'class101' in the default database if it exists.
    -- Beware that there is No UNDO!!!
```



STRUCTURE QUERY LANGUAGE (SQL) Part I

SELECT

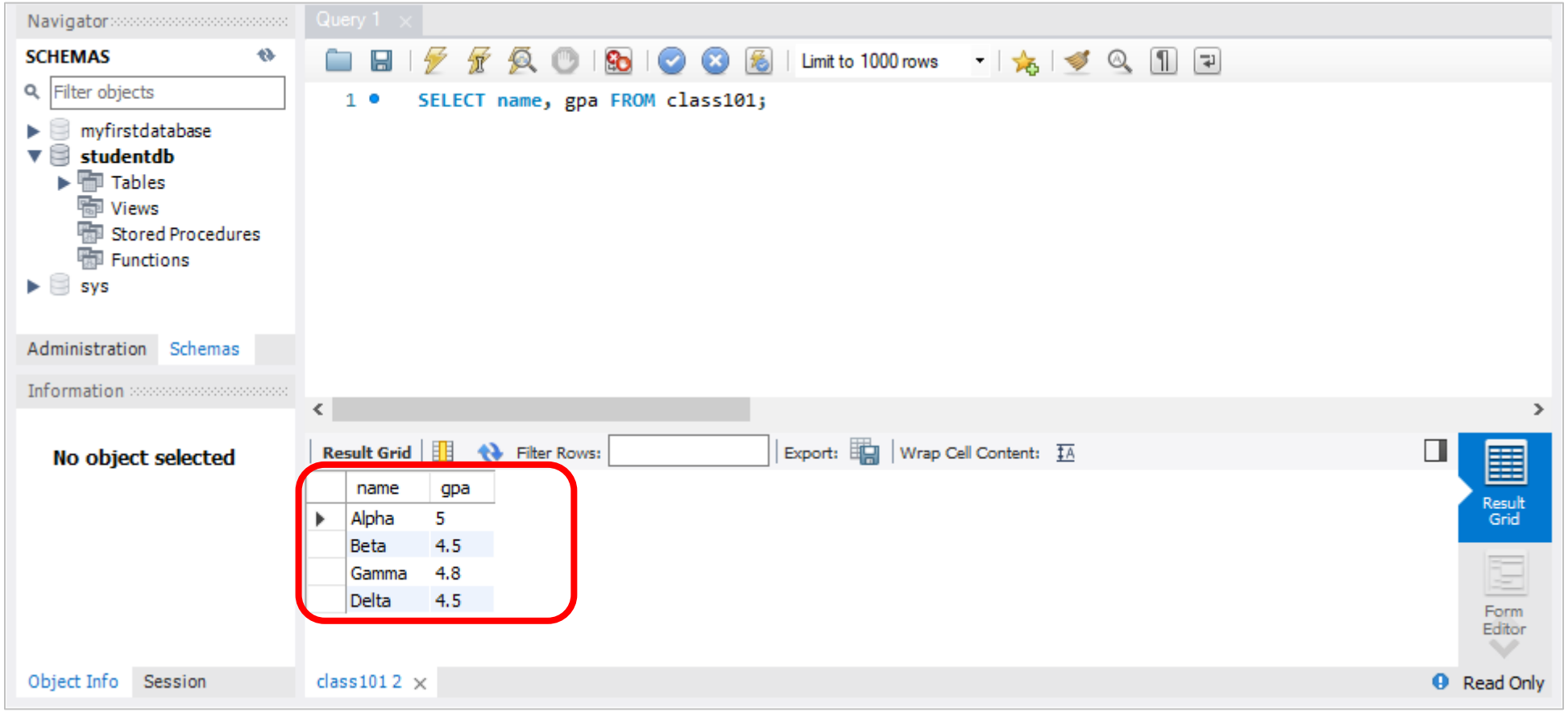
Suppose we have a database called studentdb, a table called class101 in the database with 3 columns (id, name, gpa) and 4 rows as illustrated below. Each column has a data type. We choose: INT (integer) for column id, VARCHAR(50) (variable-length string of up to 50 characters) for name, and FLOAT (floating-point number) for gpa. The content of table is as follows.



STRUCTURE QUERY LANGUAGE (SQL) Part I

SELECT

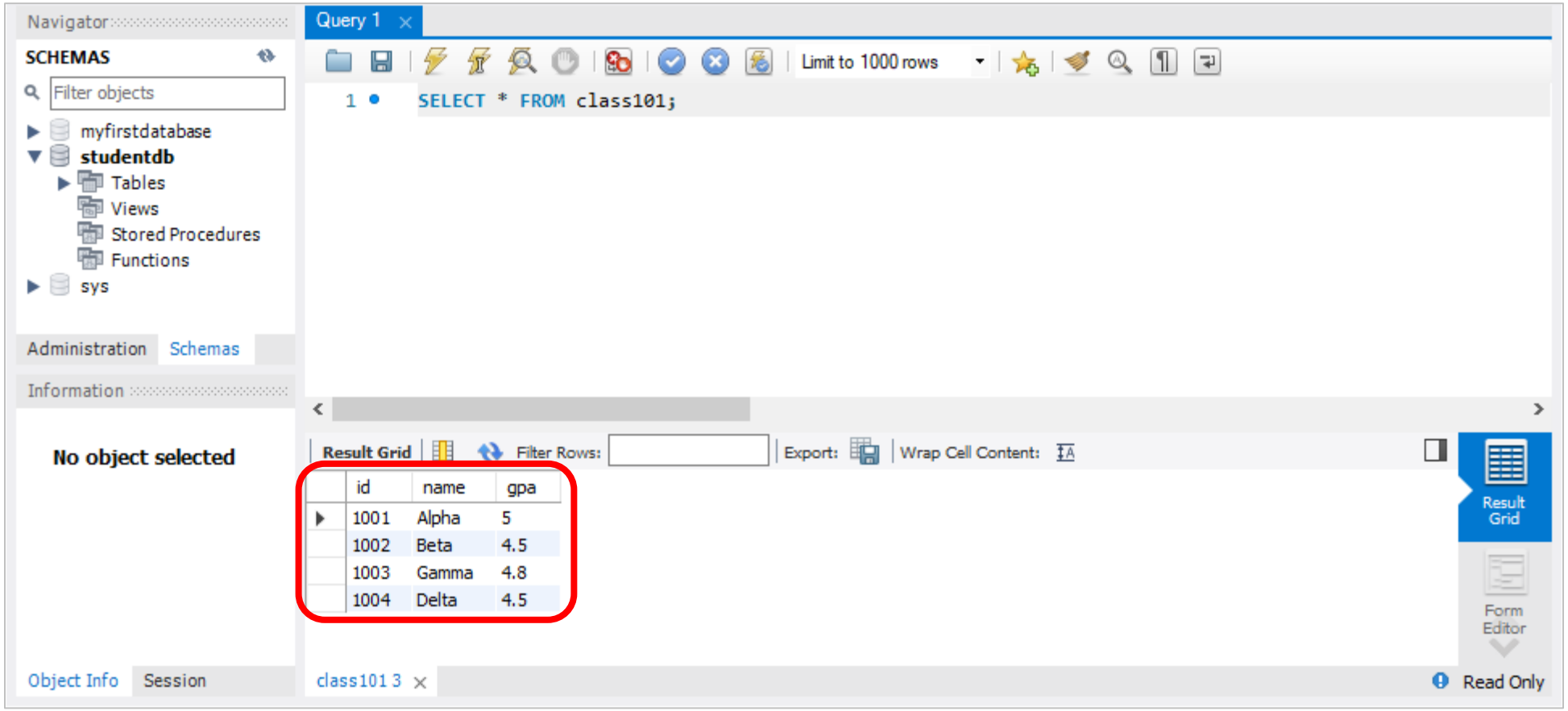
```
-- SYNTAX
SELECT column1, column2, ... FROM tableName WHERE criteria
SELECT * FROM tableName WHERE criteria

-- EXAMPLES 1
SELECT name, gpa FROM class101;
    -- Select columns name and gpa from table class101.
```



STRUCTURE QUERY LANGUAGE (SQL) Part I

SELECT

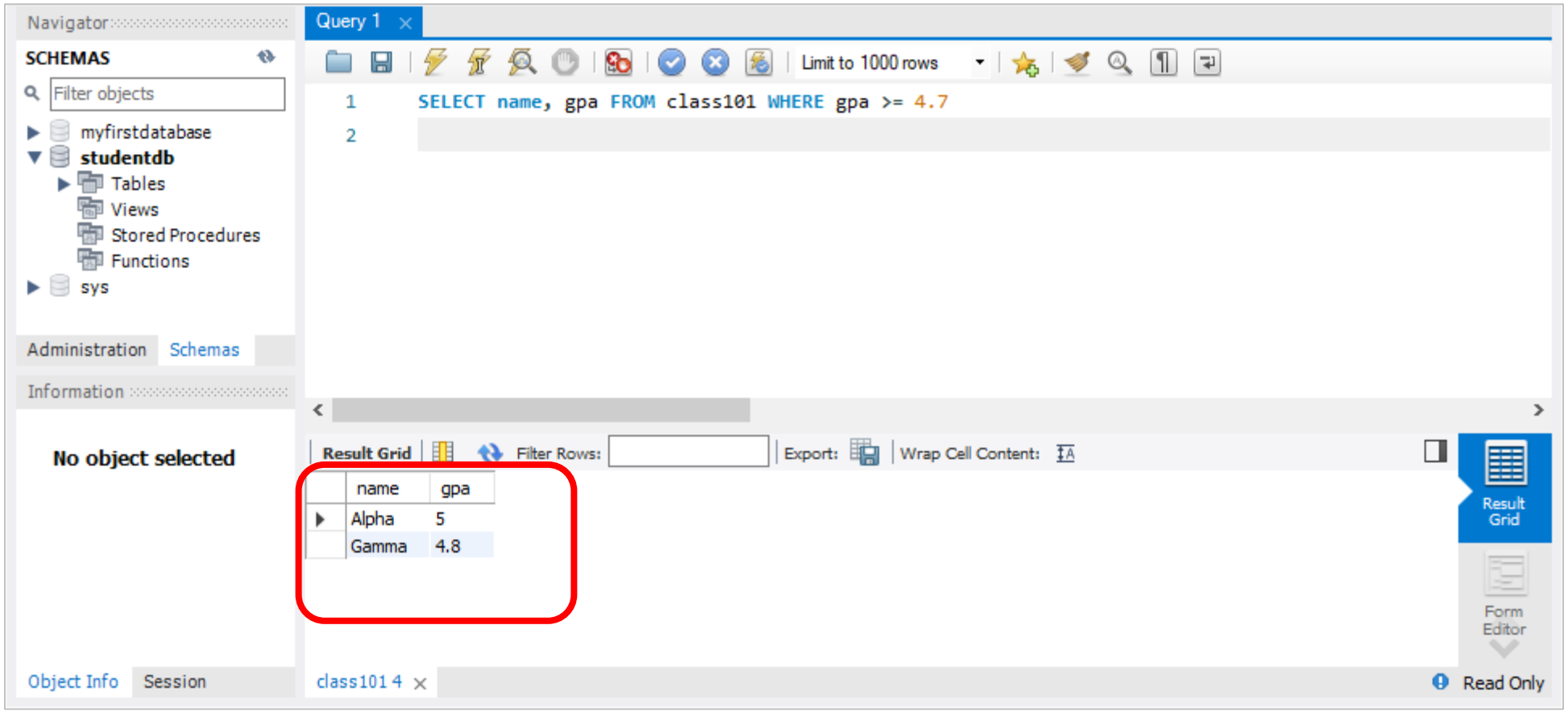
```
-- SYNTAX
SELECT column1, column2, ... FROM tableName WHERE criteria
SELECT * FROM tableName WHERE criteria

-- EXAMPLES 2
SELECT * FROM class101;
    -- Select ALL columns from table class101.
    -- The wildcard * denotes all the columns.
```



STRUCTURE QUERY LANGUAGE (SQL) Part I

SELECT

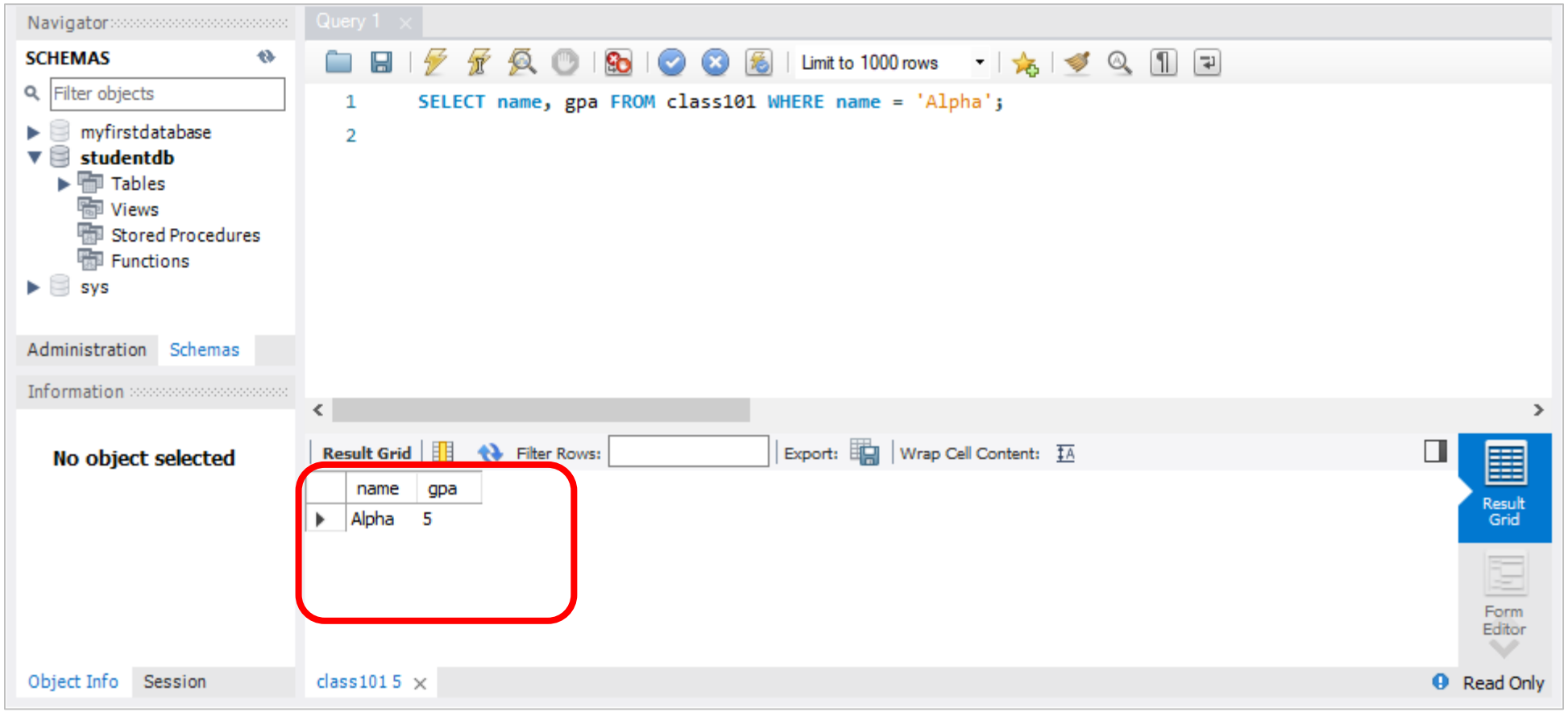
```
-- SYNTAX
SELECT column1, column2, ... FROM tableName WHERE criteria
SELECT * FROM tableName WHERE criteria

-- EXAMPLES 3
SELECT name, gpa FROM class101 WHERE gpa >= 4.7;
    -- Select columns name and gpa, where the rows meet the criteria.
    -- You can compare numbers (INT, FLOAT) using =, >, <, >=, <=, <> (!=)
```



STRUCTURE QUERY LANGUAGE (SQL) Part I

SELECT

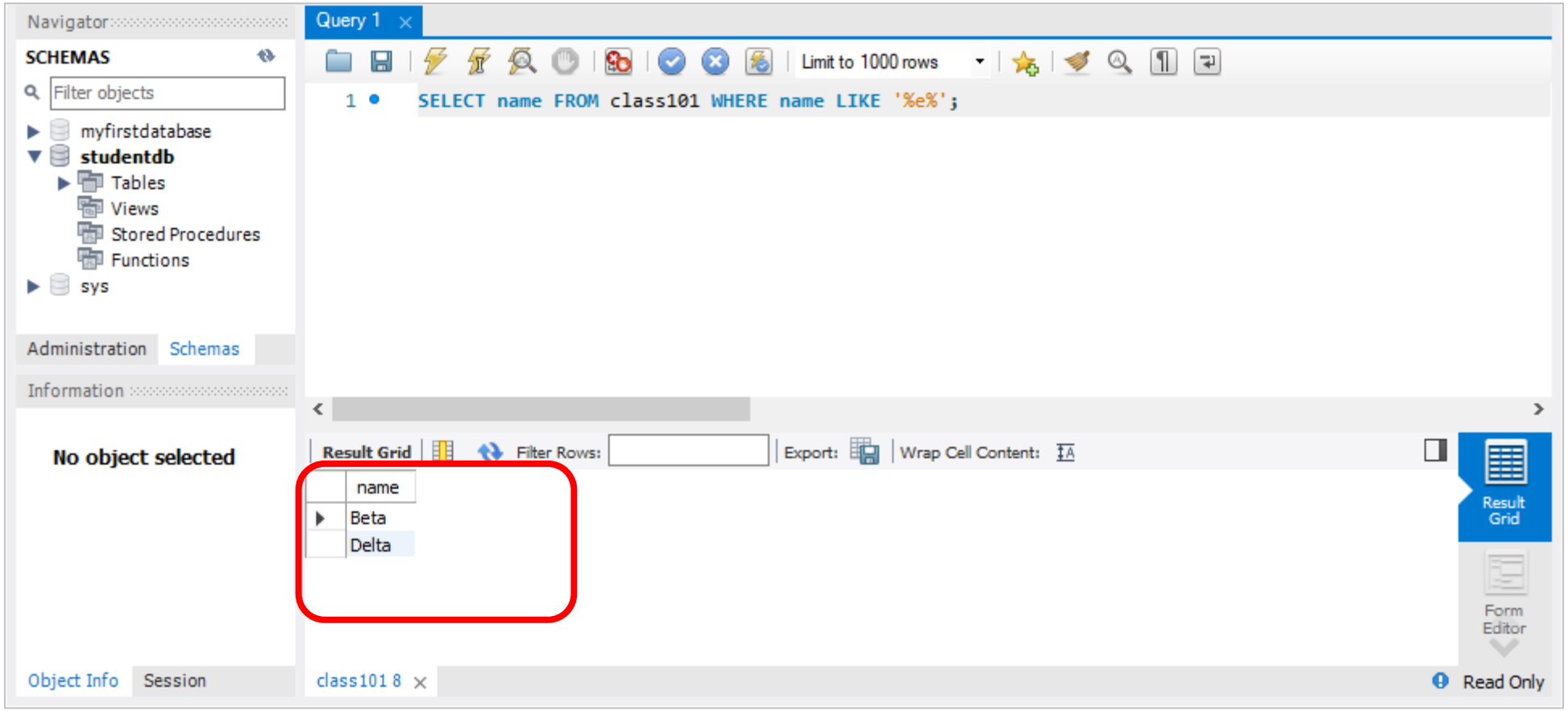
```
-- SYNTAX
SELECT column1, column2, ... FROM tableName WHERE criteria
SELECT * FROM tableName WHERE criteria

-- EXAMPLES 4
SELECT name, gpa FROM class101 WHERE name = 'Alpha';
    -- Full-match (= or !=) on string. Strings are enclosed in quotes.
```



STRUCTURE QUERY LANGUAGE (SQL) Part I

SELECT

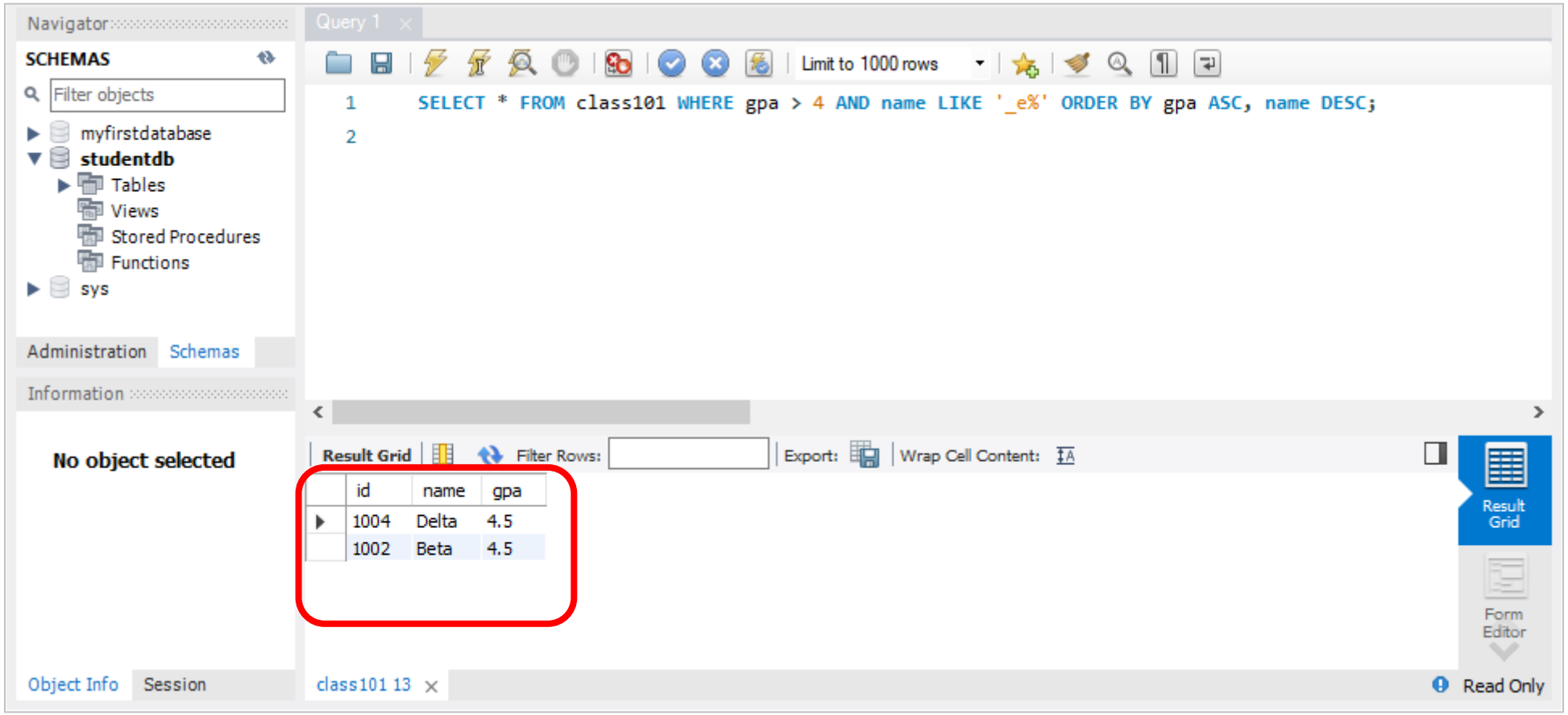
```
-- SYNTAX
SELECT column1, column2, ... FROM tableName WHERE criteria
SELECT * FROM tableName WHERE criteria

-- EXAMPLES 5
SELECT name FROM class101 WHERE name LIKE '%e%';
    -- Use "LIKE" for string pattern-matching, with
    -- wildcard % matches zero or more (any) characters;
    -- wildcard _ matches one (any) character.
```



STRUCTURE QUERY LANGUAGE (SQL) Part I

SELECT

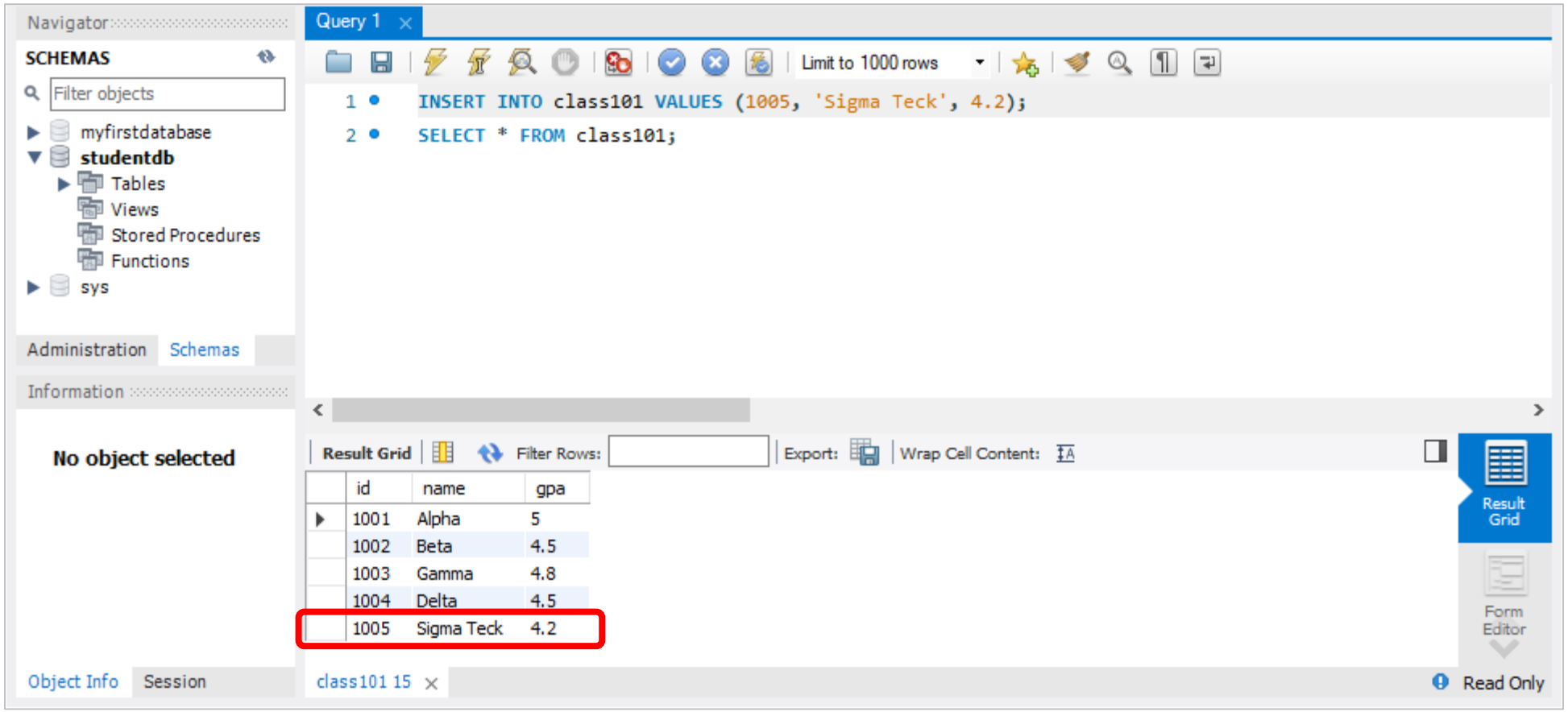
```
-- SYNTAX
SELECT column1, column2, ... FROM tableName WHERE criteria
SELECT * FROM tableName WHERE criteria

-- EXAMPLES 6
SELECT * FROM class101 WHERE gpa > 4 AND name LIKE '_e%' ORDER BY gpa ASC, name DESC;
    -- Use AND, OR, NOT to combine simple conditions.
    -- Order the results in DESC (descending) or ASC (Ascending)
```



STRUCTURE QUERY LANGUAGE (SQL) Part I

INSERT

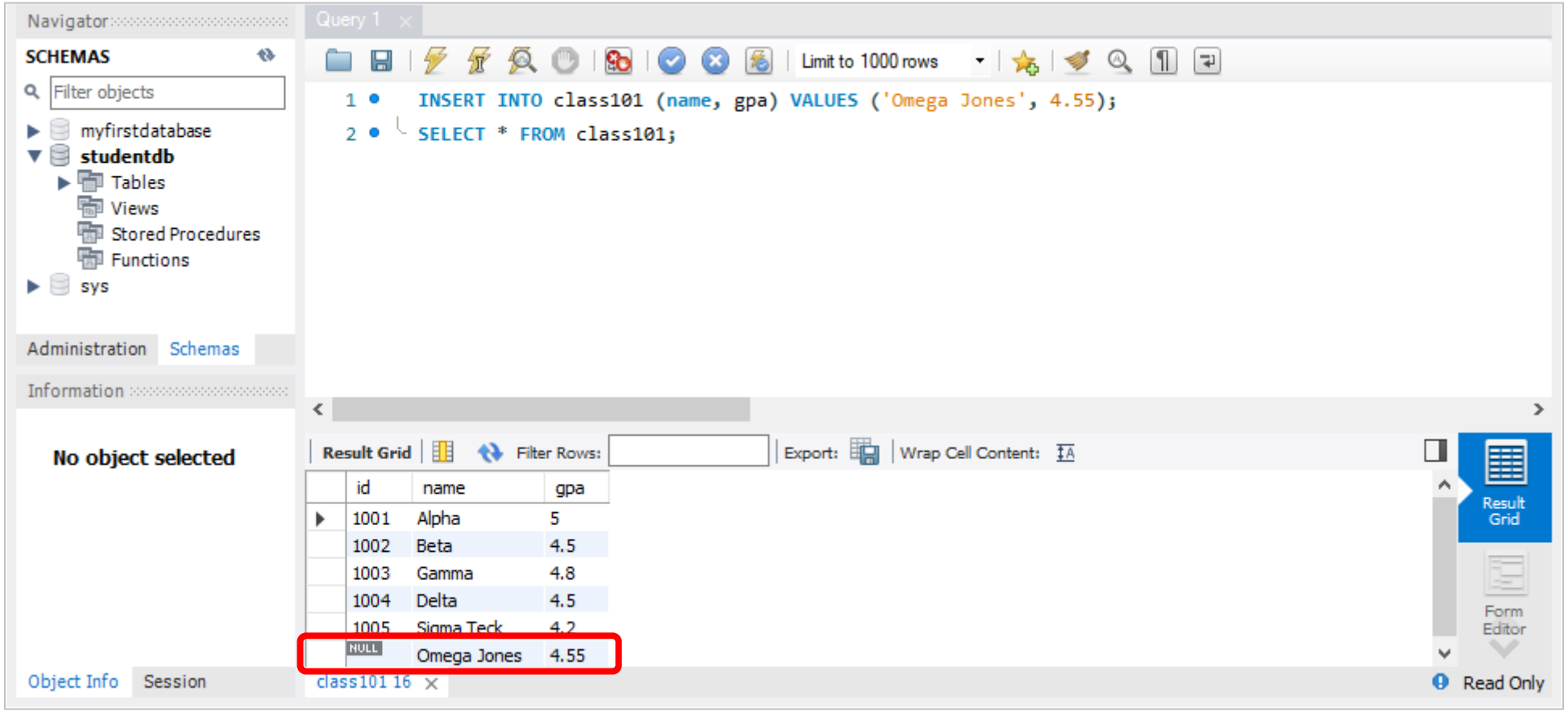
```
-- SYNTAX
INSERT INTO tableName VALUES (firstColumnValue, ..., lastColumnValue) -- All columns
INSERT INTO tableName (column1, column2, ...) VALUES (value1, value2, ...) -- Selected Columns

-- Example 1
INSERT INTO class101 VALUES (1005, 'Sigma Teck', 4.5);
    -- Insert values of all columns.
```



STRUCTURE QUERY LANGUAGE (SQL) Part I

INSERT

```
-- SYNTAX
INSERT INTO tableName VALUES (firstColumnValue, ..., lastColumnValue) -- All columns
INSERT INTO tableName (column1, column2, ...) VALUES (value1, value2, ...) -- Selected Columns

-- Example 2
INSERT INTO class101 (name, gpa) VALUES ('Omega Jones', 4.55);
    -- Missing fields will be set to their default values or NULL
```



STRUCTURE QUERY LANGUAGE (SQL) Part I

UPDATE

Note: **Safe update mode** is ON in default in MySQL Server.
You cannot update the data if this setting is ON.
You have two ways to turn off this setting.

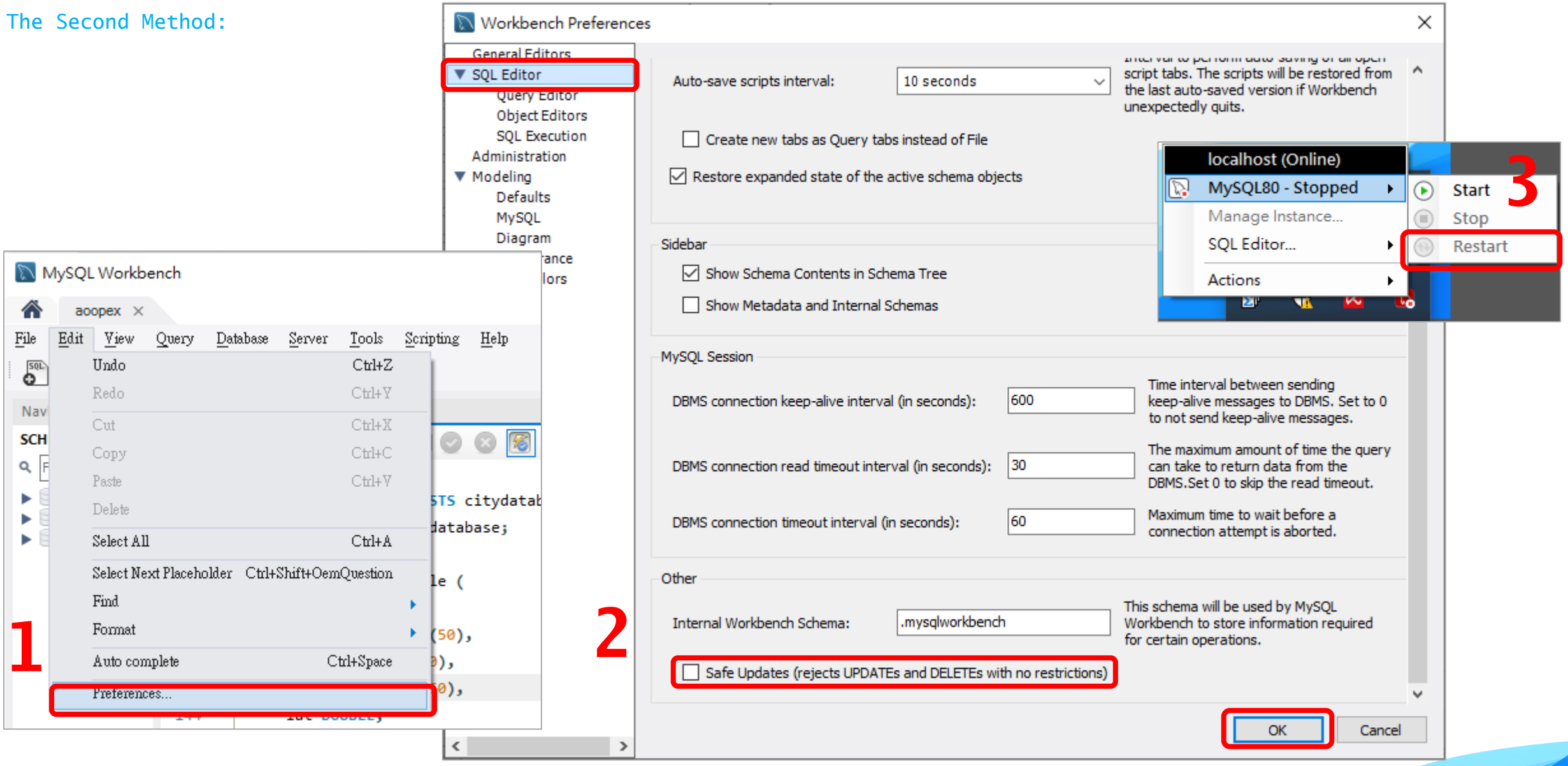
The First Method is to use following command:

```
SET SQL_SAFE_UPDATES=0;

-- Example
SET SQL_SAFE_UPDATES=0;
UPDATE class101 SET gpa = gpa + 1.0 WHERE name = 'Sigma Teck';
```

STRUCTURE QUERY LANGUAGE (SQL) Part I

UPDATE

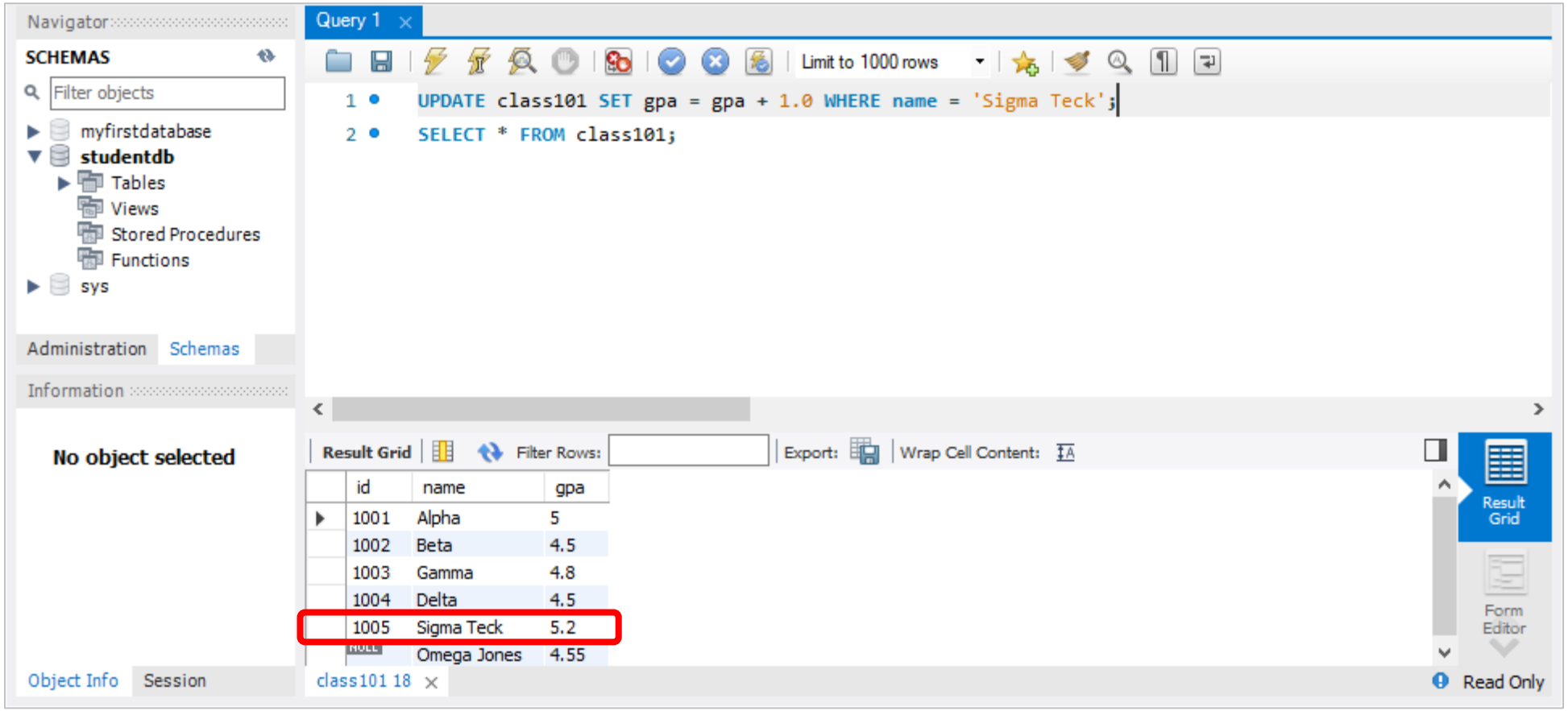
The Second Method:



STRUCTURE QUERY LANGUAGE (SQL) Part I

UPDATE

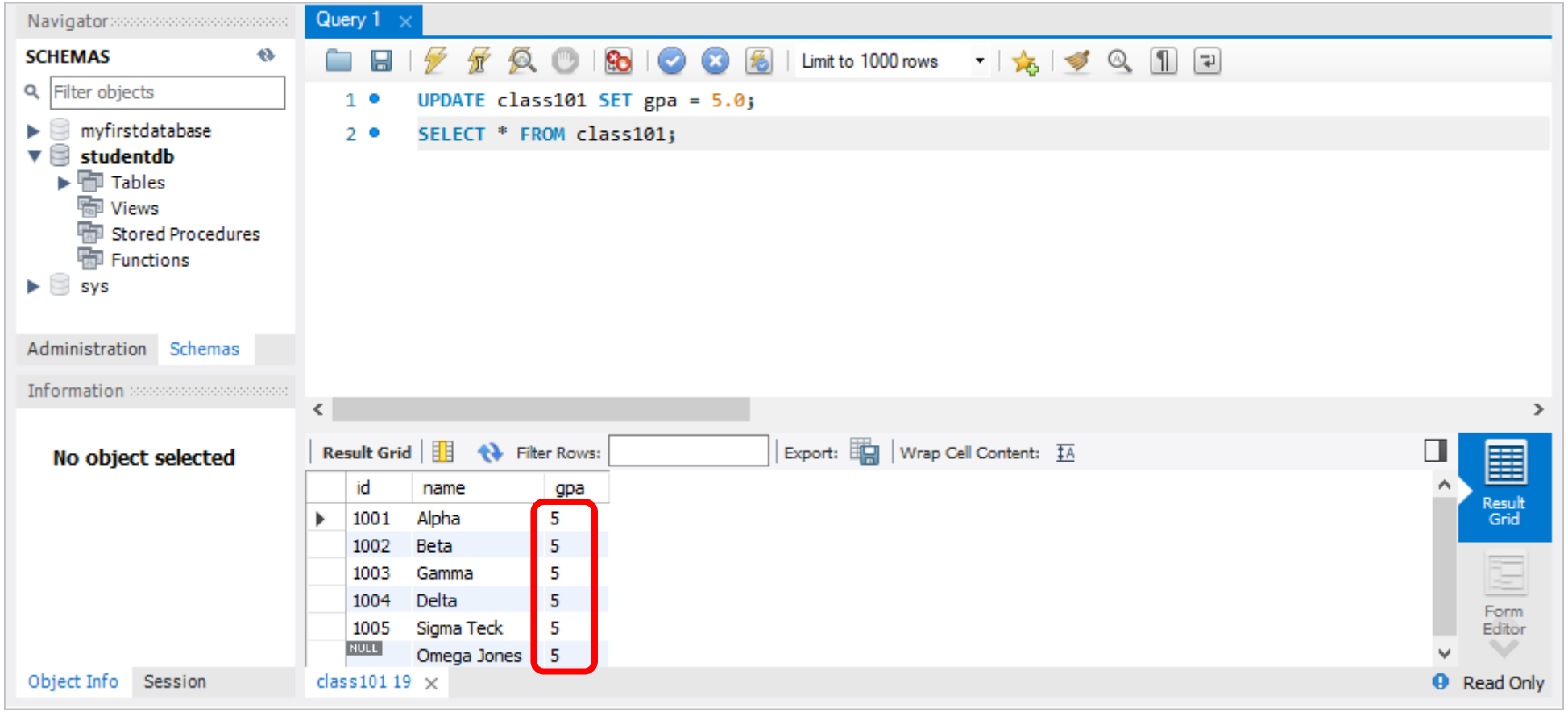
```
-- SYNTAX
UPDATE tableName SET column = value WHERE criteria

-- EXAMPLES 1
UPDATE class101 SET gpa = gpa + 1.0 WHERE name = 'Sigma Teck';
    -- Selected rows
```



STRUCTURE QUERY LANGUAGE (SQL) Part I

UPDATE

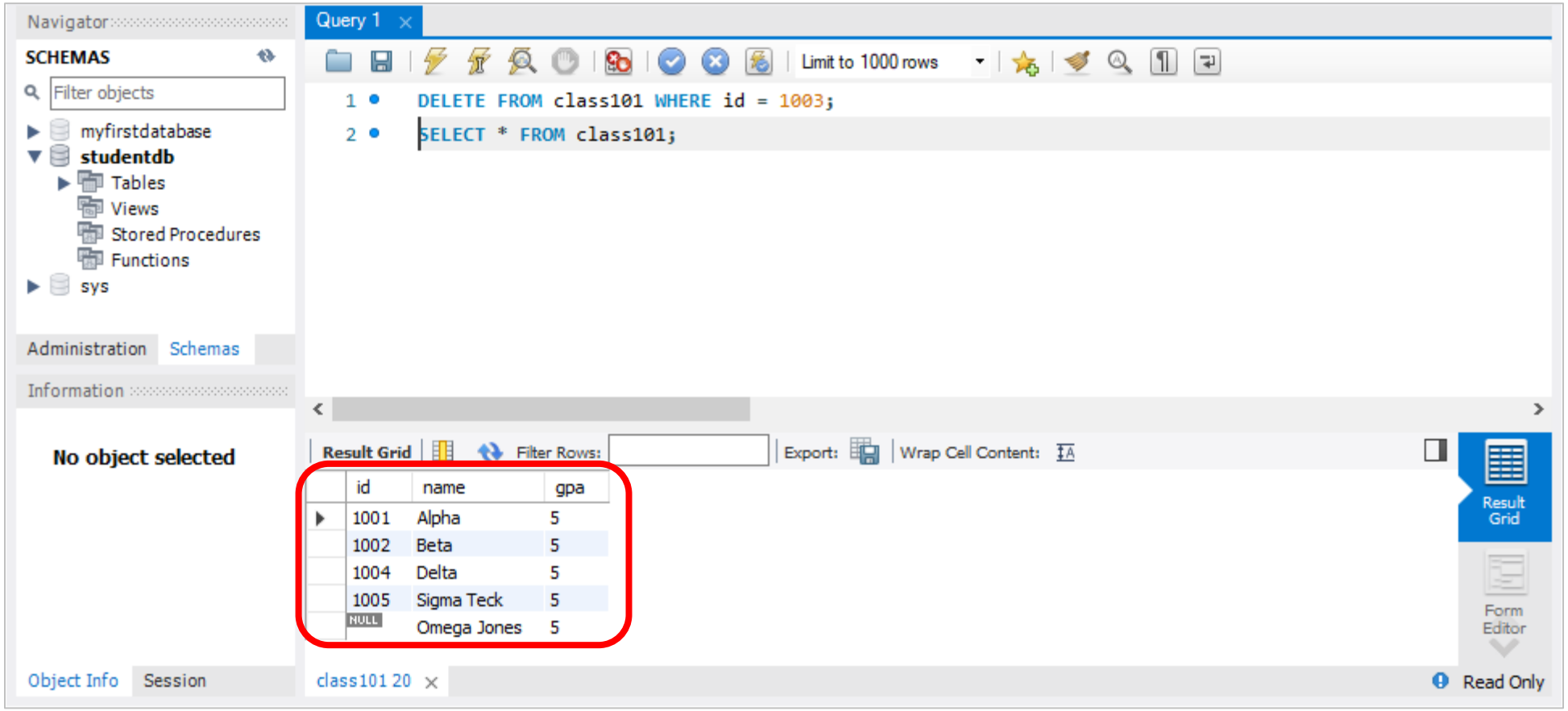
```
-- SYNTAX
UPDATE tableName SET column = value WHERE criteria

-- EXAMPLES 2
UPDATE class101 SET gpa = 5.0;
    -- ALL rows
```



STRUCTURE QUERY LANGUAGE (SQL) Part I

DELETE

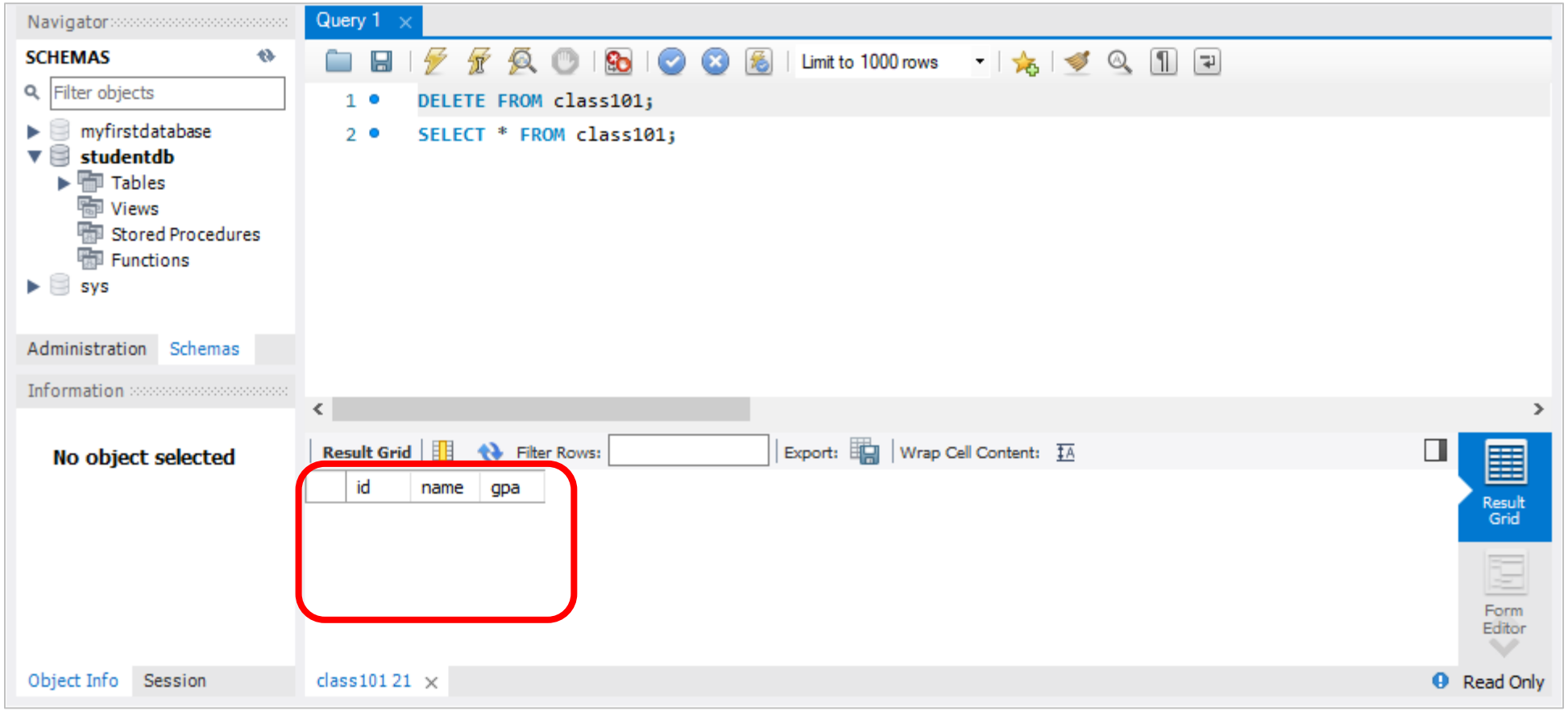
```
-- SYNTAX
DELETE FROM tableName WHERE criteria

-- EXAMPLES 1
DELETE FROM class101 WHERE id = 1003;
    -- Delete rows that meet the criteria.
```



STRUCTURE QUERY LANGUAGE (SQL) Part I

DELETE

```
-- SYNTAX
DELETE FROM tableName WHERE criteria

-- EXAMPLES 2
DELETE FROM class101;
    -- Delete ALL rows from the table class101!
    -- Beware that there is NO UNDO!
```



Structure Query Language (SQL)

Frequently-used MySQL Commands Part II

STRUCTURE QUERY LANGUAGE (SQL) Part II

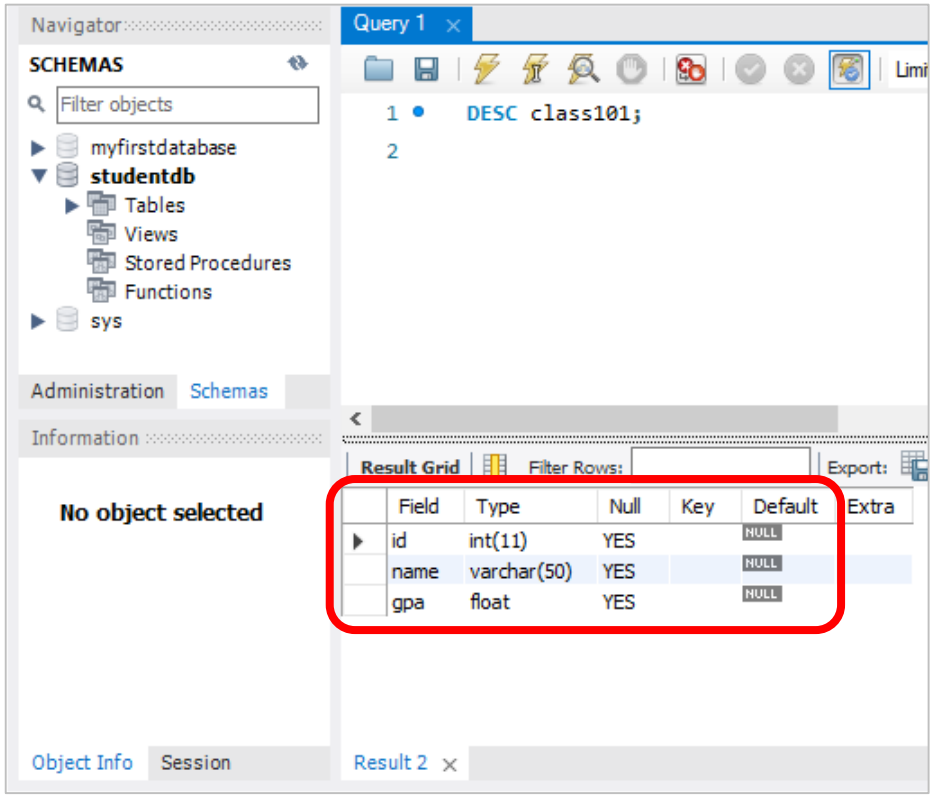
Import CSV file into MySQL table.

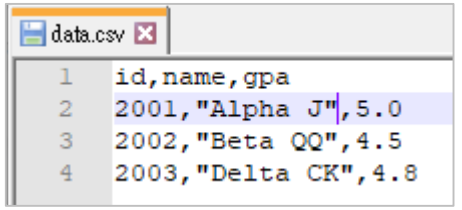
The LOAD DATA INFILE statement allows you to read data from a text file and import the file's data into a database table very fast.

Before importing the file, you need to prepare the following:

- A database table to which the data from the file will be imported.
- A CSV file with data that matches with the number of columns of the table and the type of data in each column.
- The account, which connects to the MySQL database server, has FILE and INSERT privileges.

Suppose we have a table named class101 with the following structure, and a data.csv file contains the first line as column headings and other three lines of data.





STRUCTURE QUERY LANGUAGE (SQL) Part II

Other commands

```
-- SYNTAX
LOAD DATA INFILE filePath
INTO TABLE tableName
FIELDS TERMINATED BY terminatedCharacter
ENCLOSED BY enclosedCharacter
LINES TERMINATED BY lineTerminatedCharacter
IGNORE n ROWS;

-- EXAMPLES
LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads'
INTO TABLE class101
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\r\n'
IGNORE 1 ROWS;
```

Note:

1. You have to use '/', '//', or '\\' in filePath, '\' is escaped character.
2. "\r" is carriage return and "\n" is newline. You have to check your termination symbol in you data file firstly, and select corresponding terminated symbol.

"\r\n"

```
1  ID,Country,City,Latitude,LongitudeCRLF
2  100,"Taipei","Zhongzheng",121.5198839,25.03240487CRLF
3  130,"Taipei","Datong",121.5130417,25.06342433CRLF
```

"\n"

```
1  _id,nowfloor,destination,numLF
2  00001-01,1,7,4LF
3  00001-02,2,5,1LF
```

3. "C:/ProgramData/MySQL/MySQL Server 8.0/Uploads" is default secure file path, you only can put the file which you want to load in this folder. If you put data in other path, you will meet the error shown in next page, and you can follow next page to change the default secure file path.

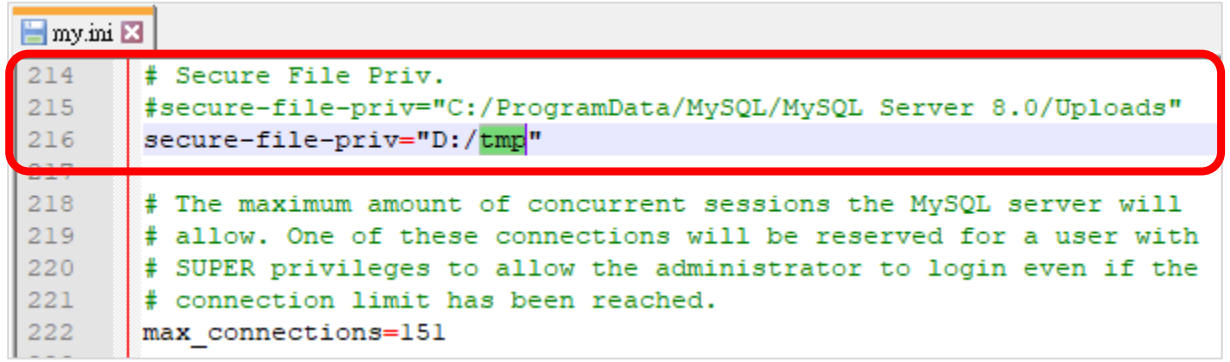
STRUCTURE QUERY LANGUAGE (SQL) Part II

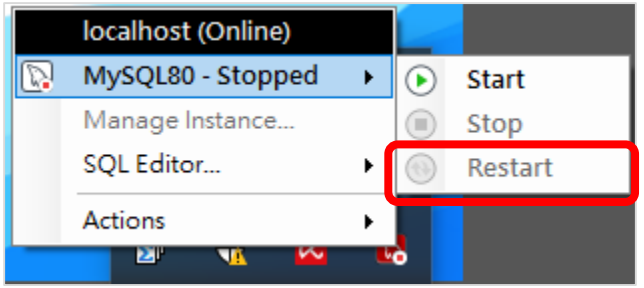
Import CSV file into MySQL table.

If you meet

Error Code: 1290. The MySQL server is running with the --secure-file-priv option so it cannot execute this statement

follow the steps to solve it.





STRUCTURE QUERY LANGUAGE (SQL) Part II

DISTINCT

```
-- SYNTAX
SELECT DISTINCT column1, column2, ... FROM tableName

-- EXAMPLES 1
SELECT DISTINCT Company FROM Orders;
    -- Select different Company name in table Orders.

-- EXAMPLES 2
    -- https://www.fooish.com/sql/distinct.html

-- EXAMPLES 3
SELECT COUNT(DISTINCT column_name) FROM table_name
    -- https://www.w3schools.com/sql/sql_distinct.asp
```

STRUCTURE QUERY LANGUAGE (SQL) Part II

SQL FUNCTIONS

Please refer to Functions Part on w3school:
https://www.w3schools.com/sql/sql_ref_mysql.asp

MySQL has many built-in functions.
This reference contains string, numeric, date,
and some advanced functions in MySQL.

Structure Query Language (SQL)

Frequently-used MySQL Commands Part III

STRUCTURE QUERY LANGUAGE (SQL) Part III

THE OTHERS

Please refer to Functions Part on w3school:
https://www.w3school.com.cn/sql/sql_top.asp

1. IN: https://www.w3school.com.cn/sql/sql_in.asp
2. BETWEEN: https://www.w3school.com.cn/sql/sql_between.asp
3. ALIASES: https://www.w3school.com.cn/sql/sql_alias.asp
4. UNION: https://www.w3school.com.cn/sql/sql_union.asp

STRUCTURE QUERY LANGUAGE (SQL) Part III

LIKE/REGEXP

```
-- MySQL LIKE SYNTAX
SELECT column1, column2, … FROM tableName WHERE columnName LIKE pattern

-- EXAMPLES
https://www.w3schools.com/sql/sql_like.asp


-- MySQL REGEXP SYNTAX
SELECT column1, column2, … FROM tableName WHERE columnName REGEXP pattern

-- EXAMPLES
https://www.runoob.com/mysql/mysql-regexp.html
```

STRUCTURE QUERY LANGUAGE (SQL) Part III

LIMIT

```
-- MySQL SYNTAX
[Other Statement, EX: SELECT, UPDATE,…] LIMIT {[offset,] row_count | row_count OFFSET offset}


-- EXAMPLES 1
SELECT * FROM myTable LIMIT 10;
    -- Select the first 10 data in default order.

-- EXAMPLES 2
SELECT * FROM myTable LIMIT 1, 100;
    -- Select data from the 2nd to (2nd + 99).

-- EXAMPLES 3
SELECT * FROM myTable LIMIT 100 OFFSET 1
    -- The same as EXAMPLES 2.
```

Note:
LIMIT is the last thing that happens as the query is executed. Please compare following two examples: (Suppose there are 100 data in table company.)

```
SELECT AVG(salary) FROM company ORDER BY salary DESC LIMIT 5;

SELECT AVG(t.salary) FROM
    (SELECT salary FROM company ORDER BY salary DESC LIMIT 5) AS t;
```

STRUCTURE QUERY LANGUAGE (SQL) Part III

SUBQUERIES

- A subquery is a SQL query nested inside a larger query.

- A subquery may occur in:
  - A SELECT clause
  - A FROM clause
  - A WHERE clause

- In MySQL subquery can be nested inside a SELECT, INSERT, UPDATE, DELETE, SET, or DO statement or inside another subquery.
- A subquery is usually added within the WHERE Clause of another SQL SELECT statement.
- You can use the comparison operators, such as >, <, or =. The comparison operator can also be a multiple-row operator, such as IN, ANY, SOME, or ALL.
- A subquery can be treated as an inner query, which is a SQL query placed as a part of another query called as outer query.
- The inner query executes first before its parent query so that the results of the inner query can be passed to the outer query.

- Reference: https://www.w3resource.com/mysql/subqueries/index.php

STRUCTURE QUERY LANGUAGE (SQL) Part III

SUBQUERIES

- Some important guidelines you have to know when using subqueries :
  - A subquery must be enclosed in parentheses.
  - Use single-row operators with single-row subqueries,
    and use multiple-row operators with multiple-row subqueries.
  - If a subquery (inner query) returns a null value to the outer query,
    the outer query will not return any rows when using certain comparison operators in a WHERE clause.
  - Every derived table must have its own alias. Every table in a FROM clause must have a name,
    therefore the [AS] name clause is mandatory. Any columns in the subquery select list must have unique names.
    EX: select count(*) from (select * from ……) as total;

```
DELETE FROM t1 WHERE s11 > ANY
    (SELECT COUNT(*) FROM t2 WHERE NOT EXISTS
        (SELECT * FROM t3 WHERE ROW(5*t2.s1,77) =
            (SELECT 50,11*s1 FROM t4 UNION
             SELECT 50,77 FROM
                 (SELECT * FROM t5))));
```

STRUCTURE QUERY LANGUAGE (SQL) Part III

SUBQUERIES

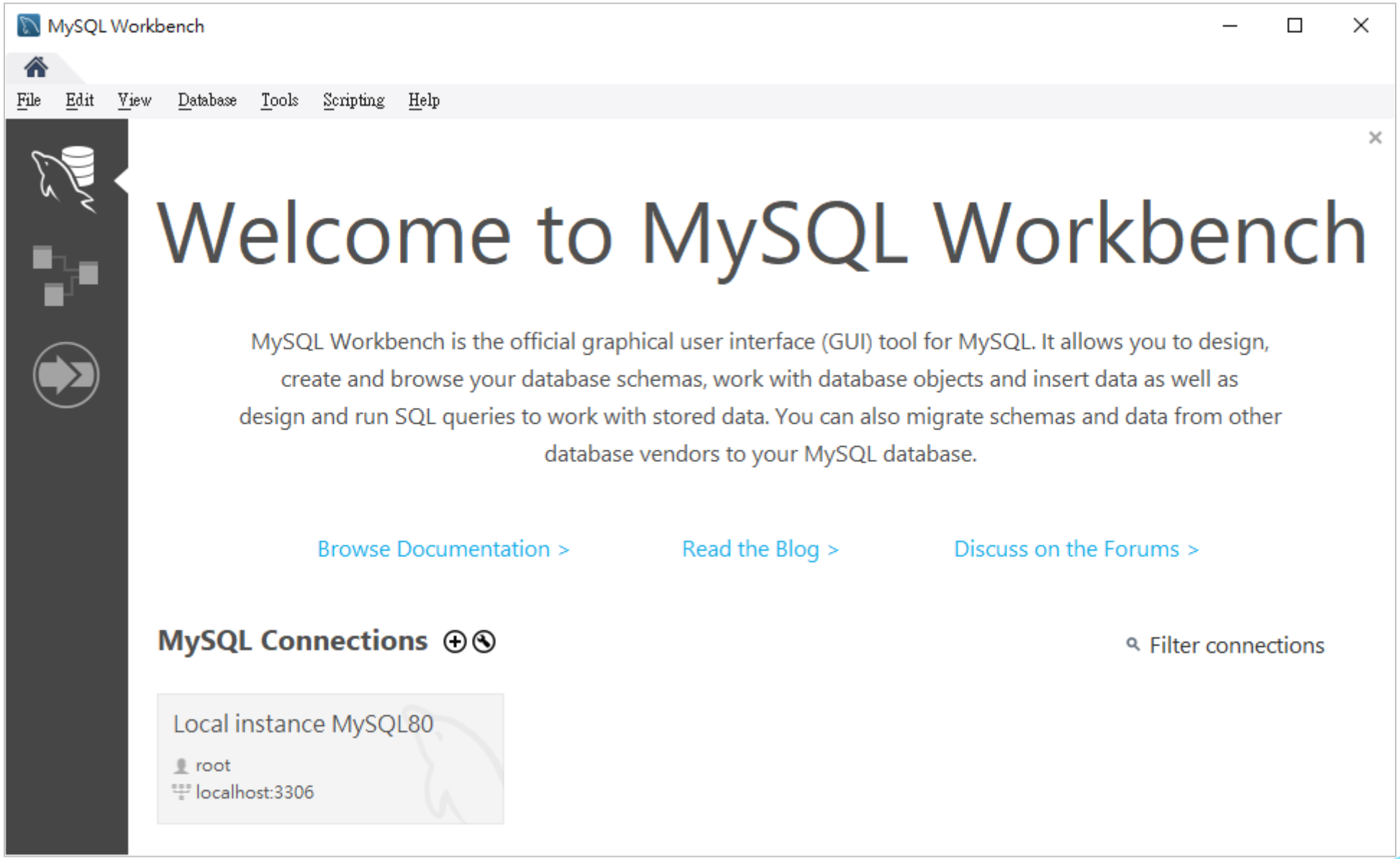
```
-- EXAMPLES 1
SELECT first_name,last_name, salary FROM emp_details WHERE salary >
    (SELECT salary FROM emp_details WHERE first_name='Alexander') AS tb;

-- EXAMPLES 2
SELECT employee_id,first_name,last_name,salary FROM employees WHERE salary >
    (SELECT AVG(SALARY) FROM employees) AS tb;

-- EXAMPLES 3
SELECT sc1, sc2, sc3 FROM
    (SELECT c1 AS sc1, c2 AS sc2, c3*3 AS sc3 FROM tb1) AS sb
WHERE sc1 > 1;

SELECT sb.c1, sb.c2, sc3 FROM
    (SELECT c1 AS sc1, c2 AS sc2, c3*3 AS sc3 FROM tb1) AS sb
WHERE sc1 > 1;
```

QT with MySQL (MySQL Part)

Example of QT connecting to MySQL

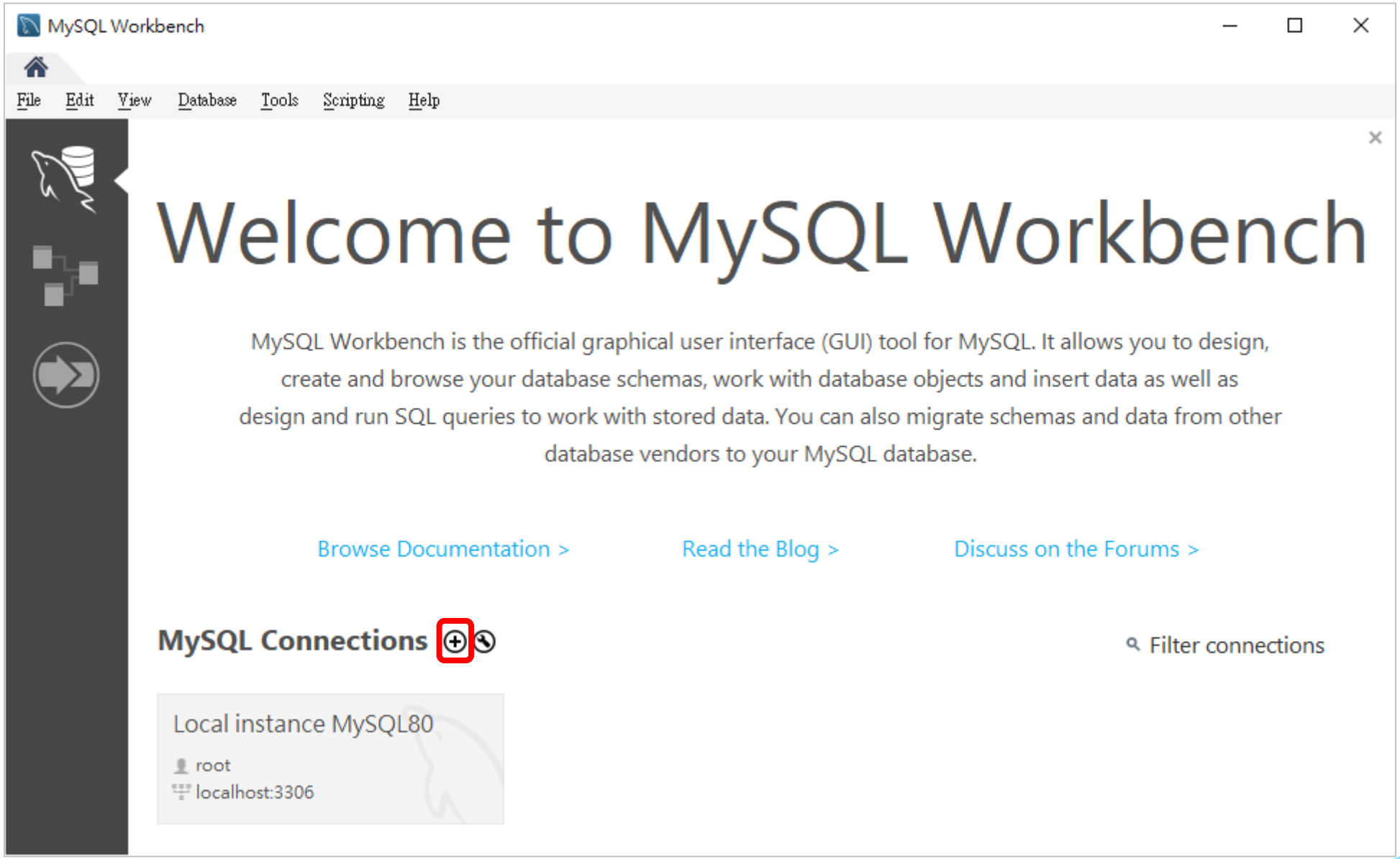
QT WITH MySQL (MySQL PART)

Execute “MySQL Workbench”.



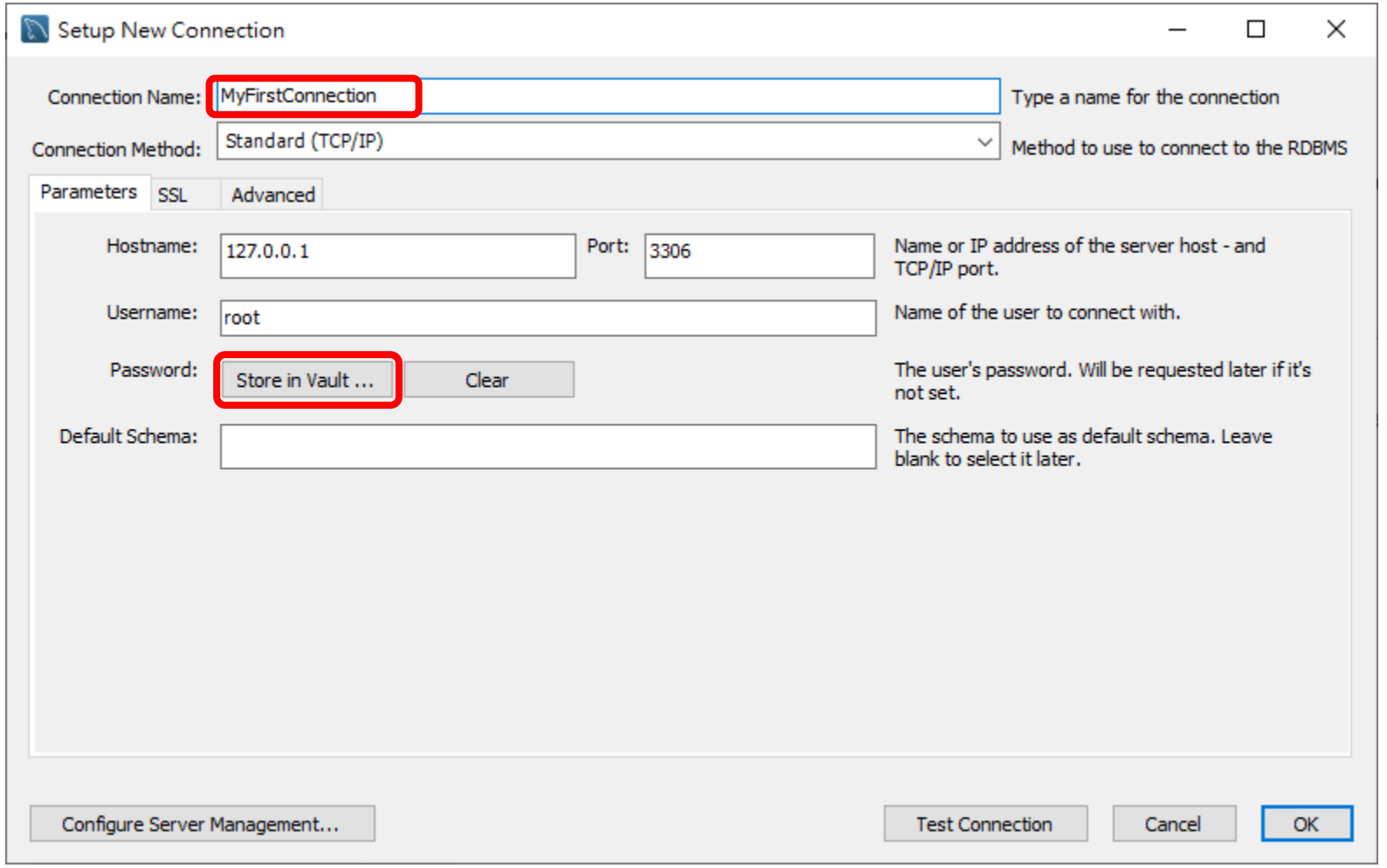
QT WITH MySQL (MySQL PART)

Click “+” button.



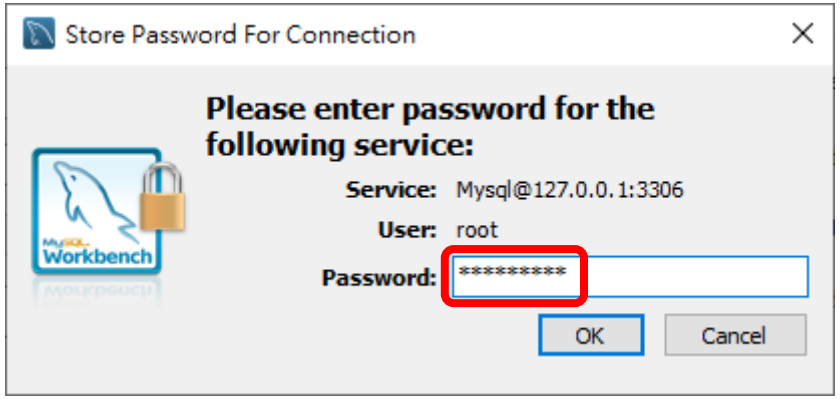
QT WITH MySQL (MySQL PART)

Key in “Connection Name” and then click “Store in Vault …”.



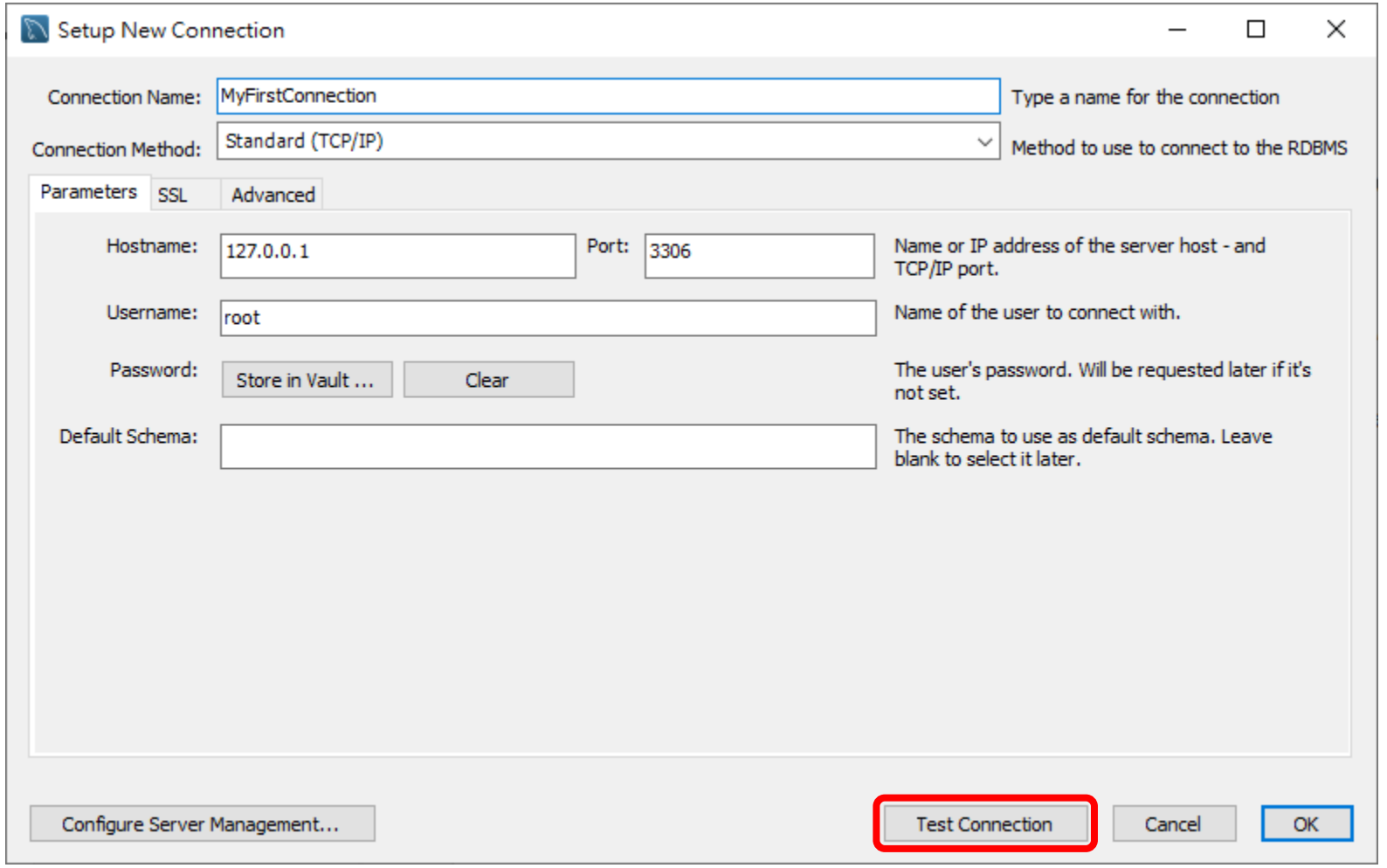
QT WITH MySQL (MySQL PART)

Key in “Password”, ex: 123456789.



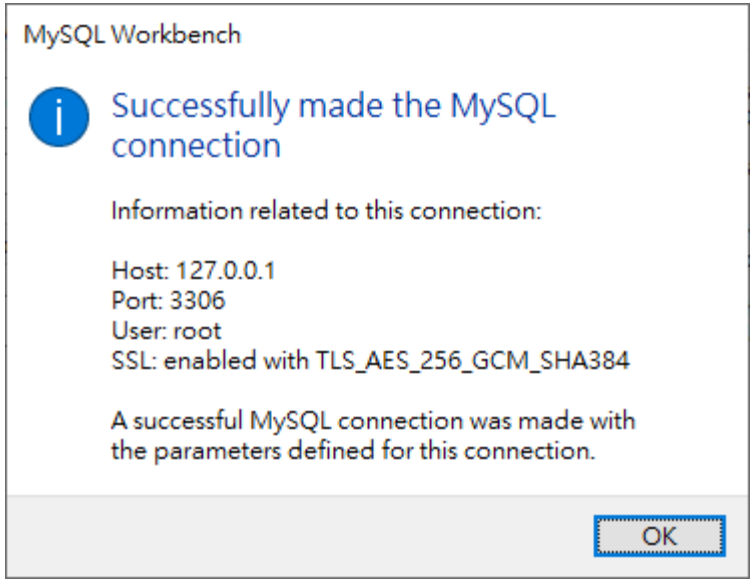
QT WITH MySQL (MySQL PART)

You can click “Test Connection” to check if it can make the MySQL connection successfully.



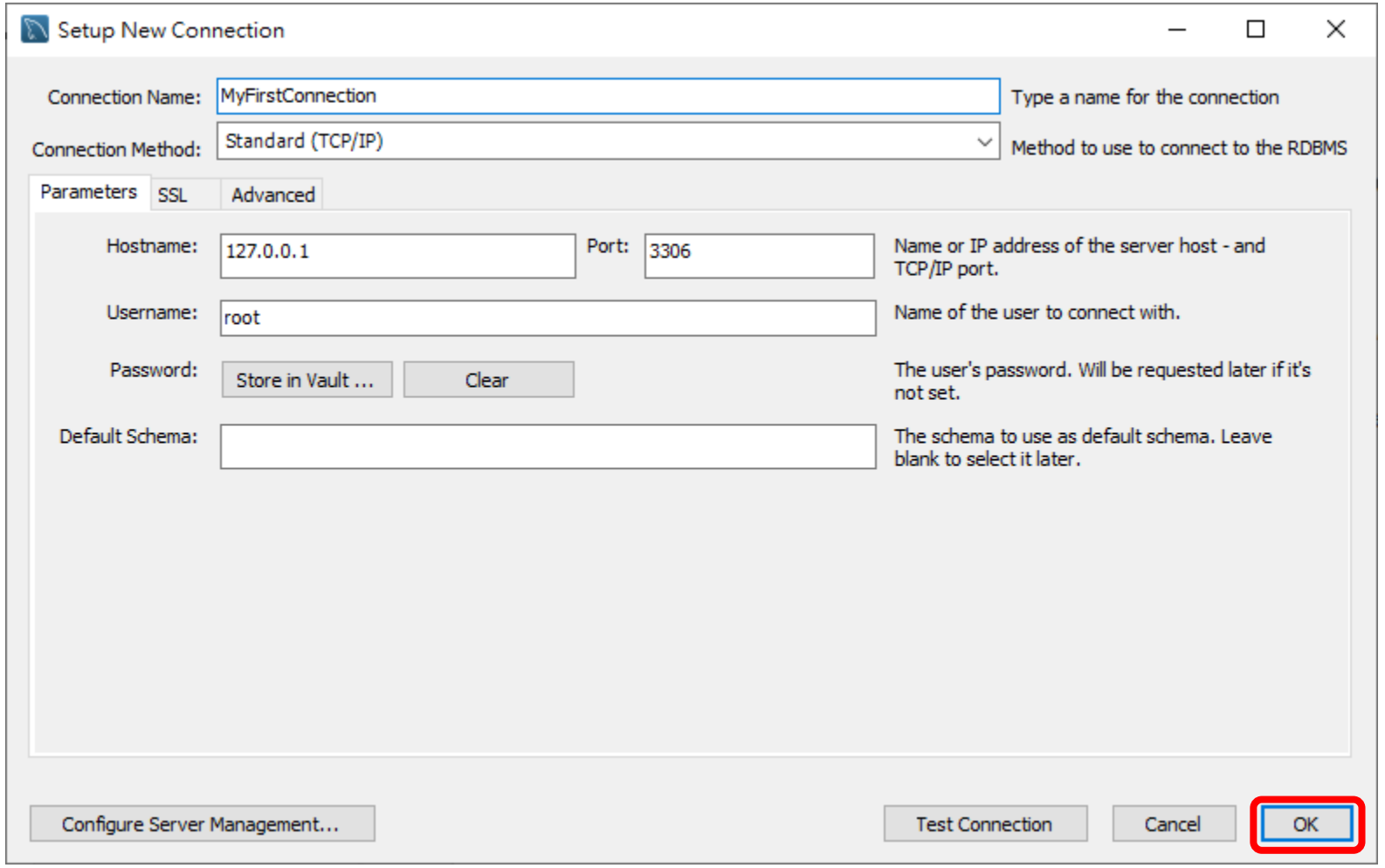
QT WITH MySQL (MySQL PART)

If connection is successful, the following result should be shown.



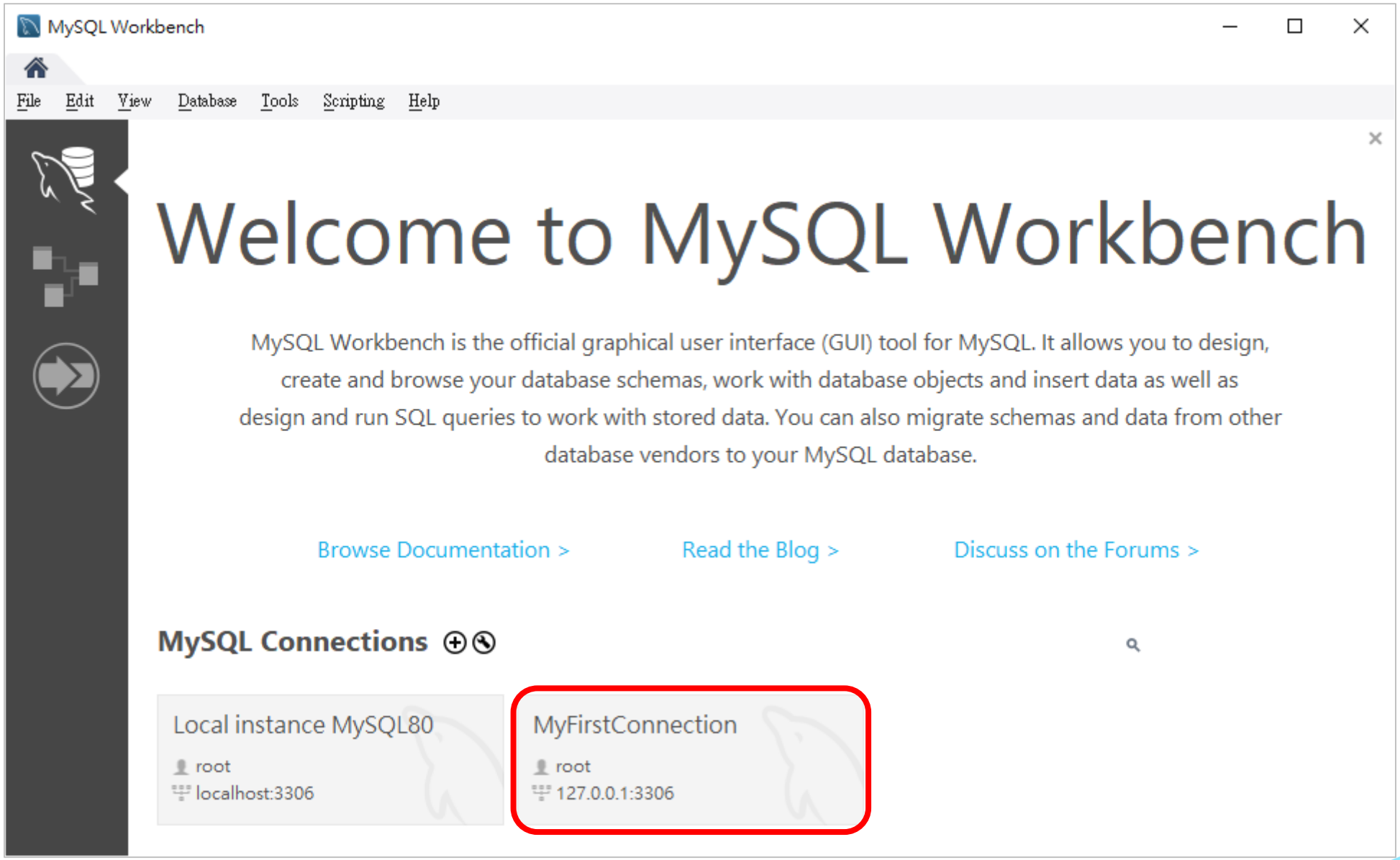
QT WITH MySQL (MySQL PART)

Click “OK”.



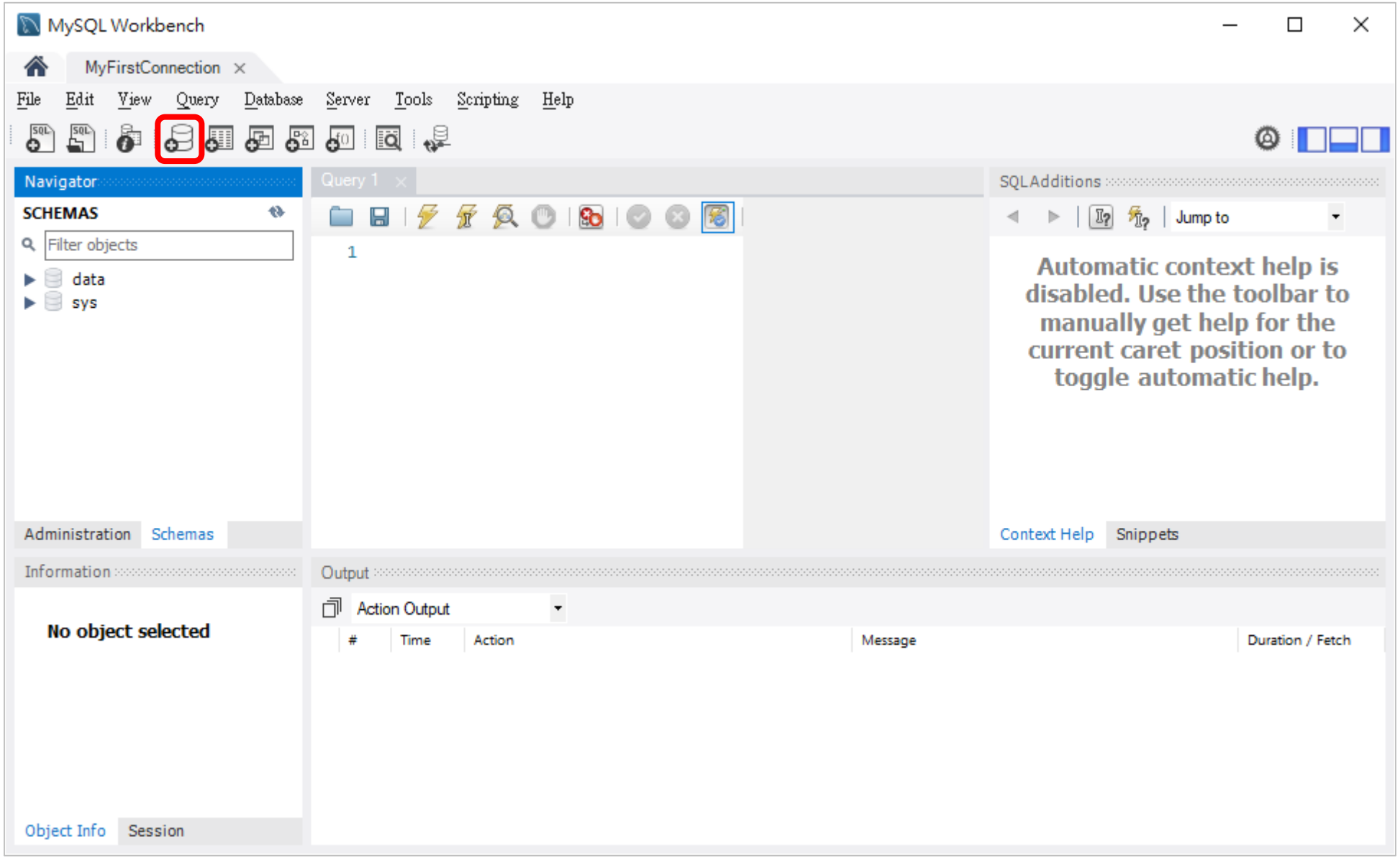
QT WITH MySQL (MySQL PART)

A new connection will be shown as below, and then click it.



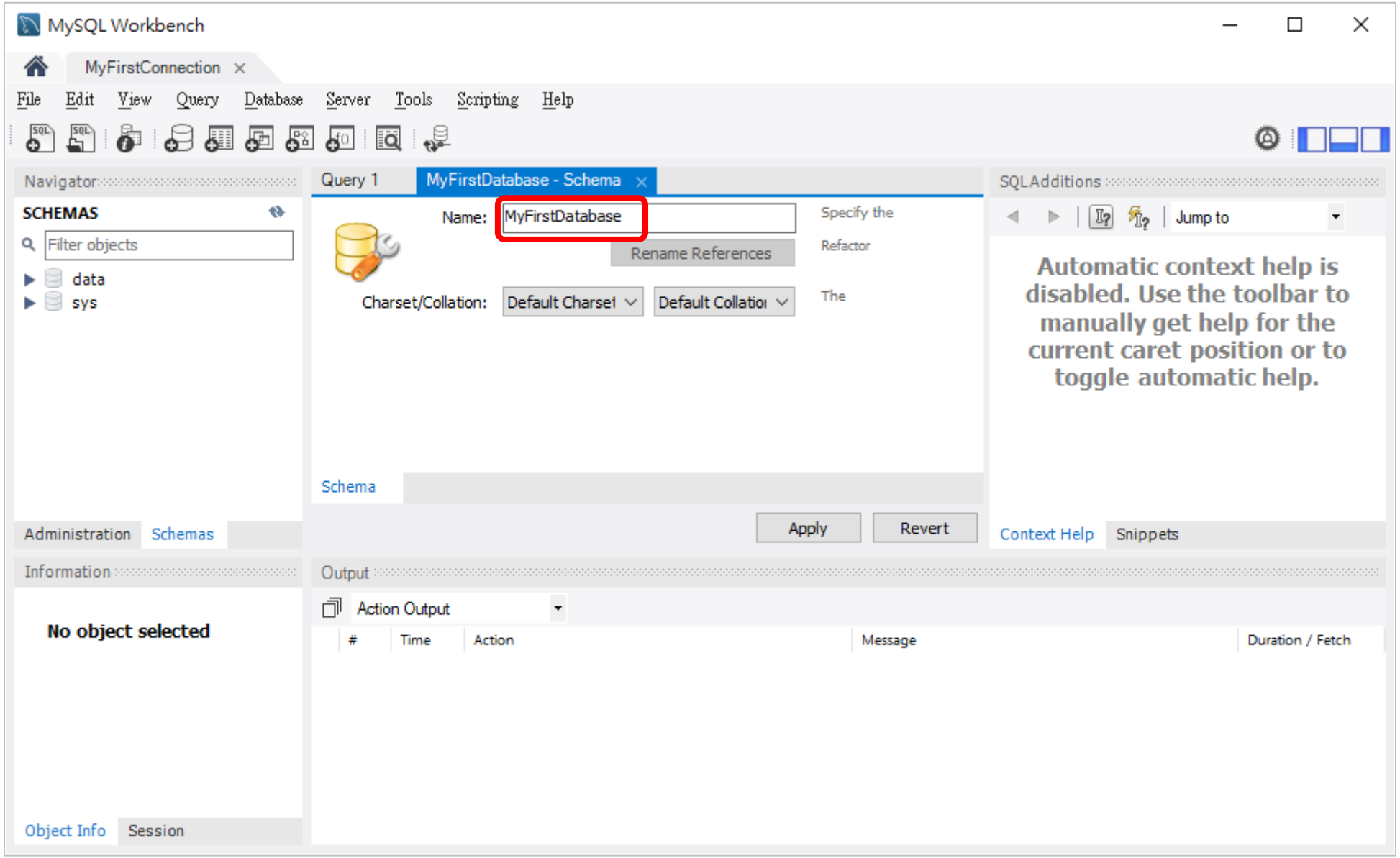
QT WITH MySQL (MySQL PART)

Click the button “Create a new schema in the connected server” to create a new schema.



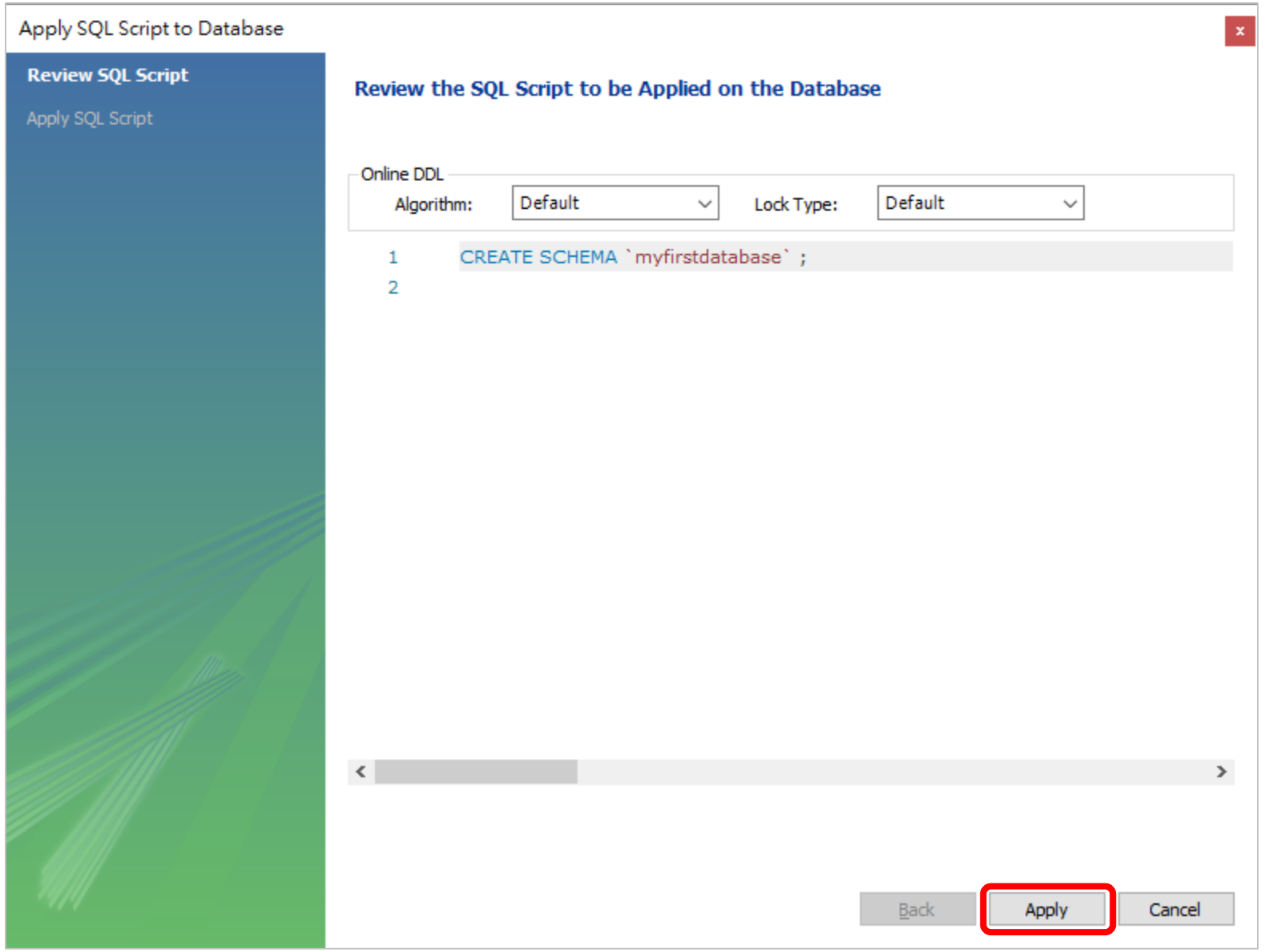
QT WITH MySQL (MySQL PART)

Key in the database name, and then click “Apply”.



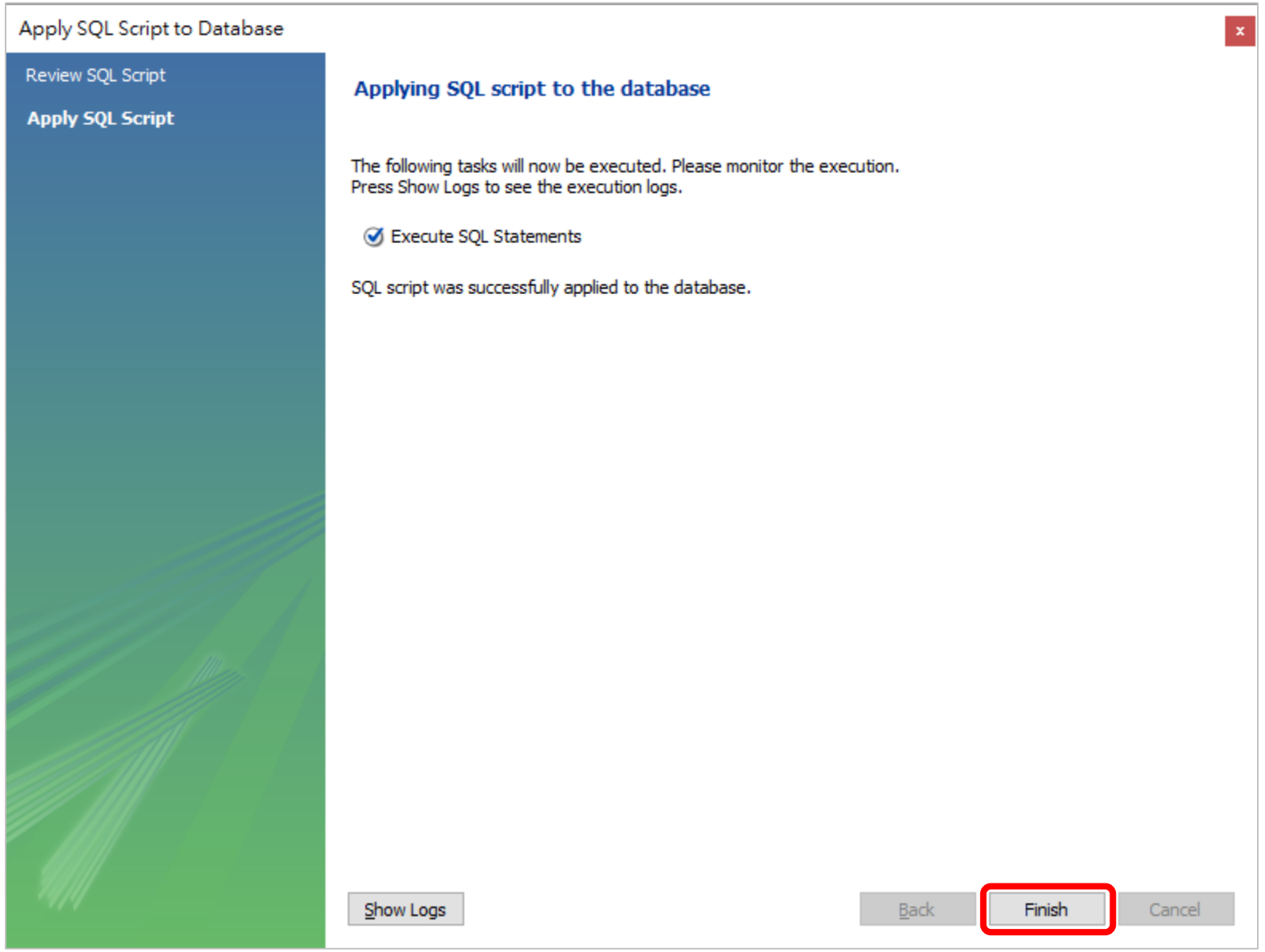
QT WITH MySQL (MySQL PART)

Following SQL script will be shown, and you can modify the command if necessary.



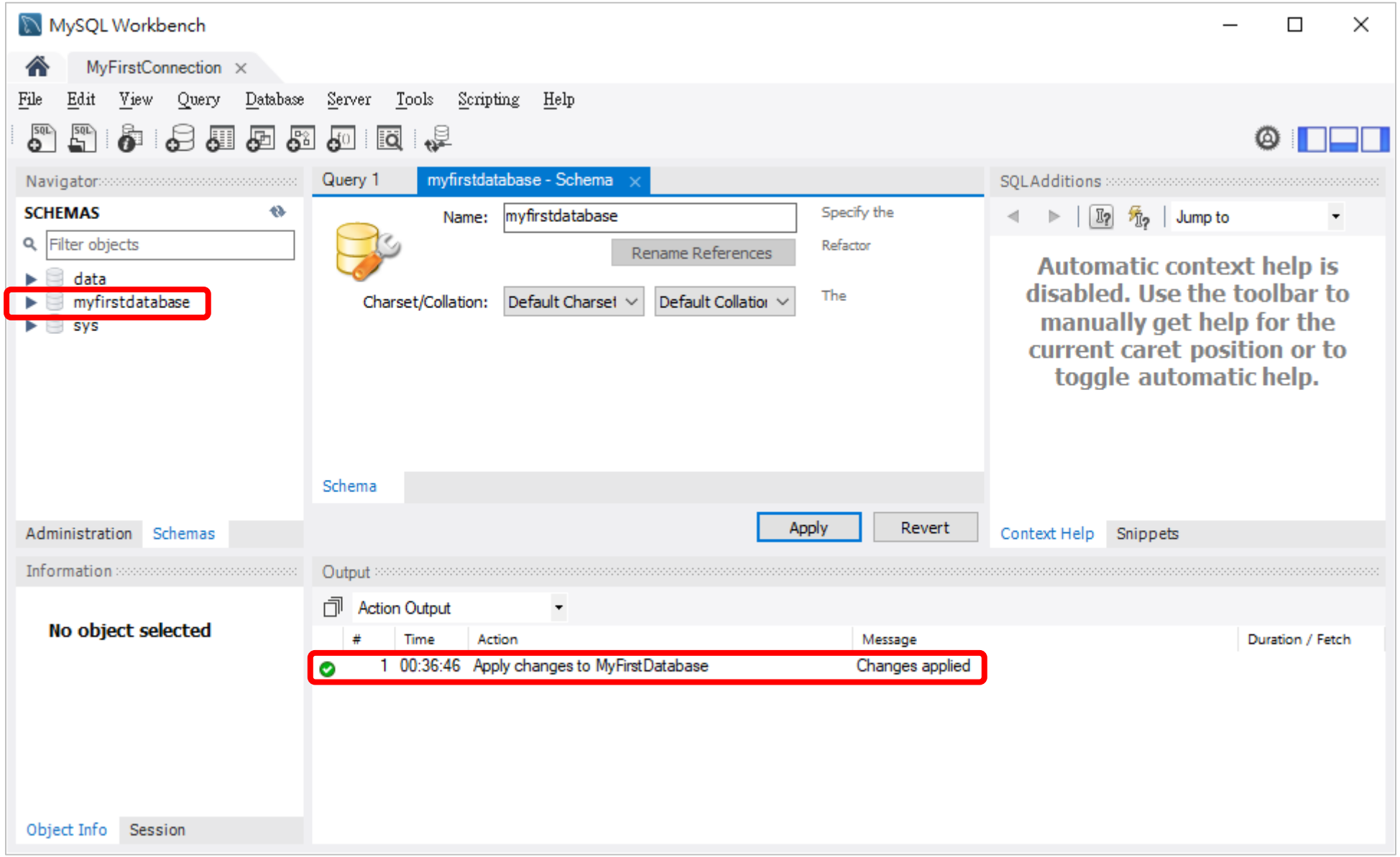
QT WITH MySQL (MySQL PART)

Click “Finish”.



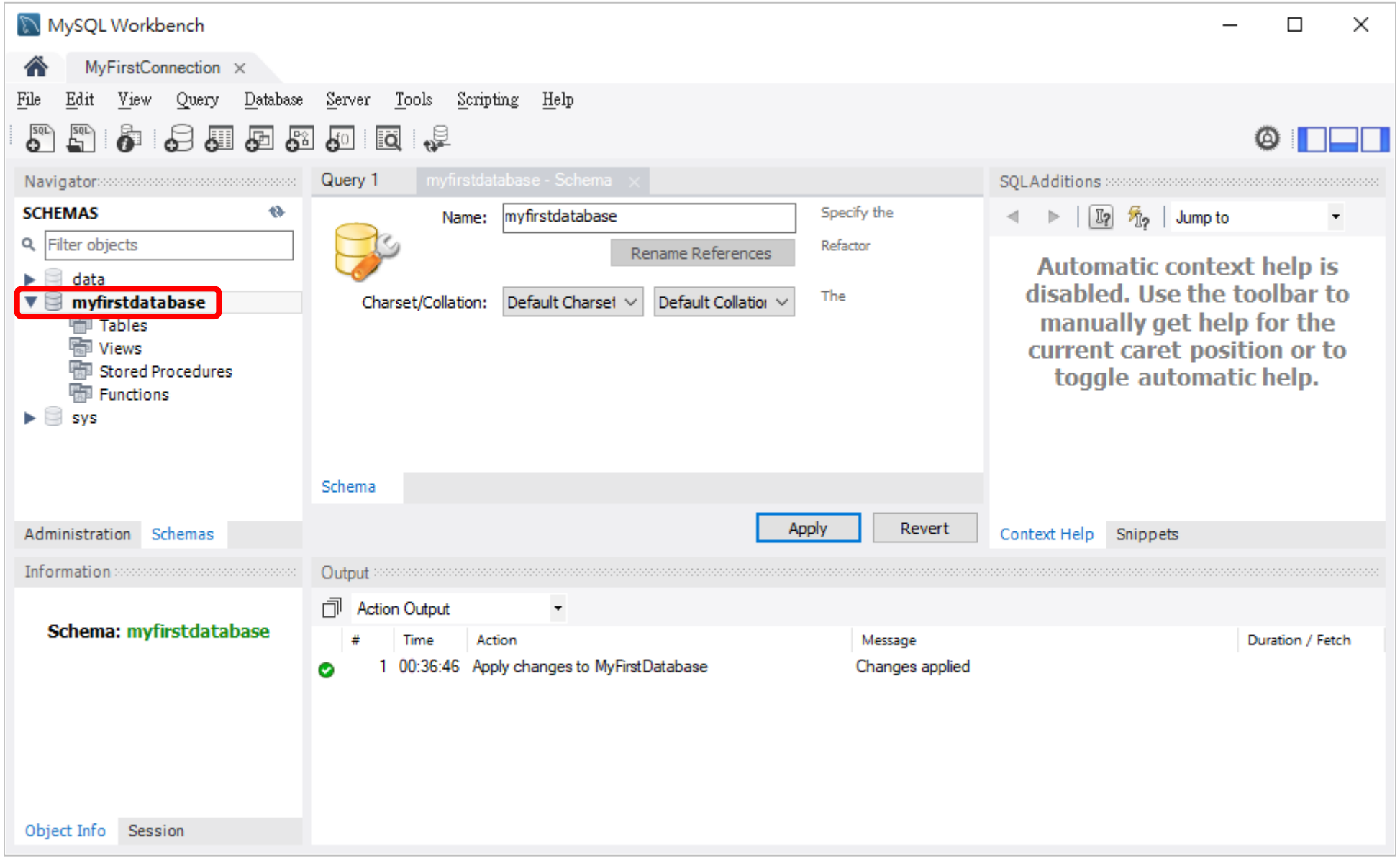
QT WITH MySQL (MySQL PART)

The new schema will be created.



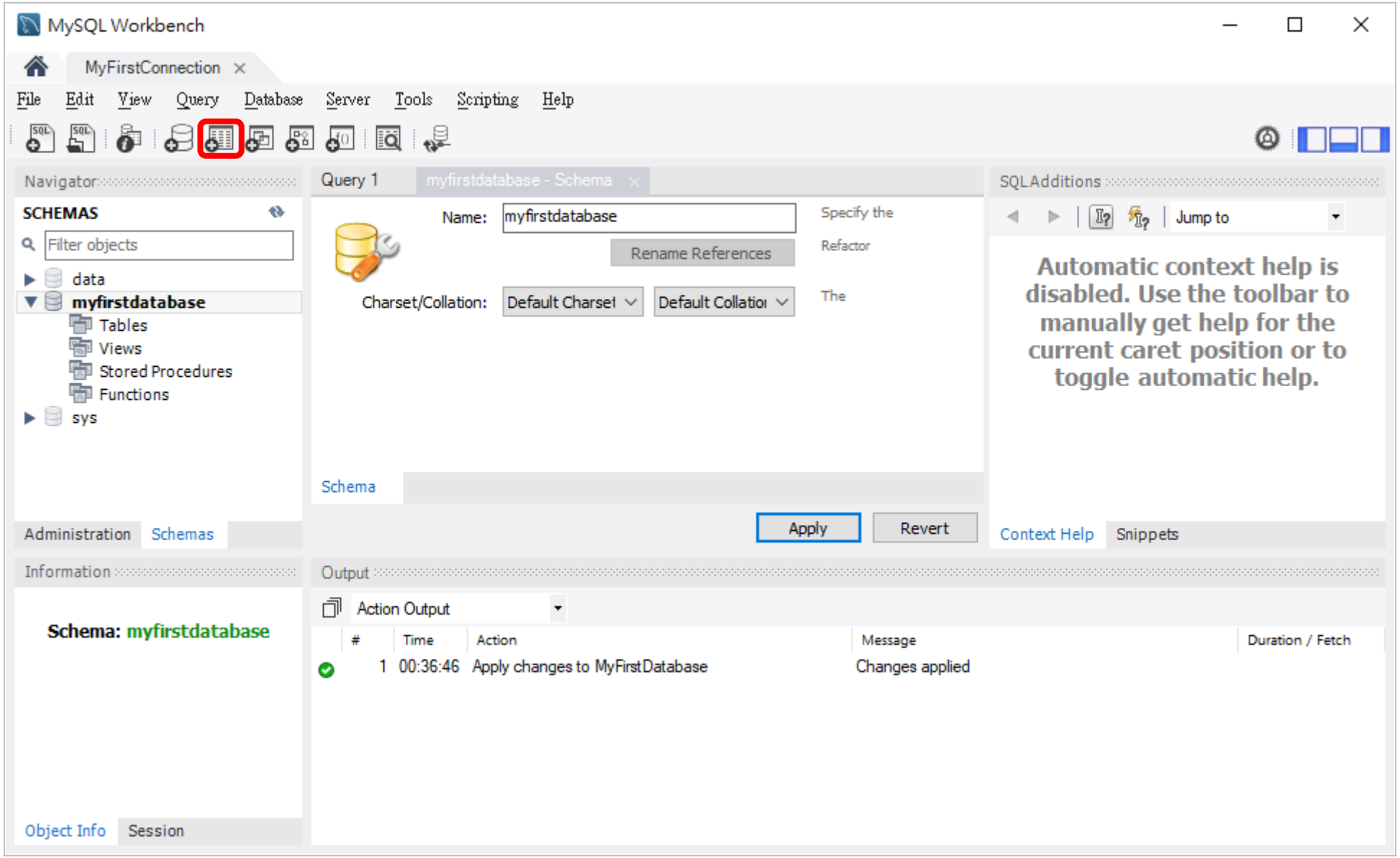
QT WITH MySQL (MySQL PART)

Double click schema “myfirstdatabase” and it will be highlighted.



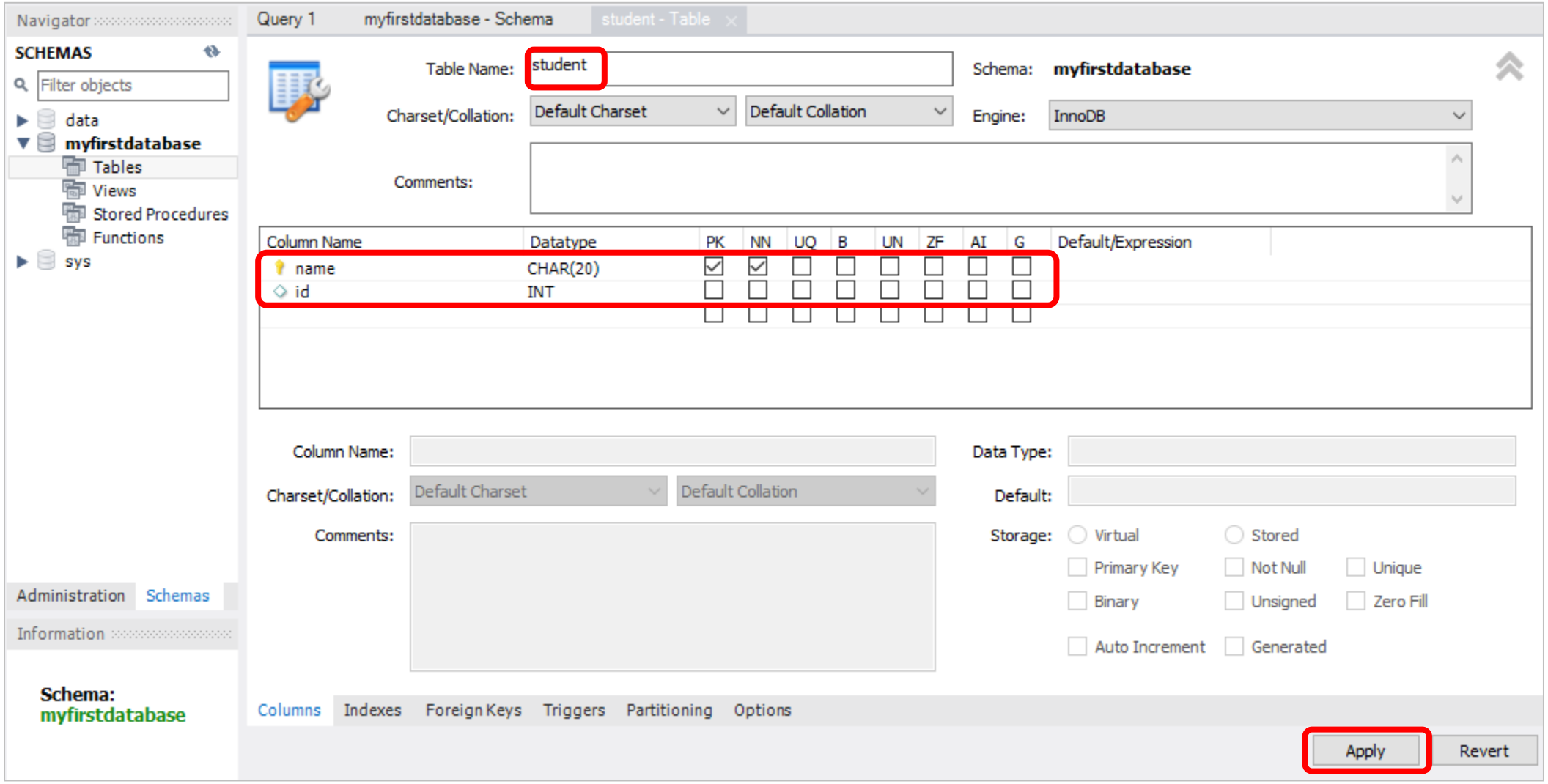
QT WITH MySQL (MySQL PART)

Click the button “Create a new table in the active schema in the connected server” to create a new table.



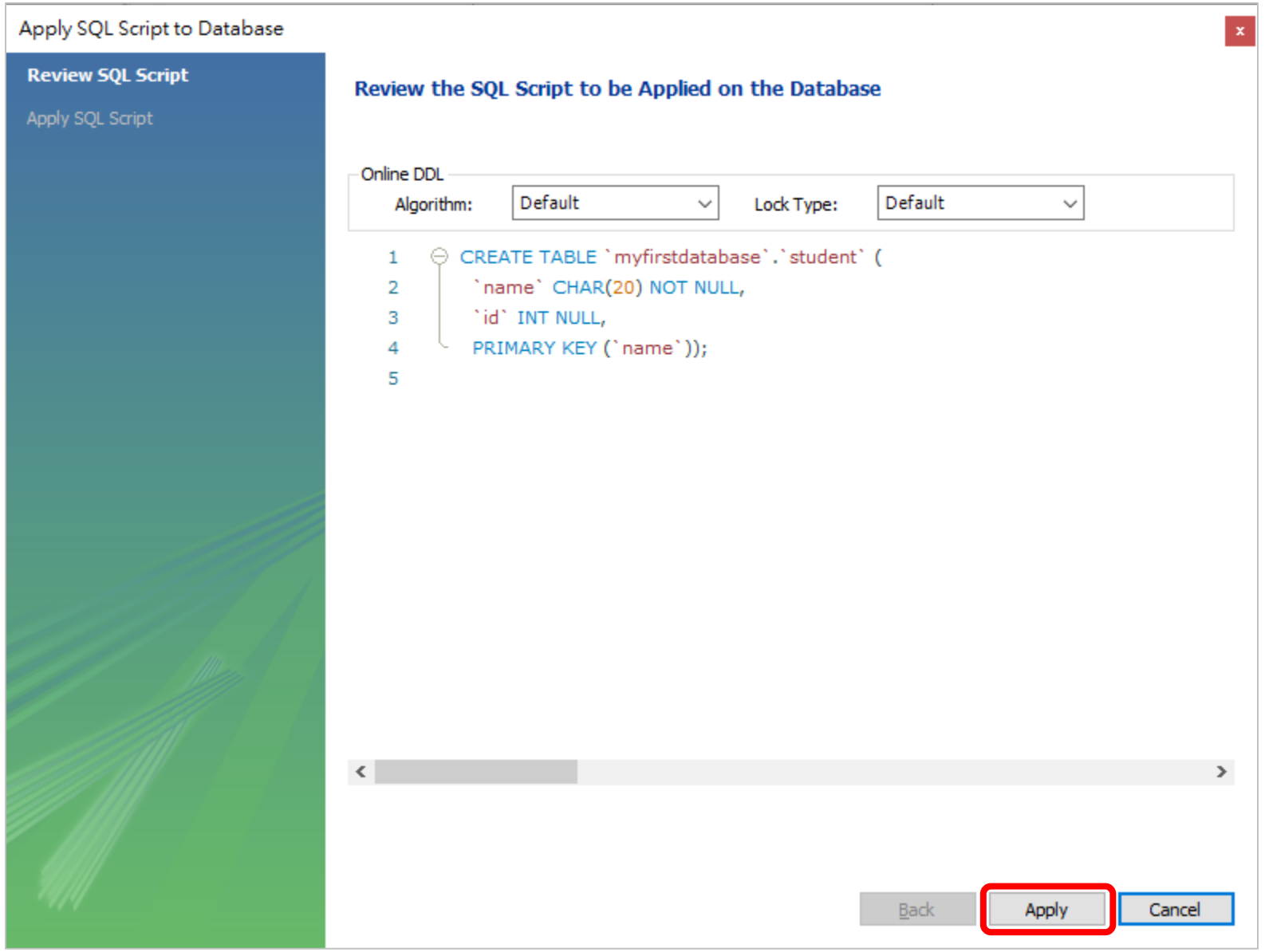
QT WITH MYSQL (MYSQL PART)

Key in the “Table Name”, and then key in data information, and then click “Apply”.



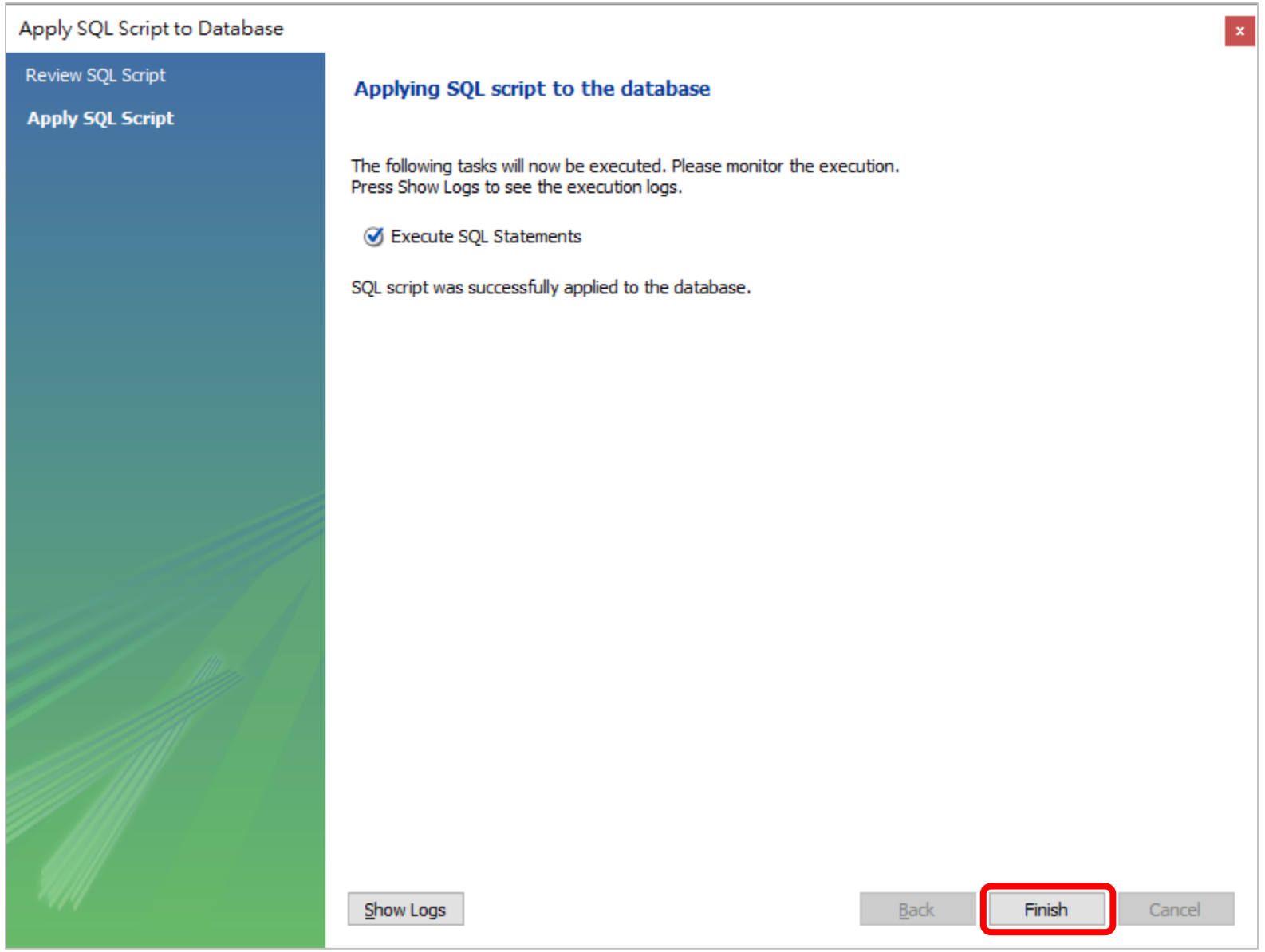
QT WITH MySQL (MySQL PART)

Following SQL script will be shown, and you can modify the command if necessary.



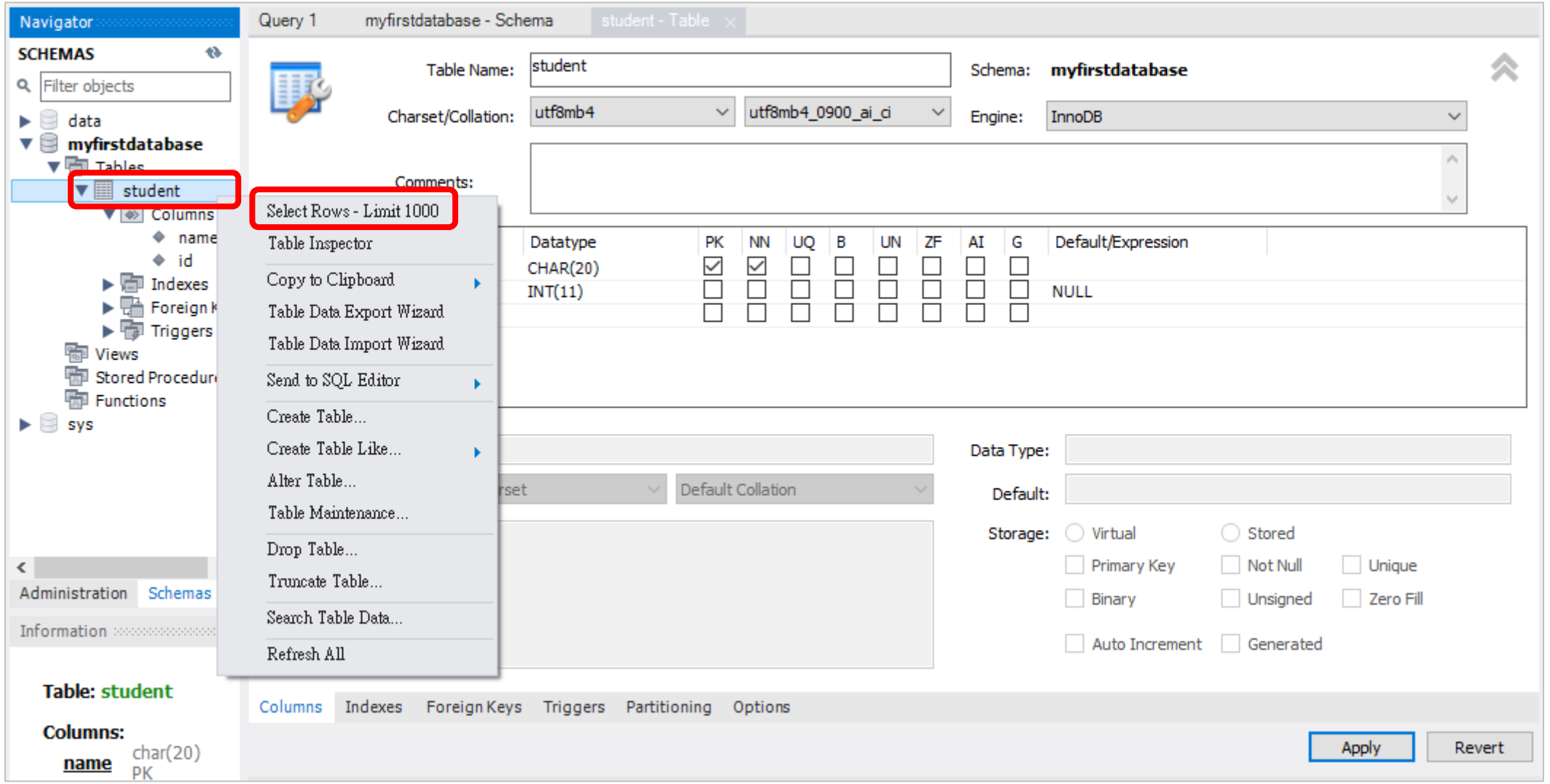
QT WITH MySQL (MySQL PART)

Click “Finish”.



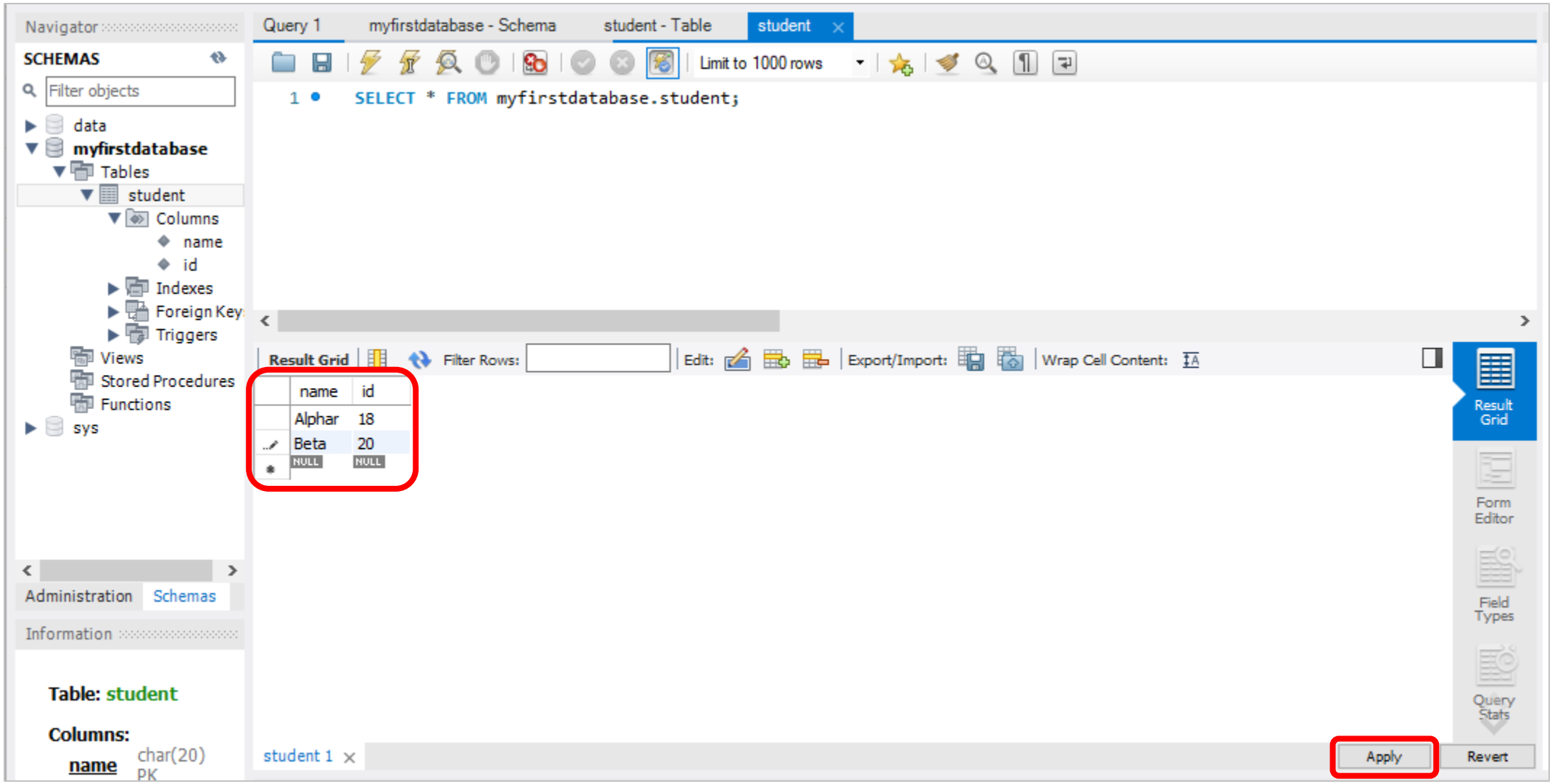
QT WITH MySQL (MySQL PART)

Right click on table “student and select ”Select Tows – Limit 1000.



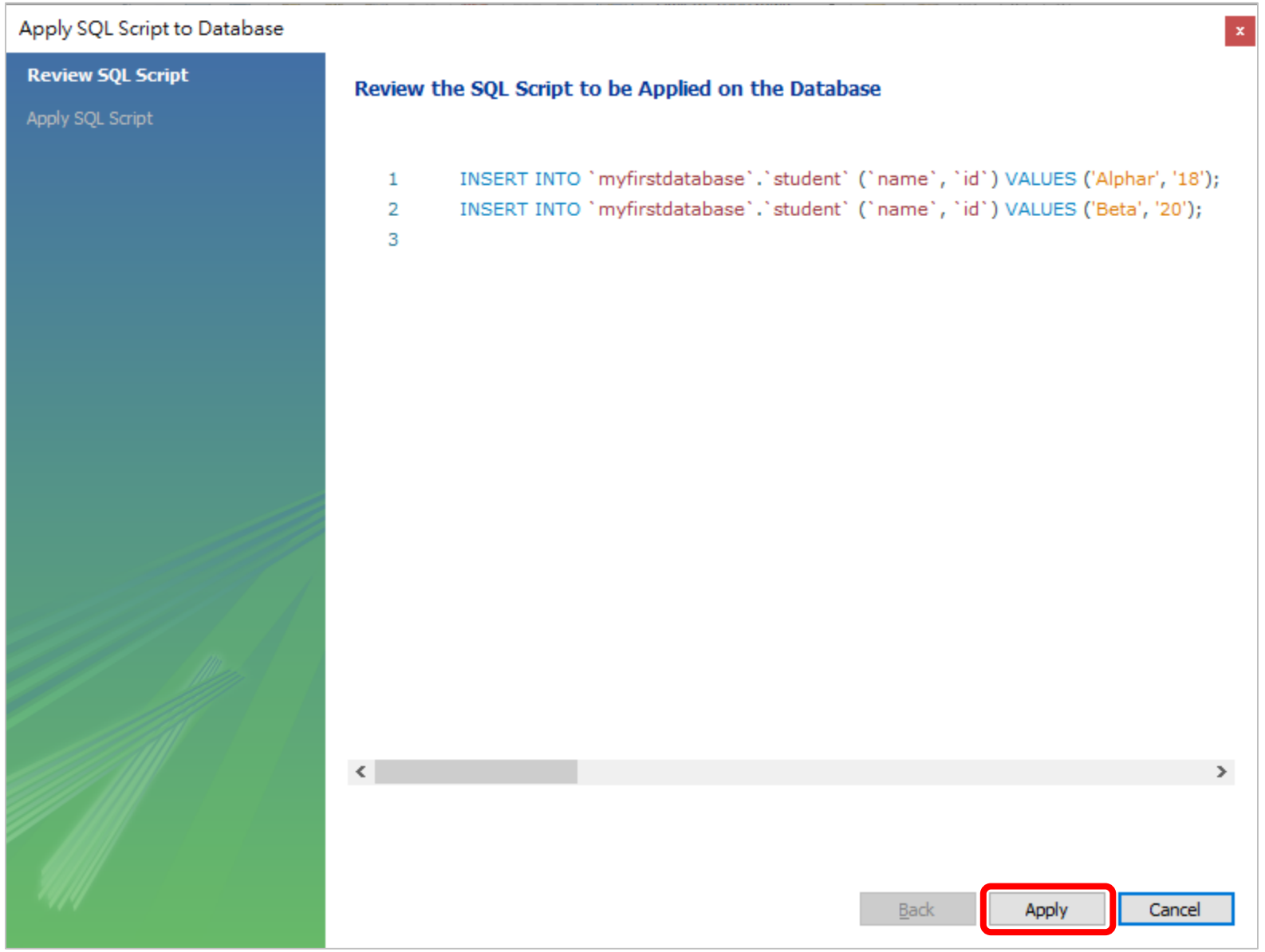
QT WITH MySQL (MySQL PART)

Key in data and click “Apply”.



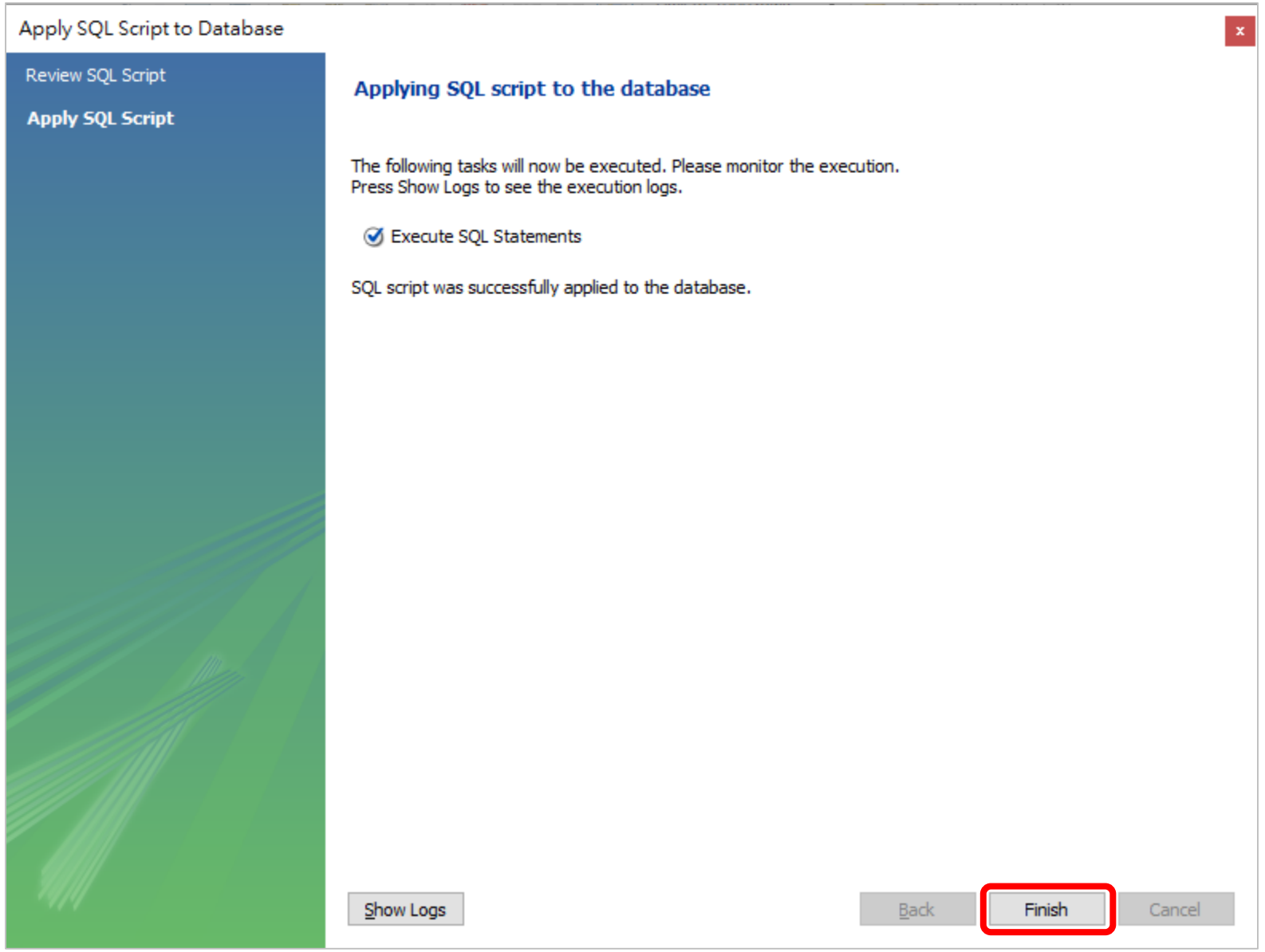
QT WITH MySQL (MySQL PART)

Following SQL script will be shown, and you can modify the command if necessary.



QT WITH MySQL (MySQL PART)

Click “Finish”.

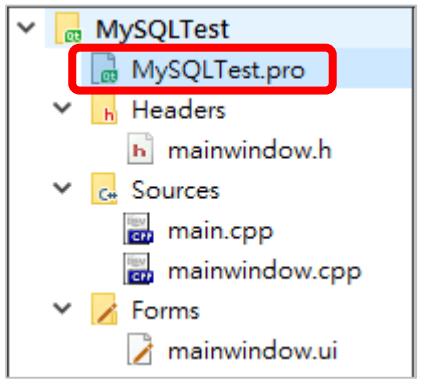


QT with MySQL (QT Part)

Example of QT connecting to MySQL

QT WITH MySQL (QT PART)

Add “QT += sql” in .pro file.



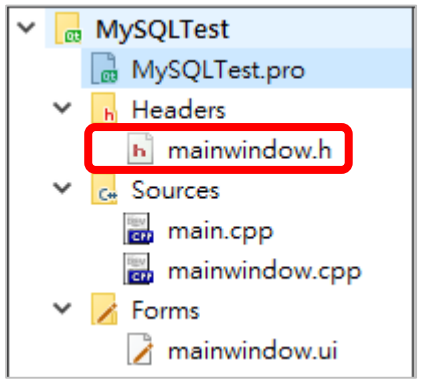
Tree of QT Widget Application

MySQLTest.pro

```
#-------------------------------------------------
#
# Project created by QtCreator 2019-11-18T17:55:24
#
#-------------------------------------------------

QT       += core gui sql

greaterThan(QT_MAJOR_VERSION, 4): QT += widgets

TARGET = MySQLTest
TEMPLATE = app

# The following define makes your compiler emit warnings if you use
# any feature of Qt which has been marked as deprecated (the exact warnings
# depend on your compiler). Please consult the documentation of the
# deprecated API in order to know how to port your code away from it.
DEFINES += QT_DEPRECATED_WARNINGS

# You can also make your code fail to compile if you use deprecated APIs.
# In order to do so, uncomment the following line.
# You can also select to disable deprecated APIs only up to a certain version of
Qt.
# disables all the APIs deprecated before Qt 6.0.0
#DEFINES += QT_DISABLE_DEPRECATED_BEFORE=0x060000

CONFIG += c++11

SOURCES += \
        main.cpp \
        mainwindow.cpp

HEADERS += \
        mainwindow.h

FORMS += \
        mainwindow.ui

# Default rules for deployment.
qnx: target.path = /tmp/$${TARGET}/bin
else: unix:!android: target.path = /opt/$${TARGET}/bin
!isEmpty(target.path): INSTALLS += target
```

QT WITH MySQL (QT PART)

Add following code in mainwindow.h.



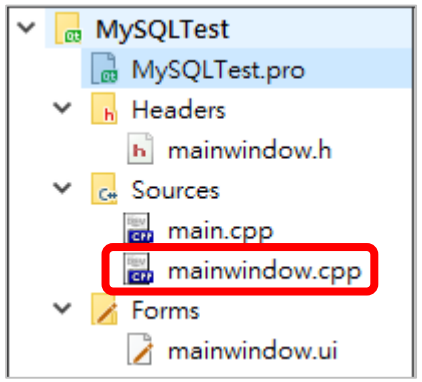
Tree of QT Widget Application

Mainwindow.h

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include <QDebug>
#include <QSqlResult>
#include <QSqlDatabase>


namespace Ui {
class MainWindow;
}

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    explicit MainWindow(QWidget *parent = nullptr);
    ~MainWindow();

    void connectDB();

private:
    Ui::MainWindow *ui;
};

#endif // MAINWINDOW_H
```

QT WITH MySQL (QT PART)

Add following code in mainwindow.cpp.



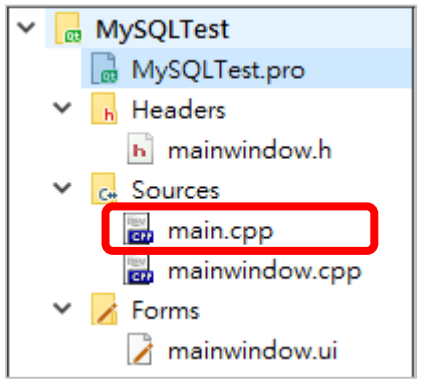
Tree of QT Widget Application

mainwindow.cpp

```
#include "mainwindow.h"
#include "ui_mainwindow.h"

MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{
    ui->setupUi(this);
}

MainWindow::~MainWindow()
{
    delete ui;
}

void MainWindow::connectDB()
{
    QSqlDatabase database;
    database = QSqlDatabase::addDatabase("QMYSQL");
    database.setHostName("localhost");
    database.setDatabaseName("myfirstdatabase"); // schema name
    database.setUserName("root");
    database.setPassword("123456789"); // your password
    database.setPort(3306);
    bool ok = database.open();
    if(ok){
        qDebug()<<"Successful Connection.";

    }else{
        qDebug()<<"Error: Cannot connect!!!";
    }
}
```

QT WITH MySQL (QT PART)

Add following code in main.cpp.



Tree of QT Widget Application

main.cpp

```cpp
#include "mainwindow.h"
#include <QApplication>
#include <QSqlDatabase>
#include <QSqlQuery>


int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    MainWindow w;
    w.show();

    w.connectDB();

    QSqlQuery query;

    query.exec("select * from student");

    while(query.next())
    {
        qDebug()<<"Query Result:"
                <<query.value(0).toString()
                <<query.value(1).toString();
    }

    return a.exec();
}
```

QT WITH MySQL (QT PART)

The result should be shown as follows.