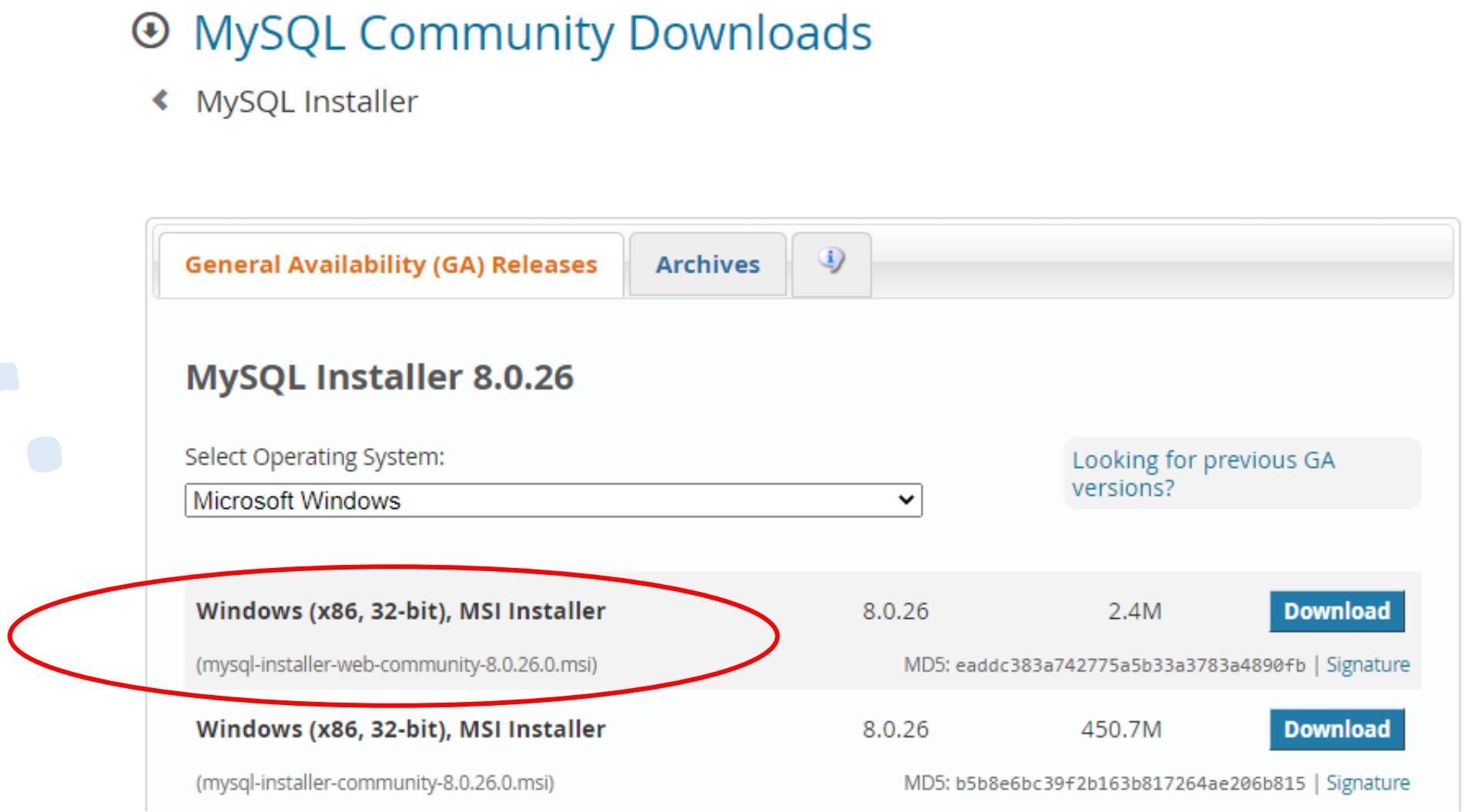
Experiment 1

Dr. Jyotismita Chaki

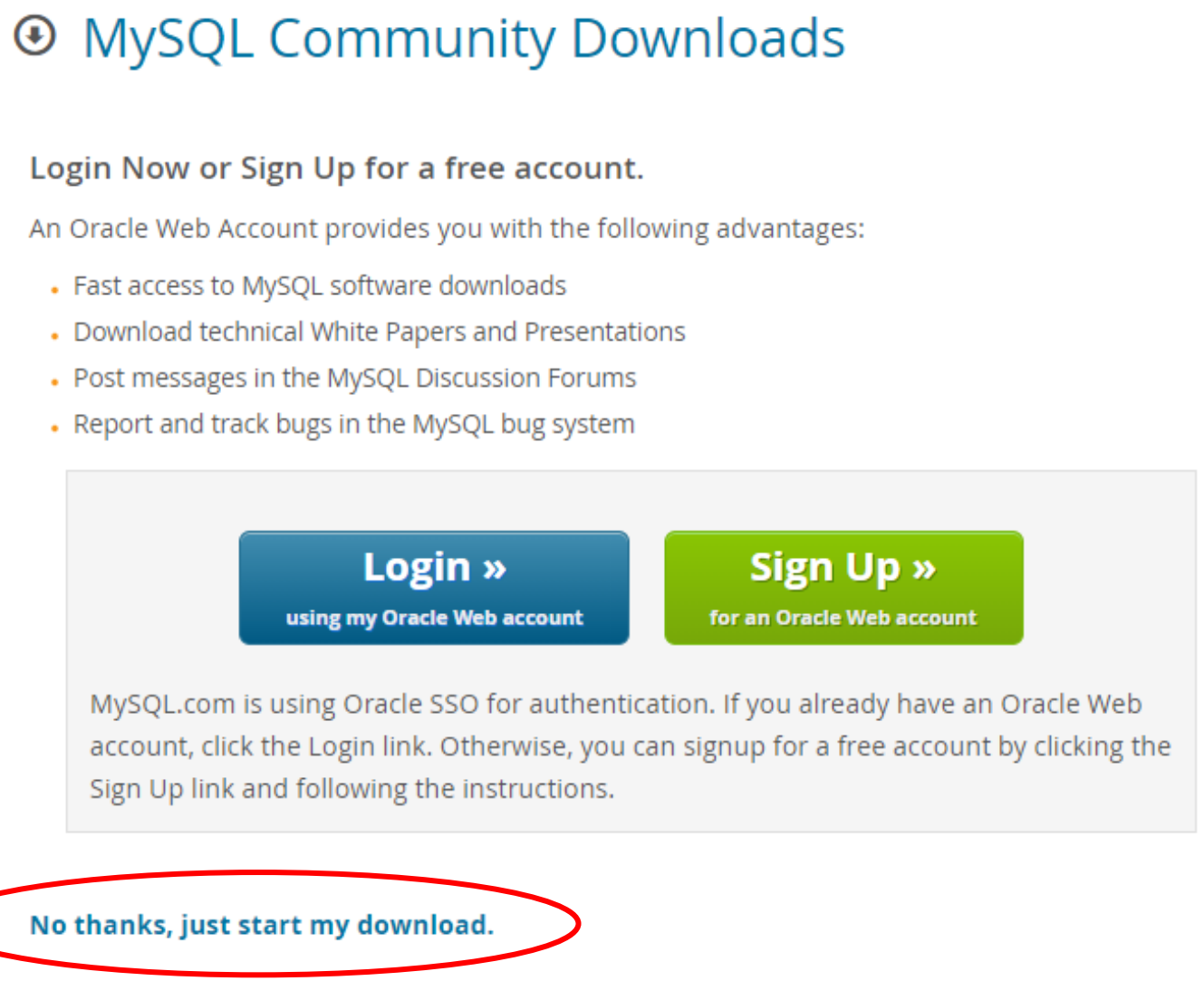
Installation: Download MySQL Installer

- If you want to install MySQL on the Windows environment, using MySQL installer is the easiest way.
- To download MySQL installer, go to the following link http://dev.mysql.com/downloads/installer/. There are two installer files:
- If you are connecting to the internet while installing MySQL, you can choose the online installation version mysql-installer-web-community-<version>.exe .
- In case you want to install MySQL offline, you can download the mysql-installer-community-<version>.exe file.

Installation

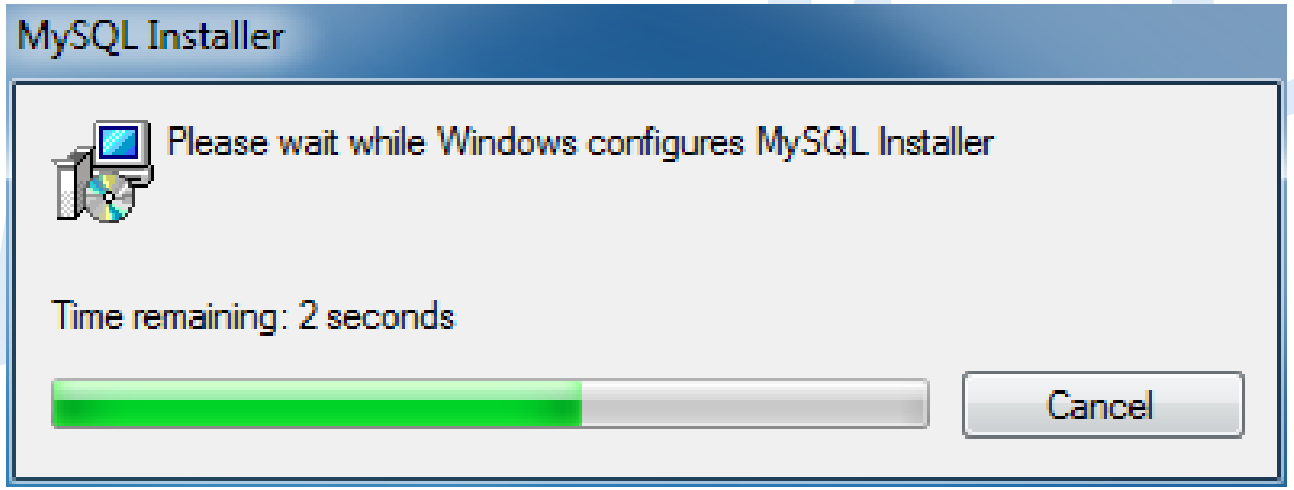


Installation

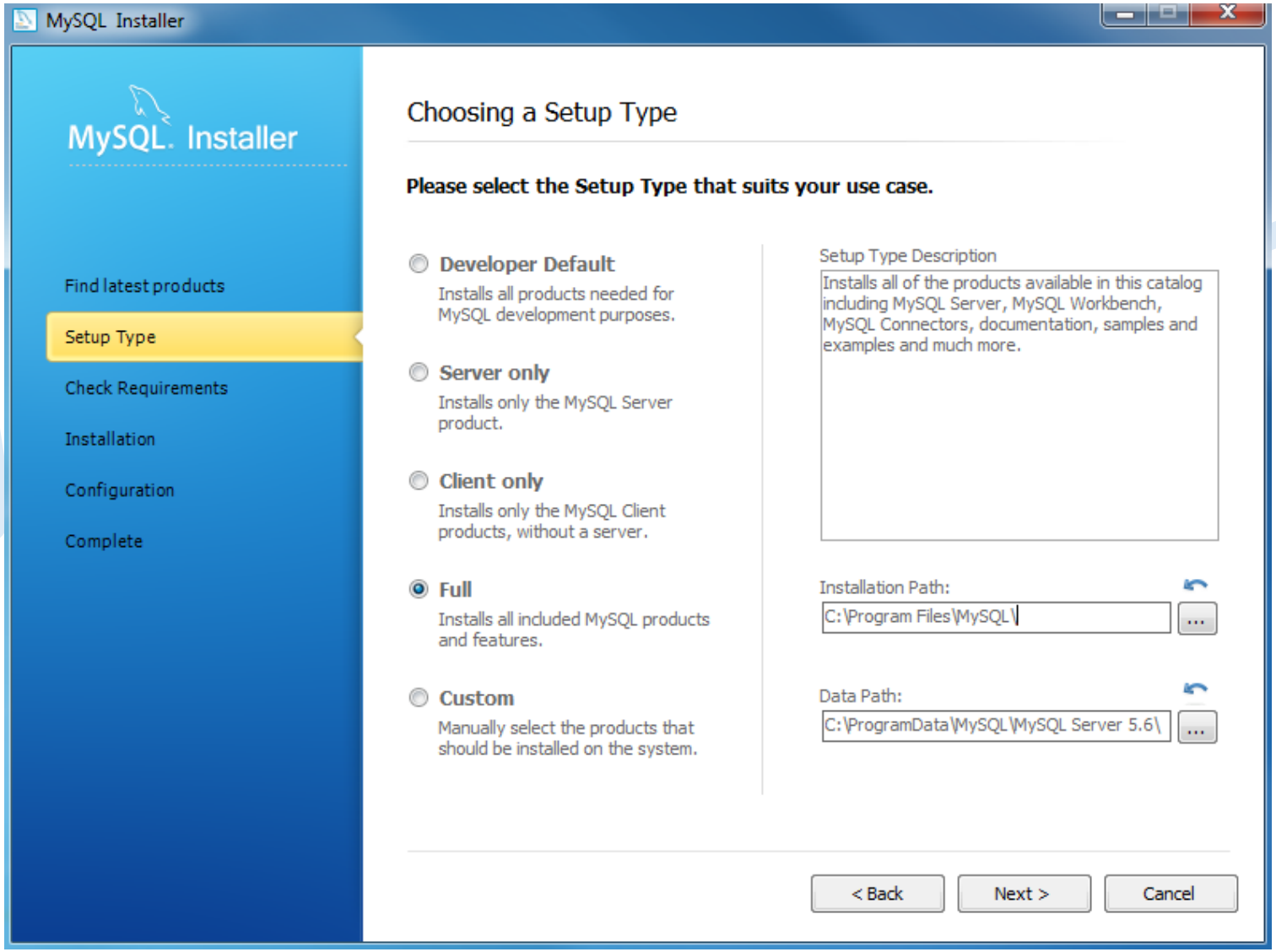


Installation: Install MySQL via MySQL Installer

- To install MySQL using the MySQL installer, double-click on the MySQL installer file and follow the steps below:

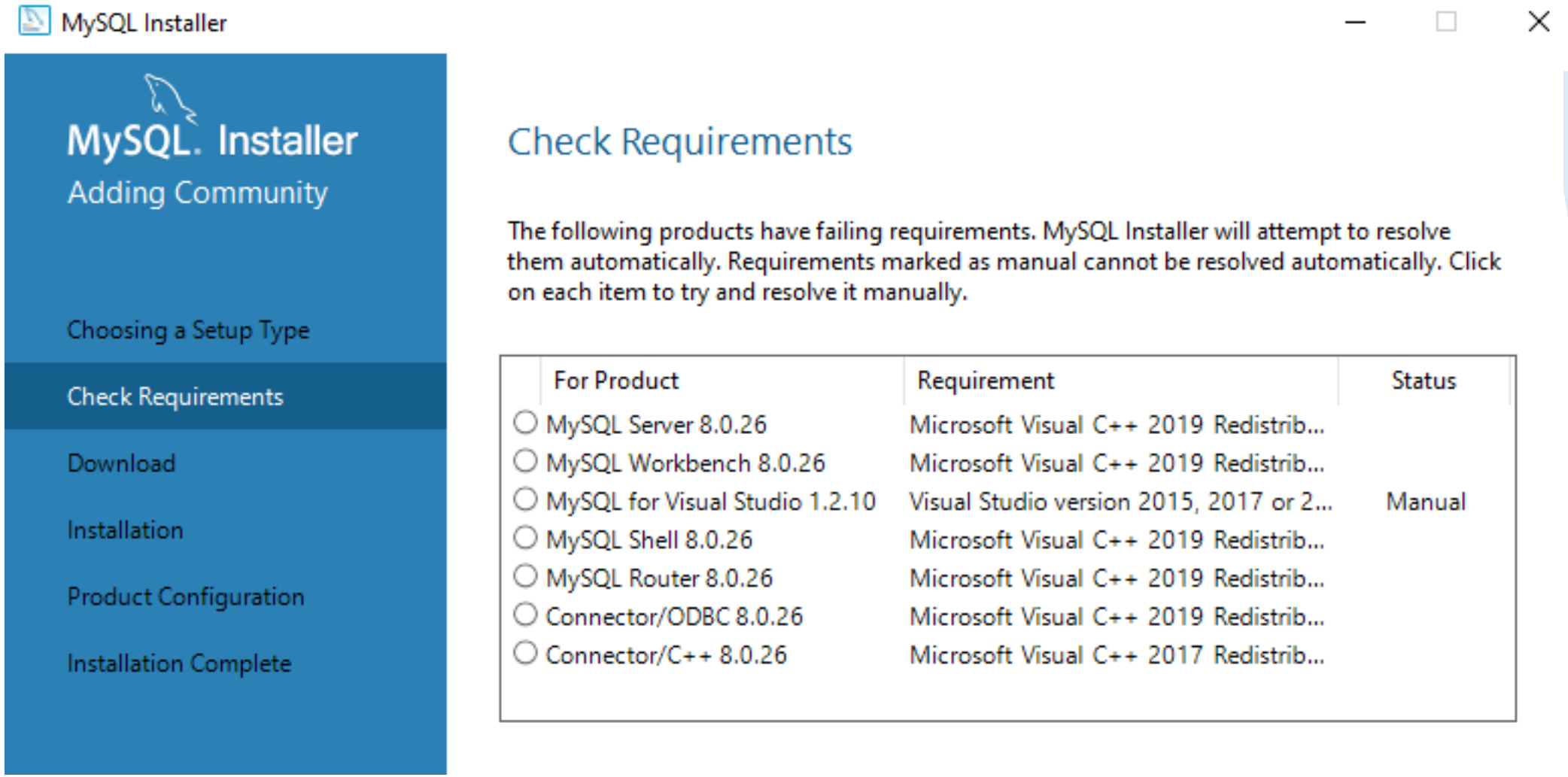


Installation: Setup Type



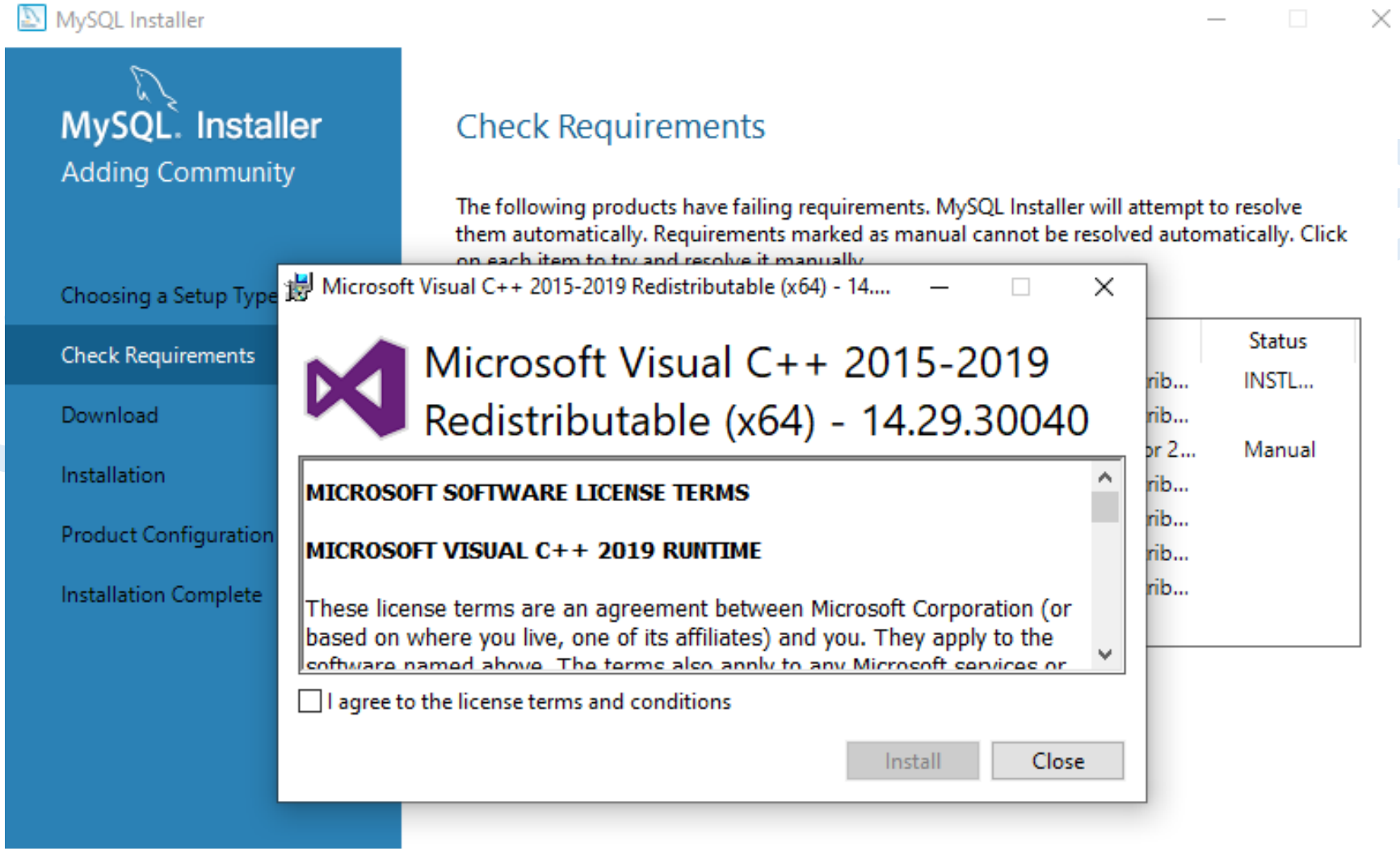
Installation

Select Execute



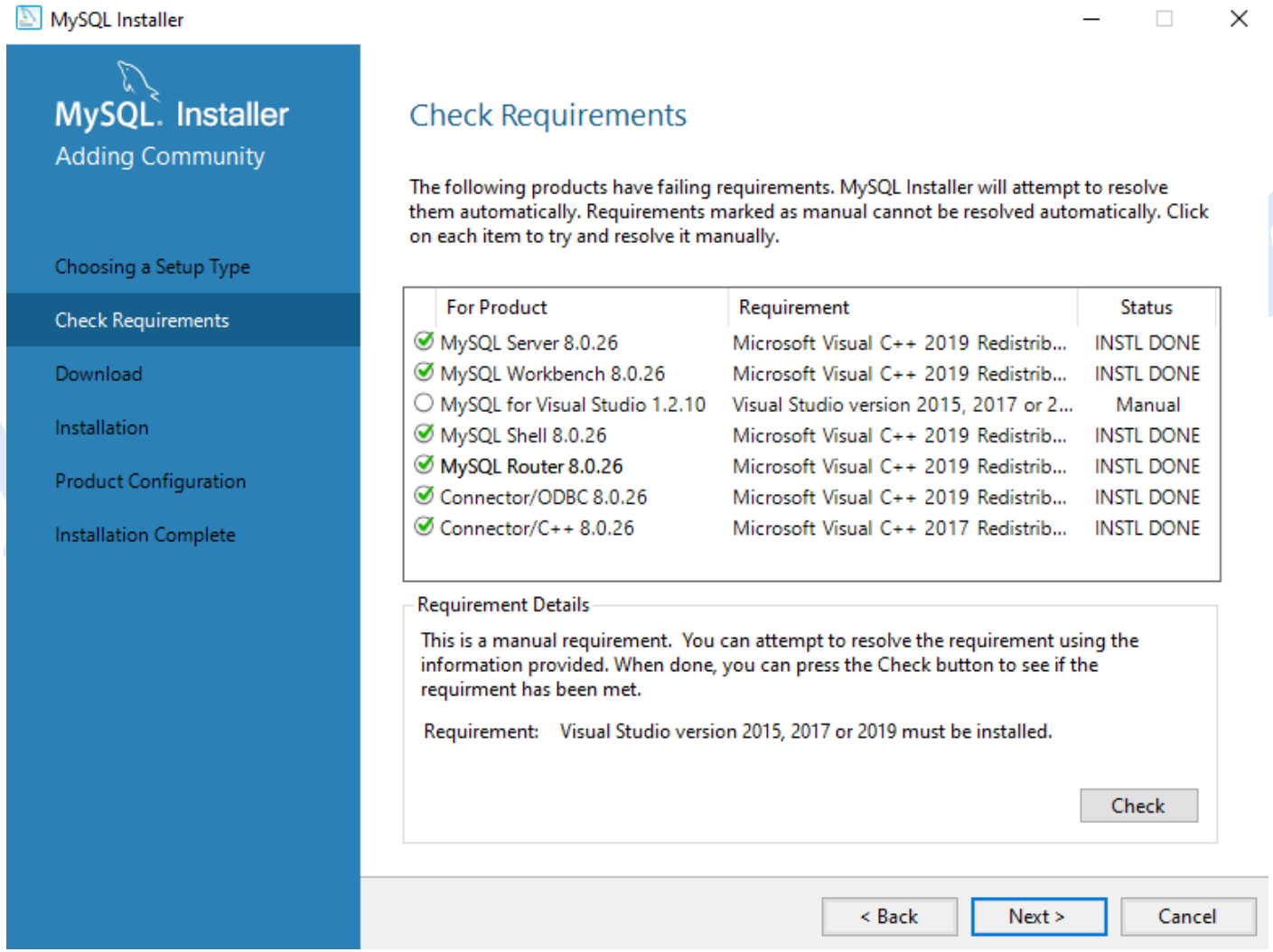
Installation

Select I agree, then Next



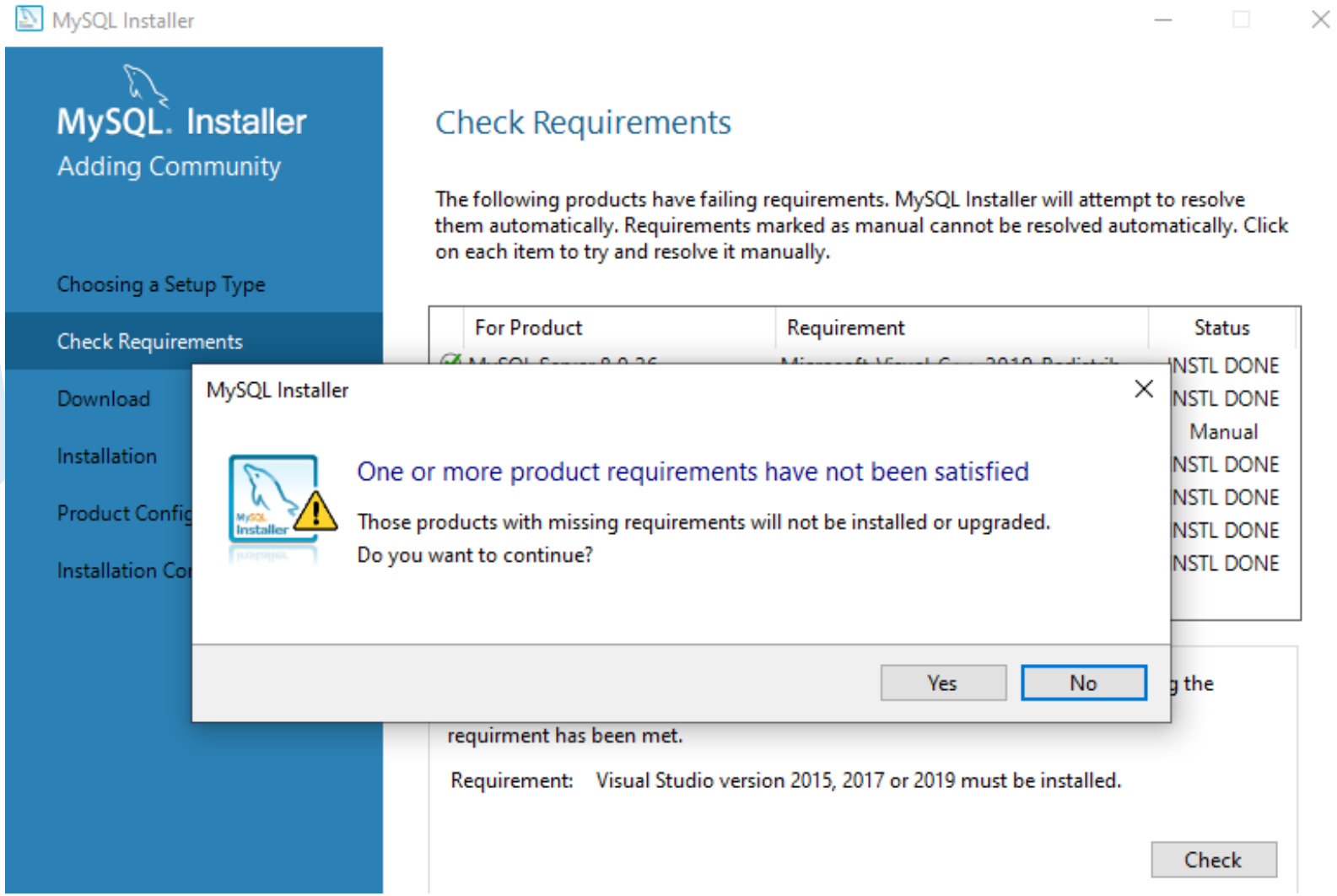
Installation

Select Next



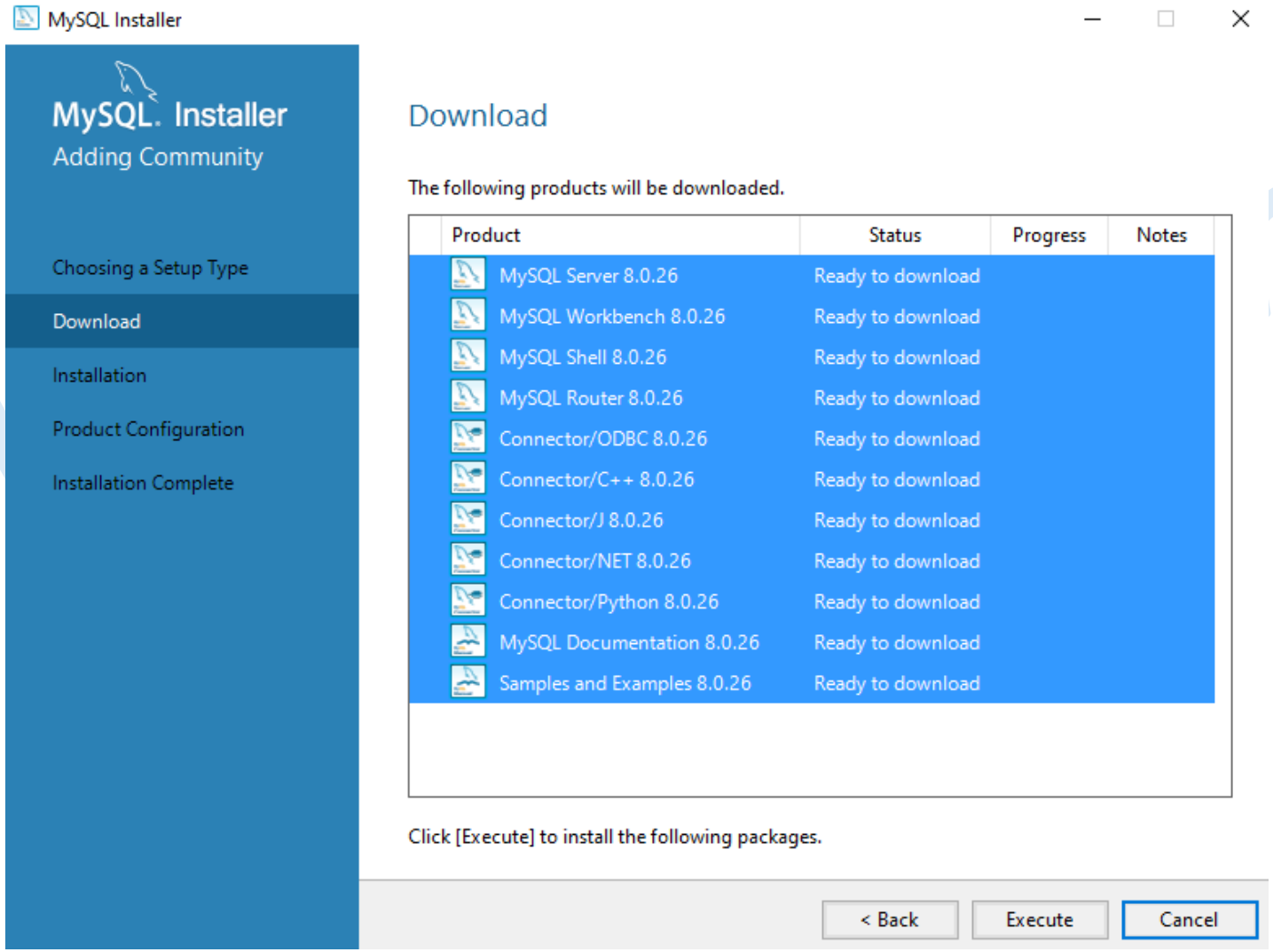
Installation

Select Yes



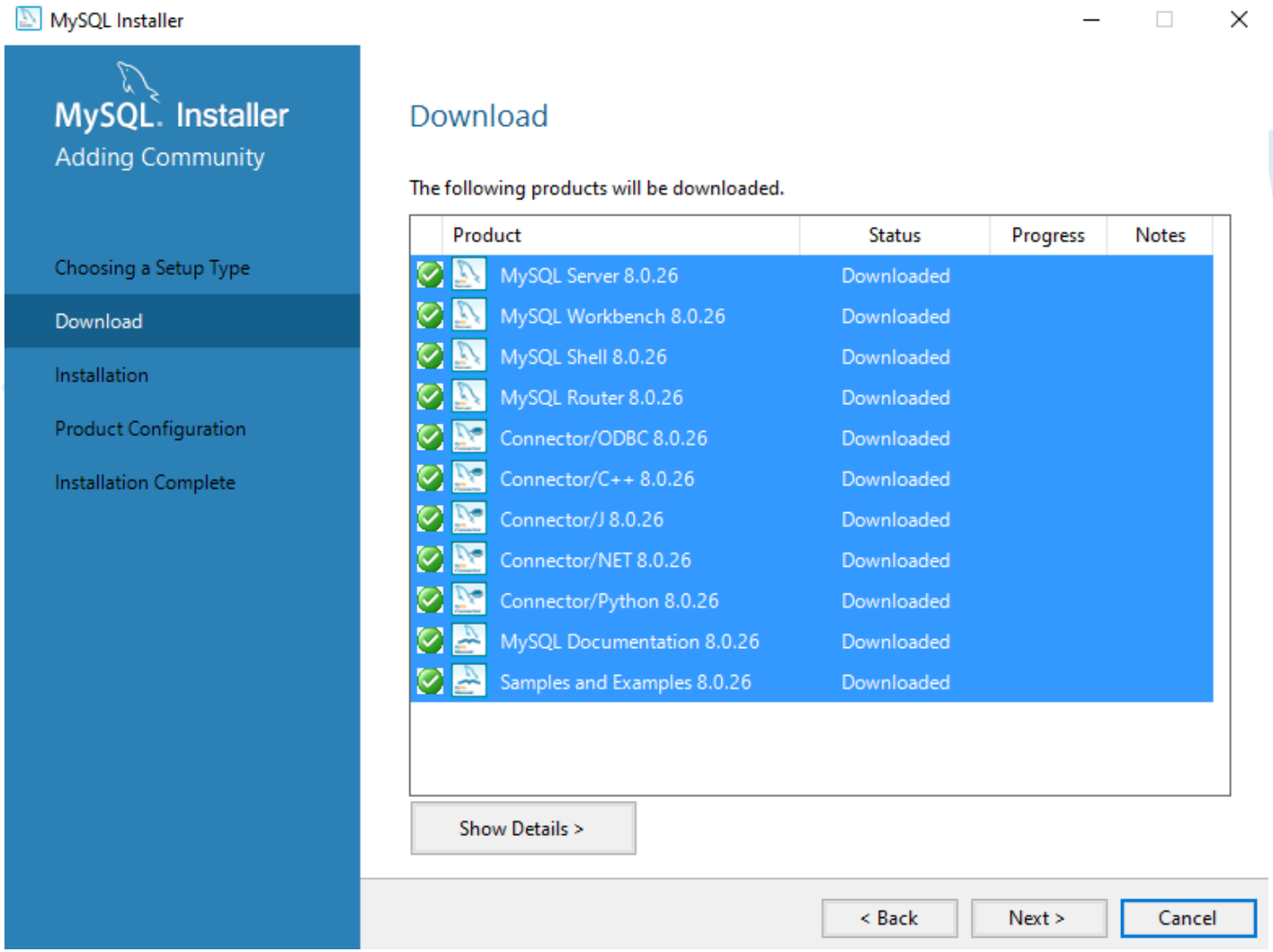
Installation

Select Execute



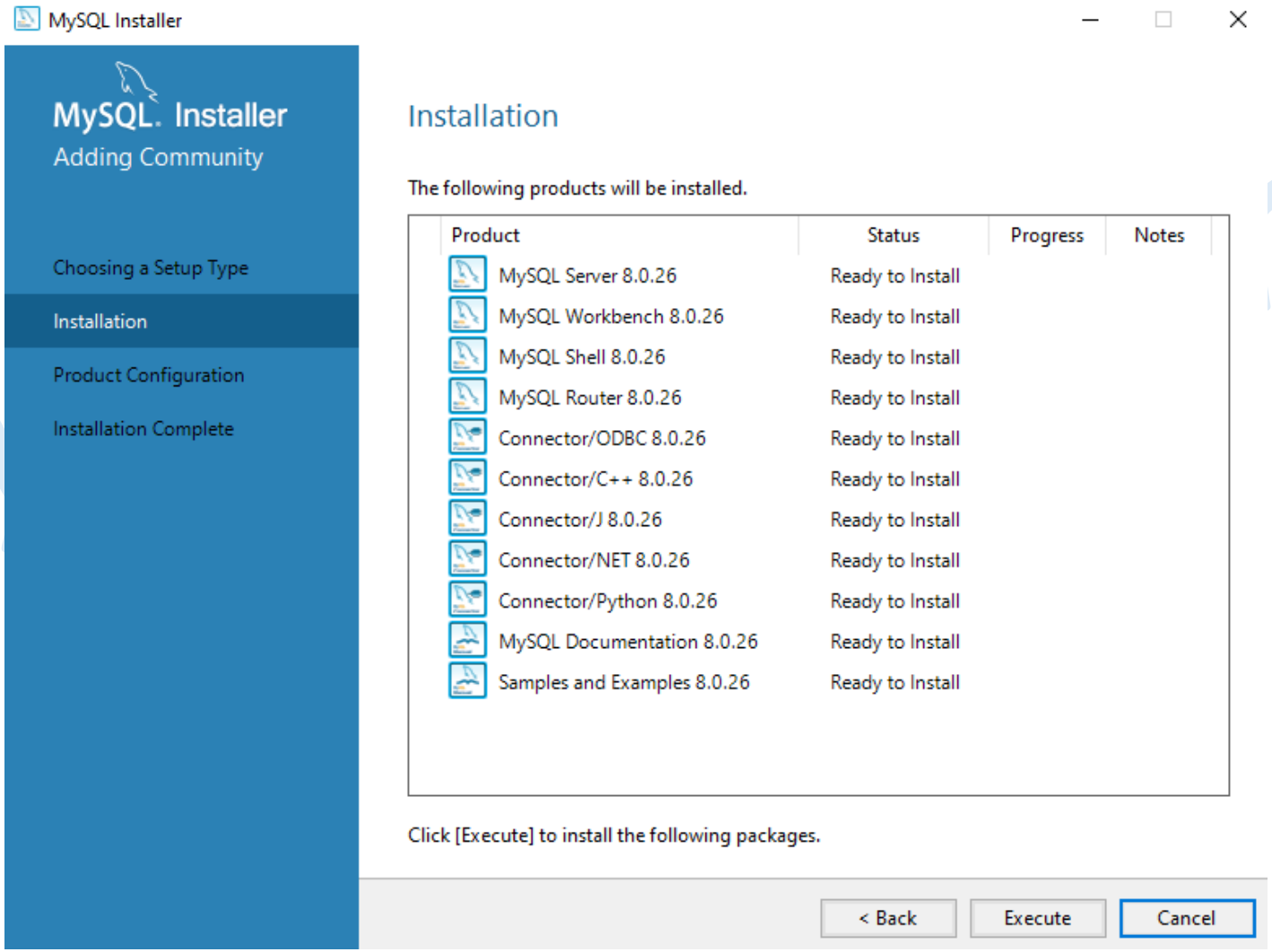
Installation

Select Next



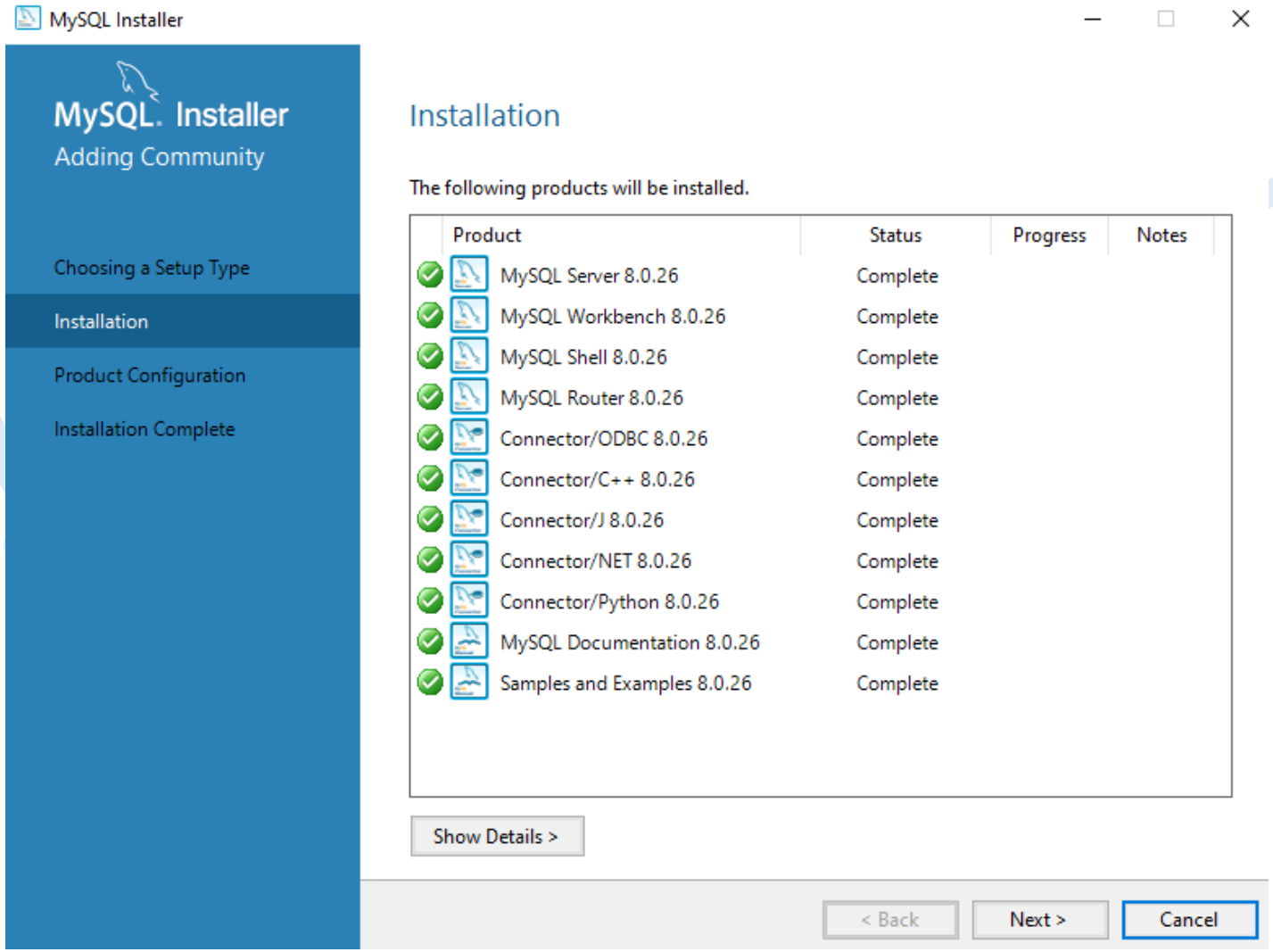
Installation

Select Execute



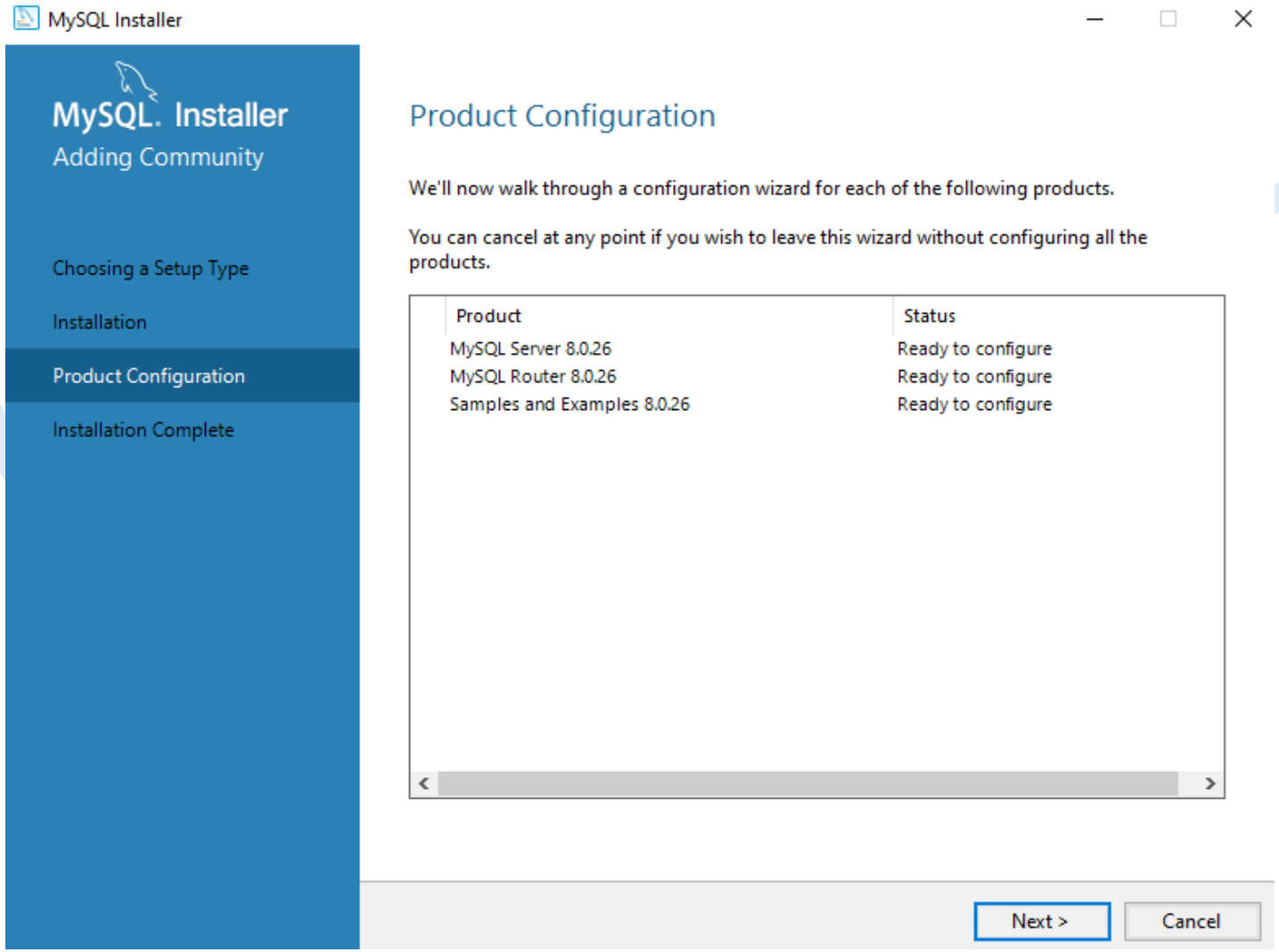
Installation

Select Next



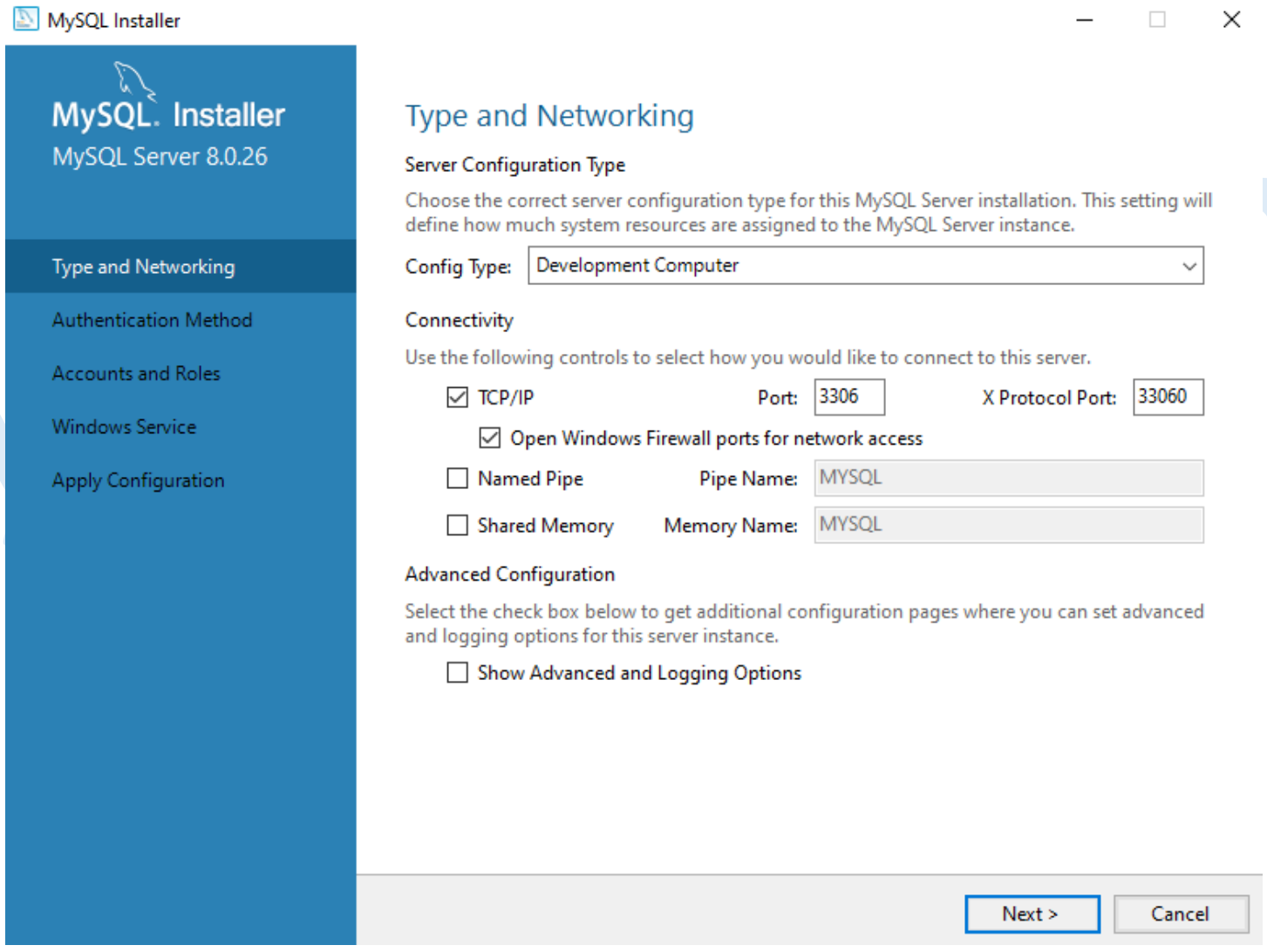
Installation

Select Next



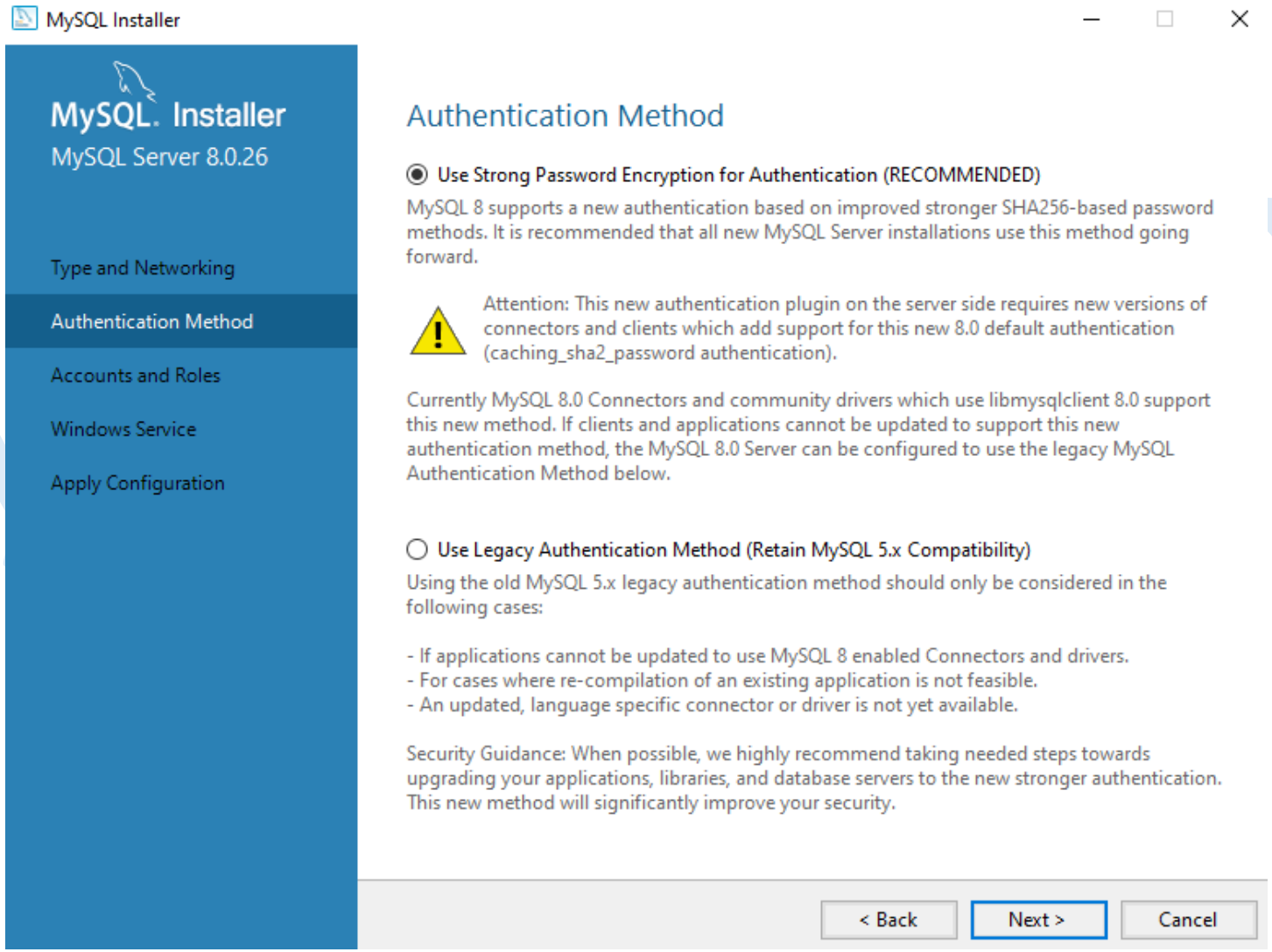
Installation

Select Next



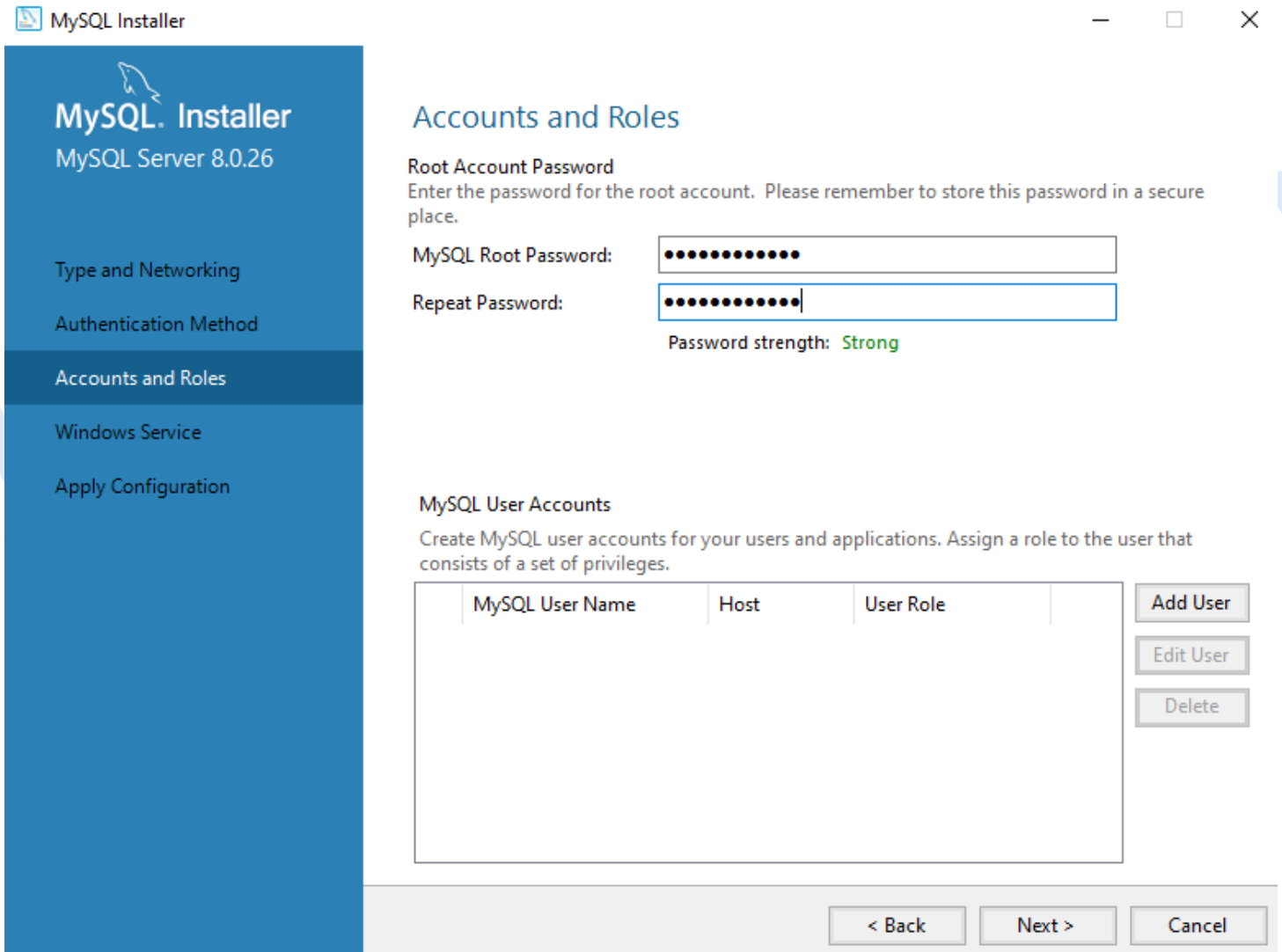
Installation

Select Next



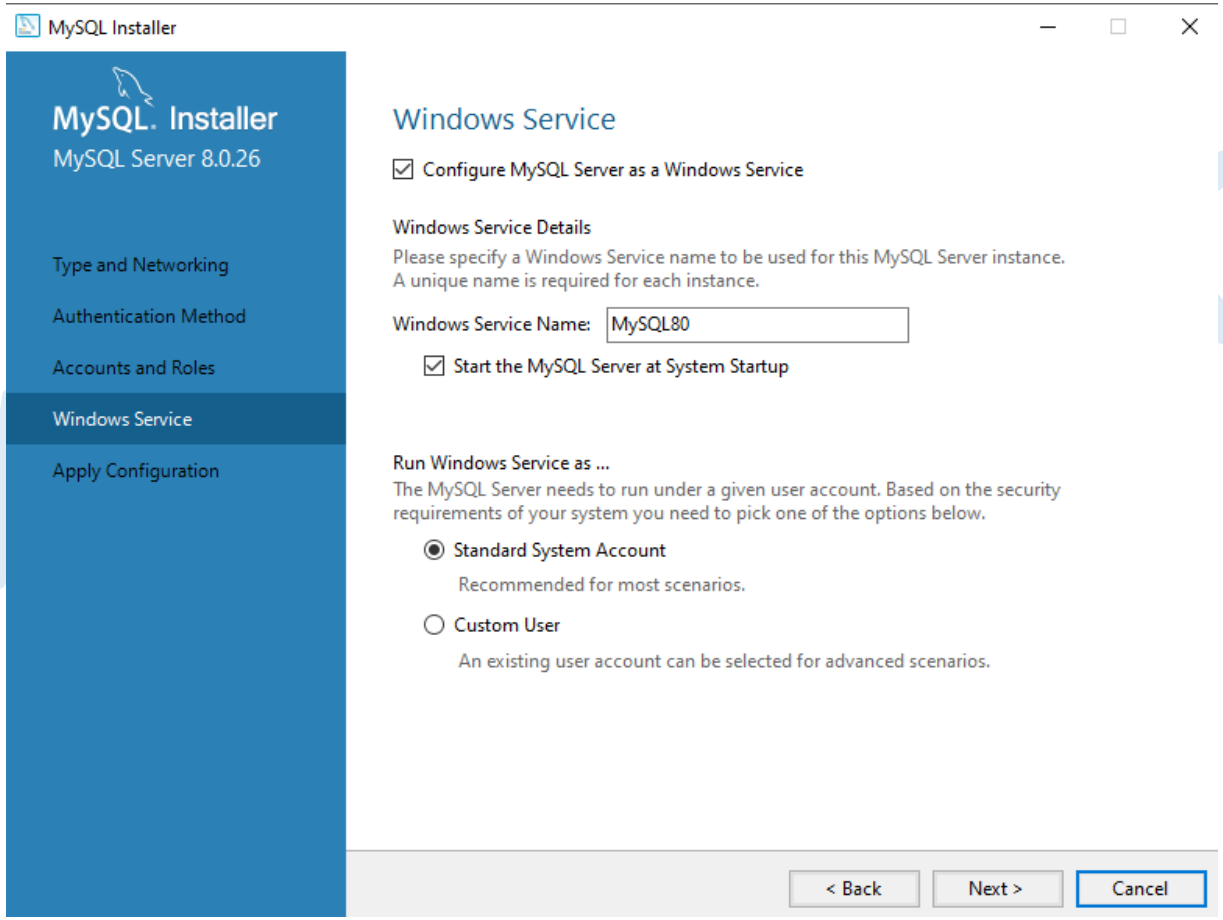
Installation

Select Next



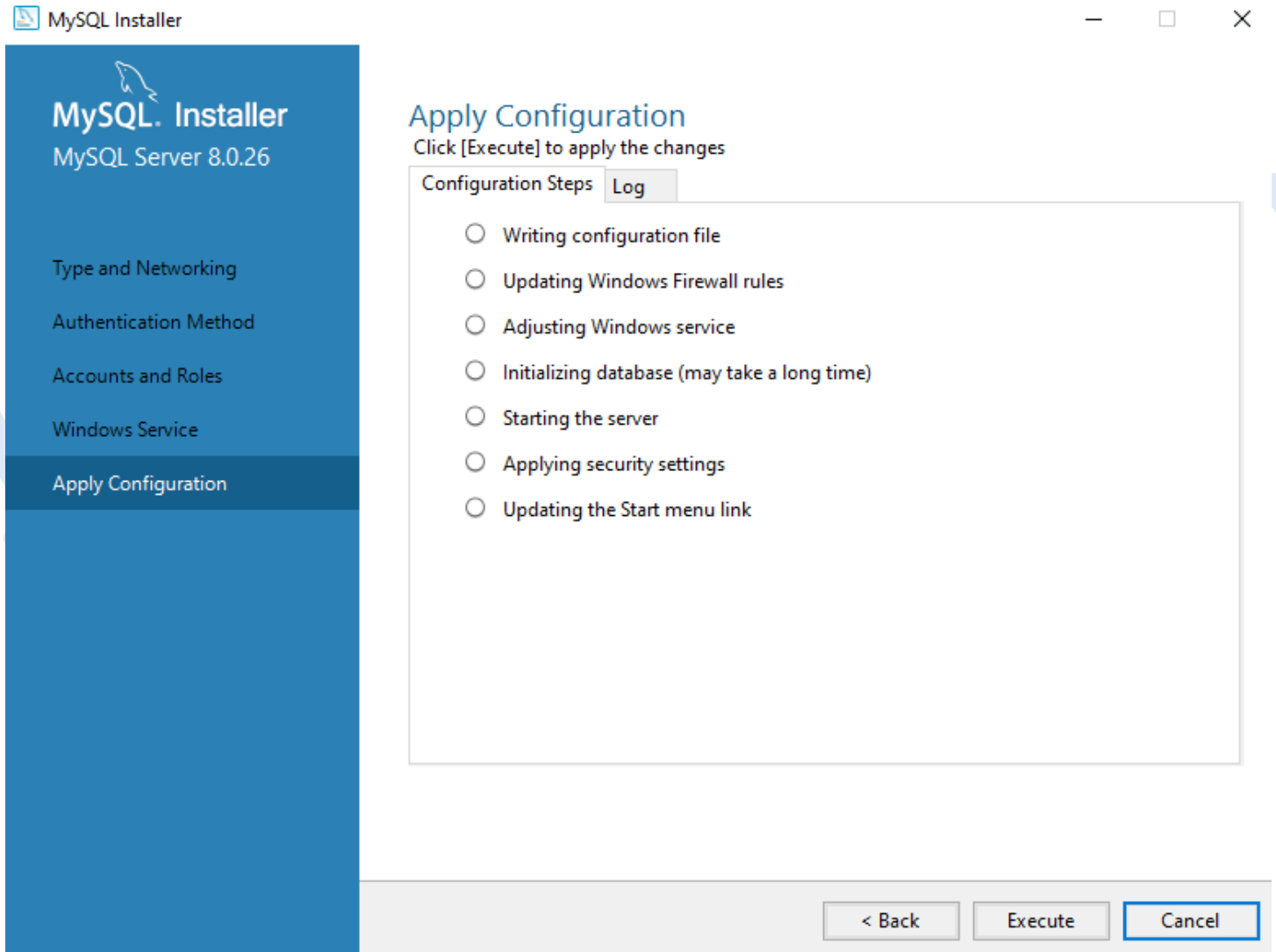
Installation

Select Next



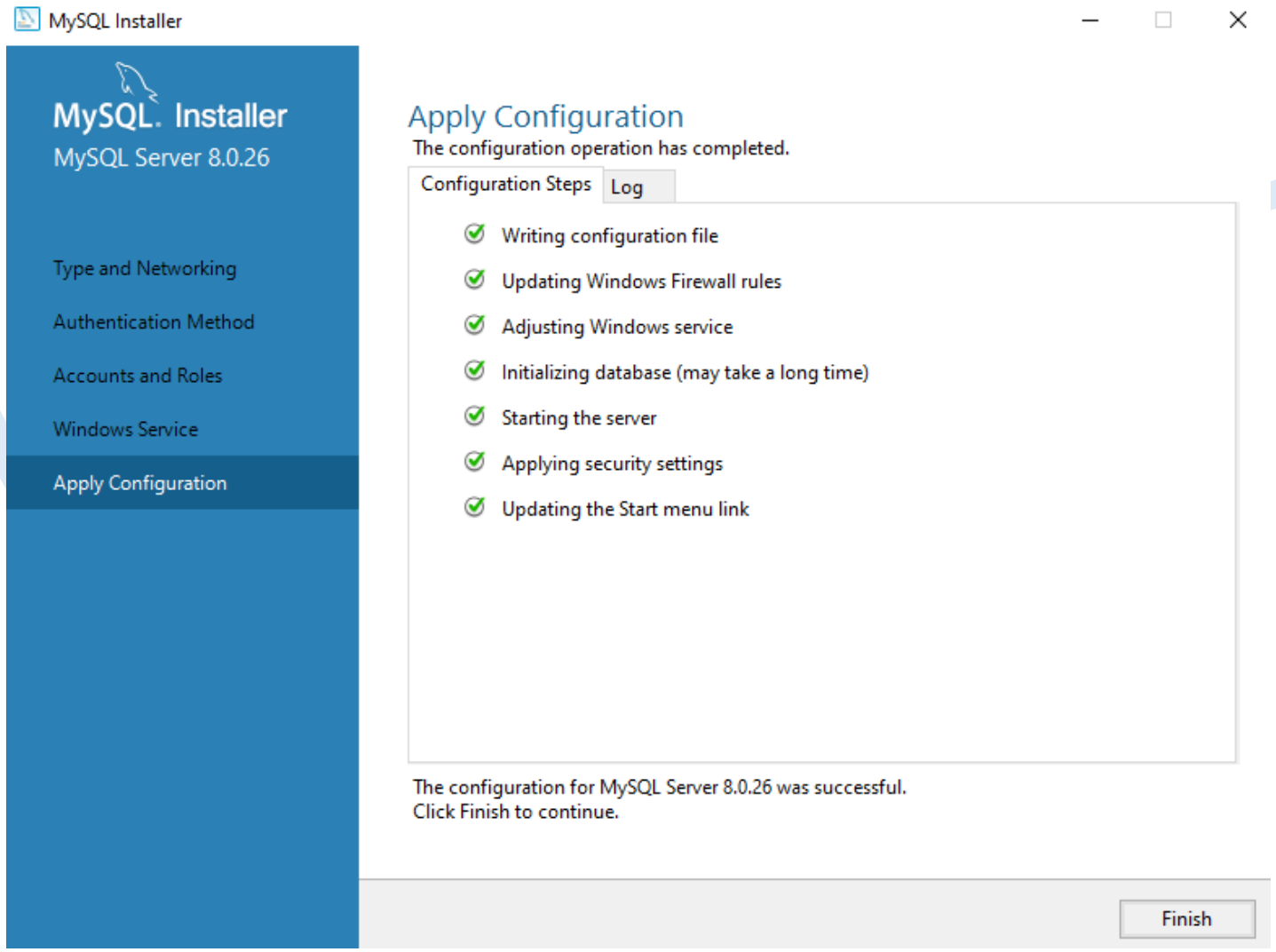
Installation

Select Execute



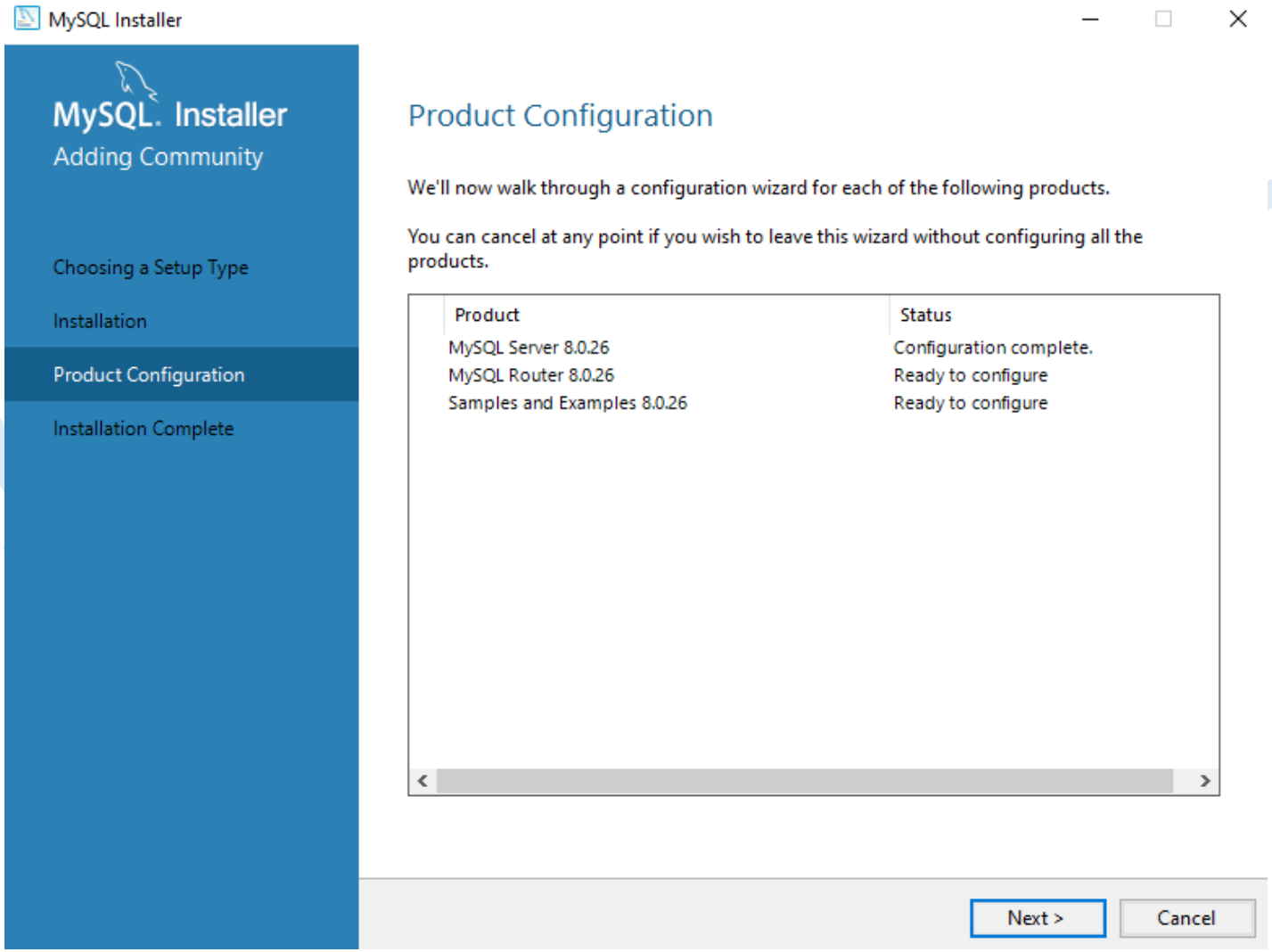
Installation

Select Finish



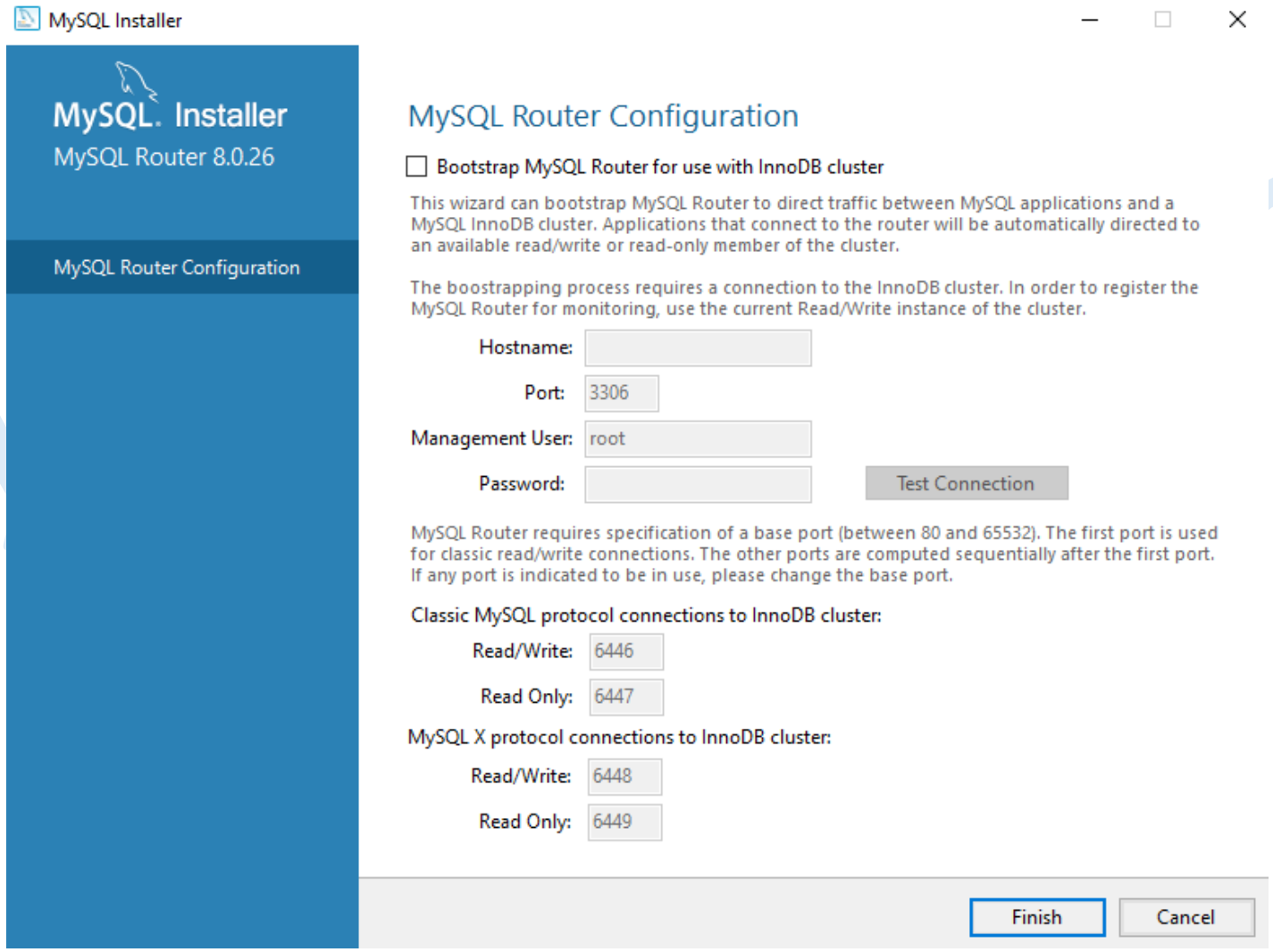
Installation

Select Next



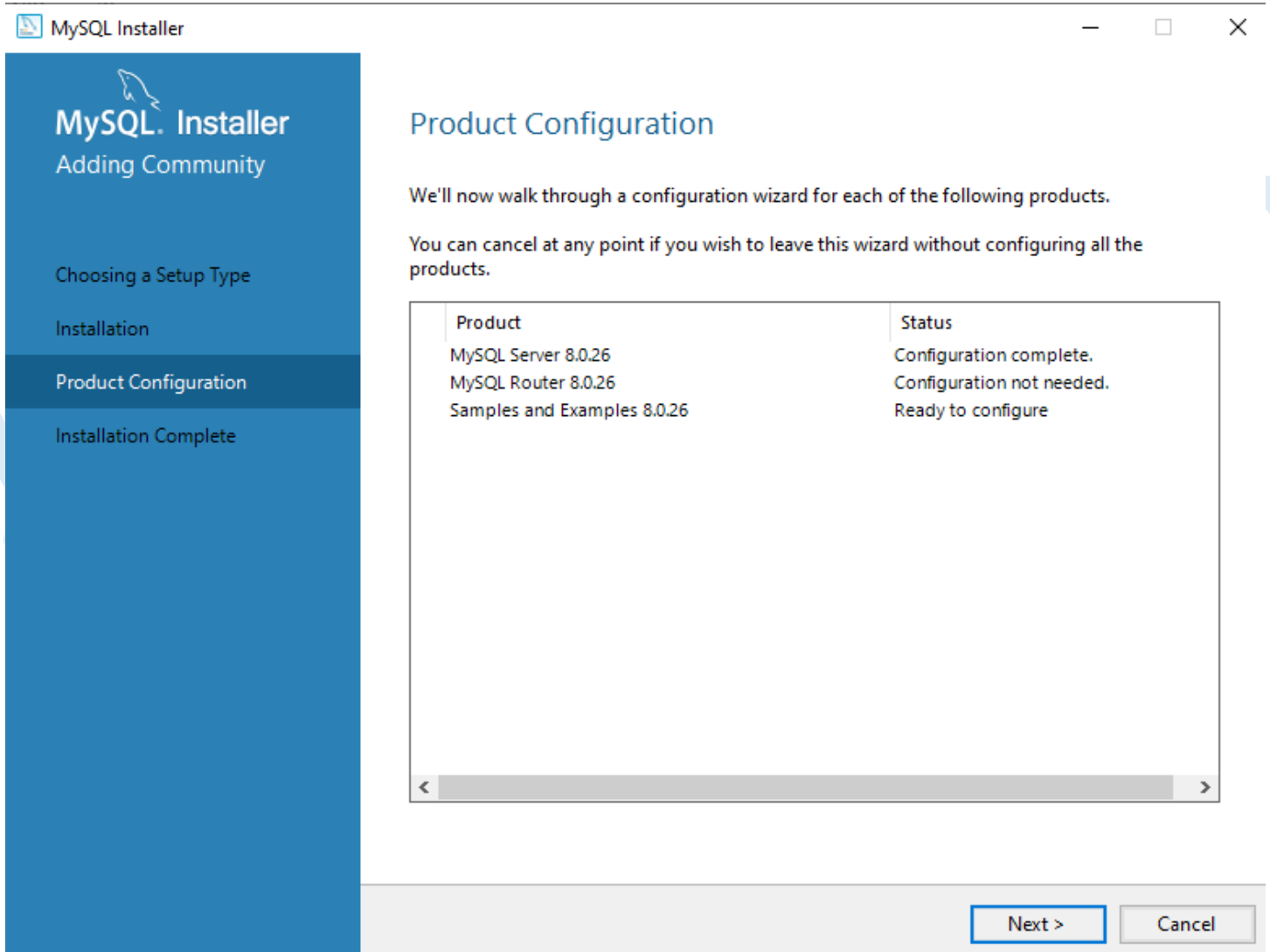
Installation

Select Finish



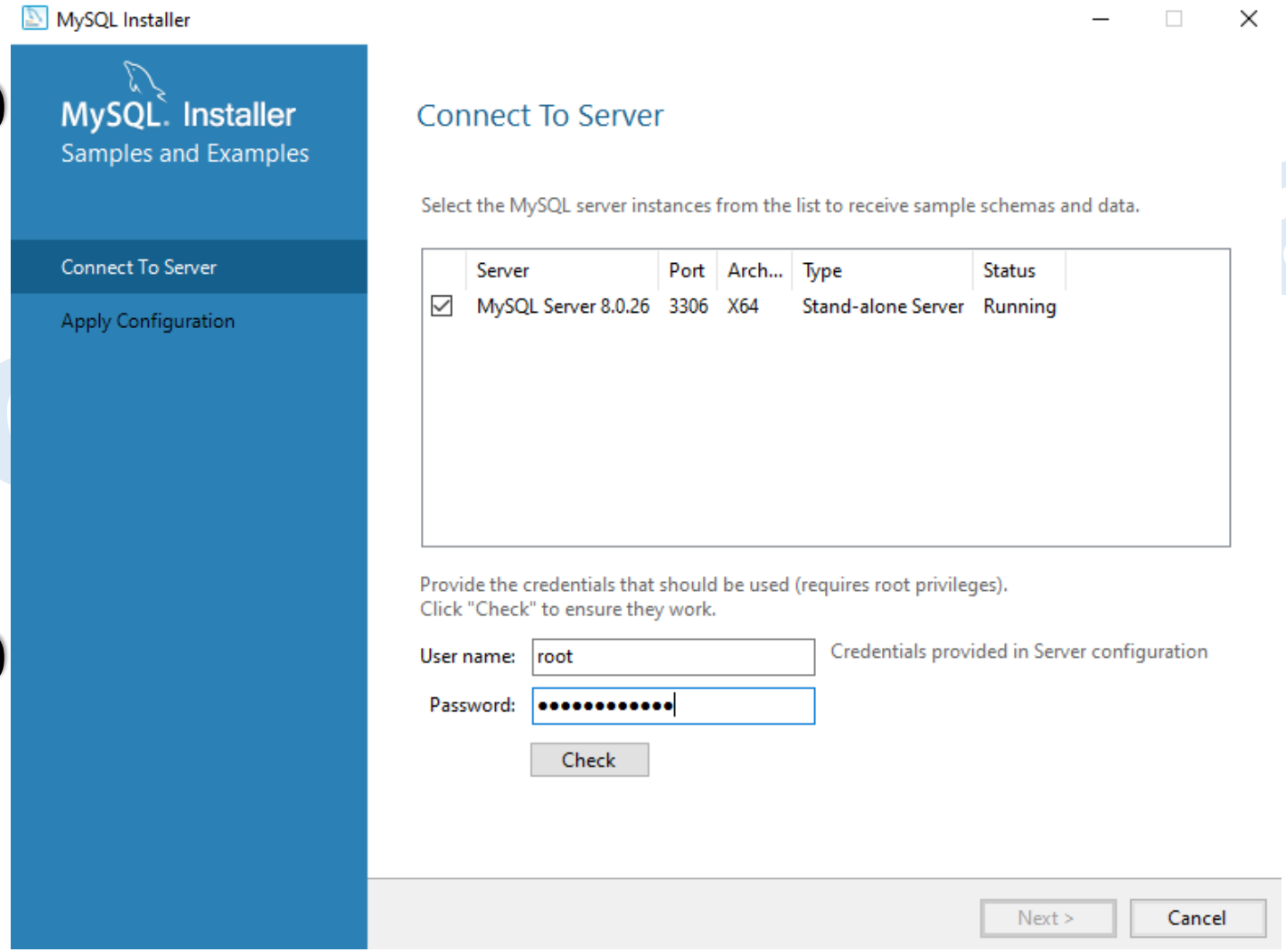
Installation

Select Next



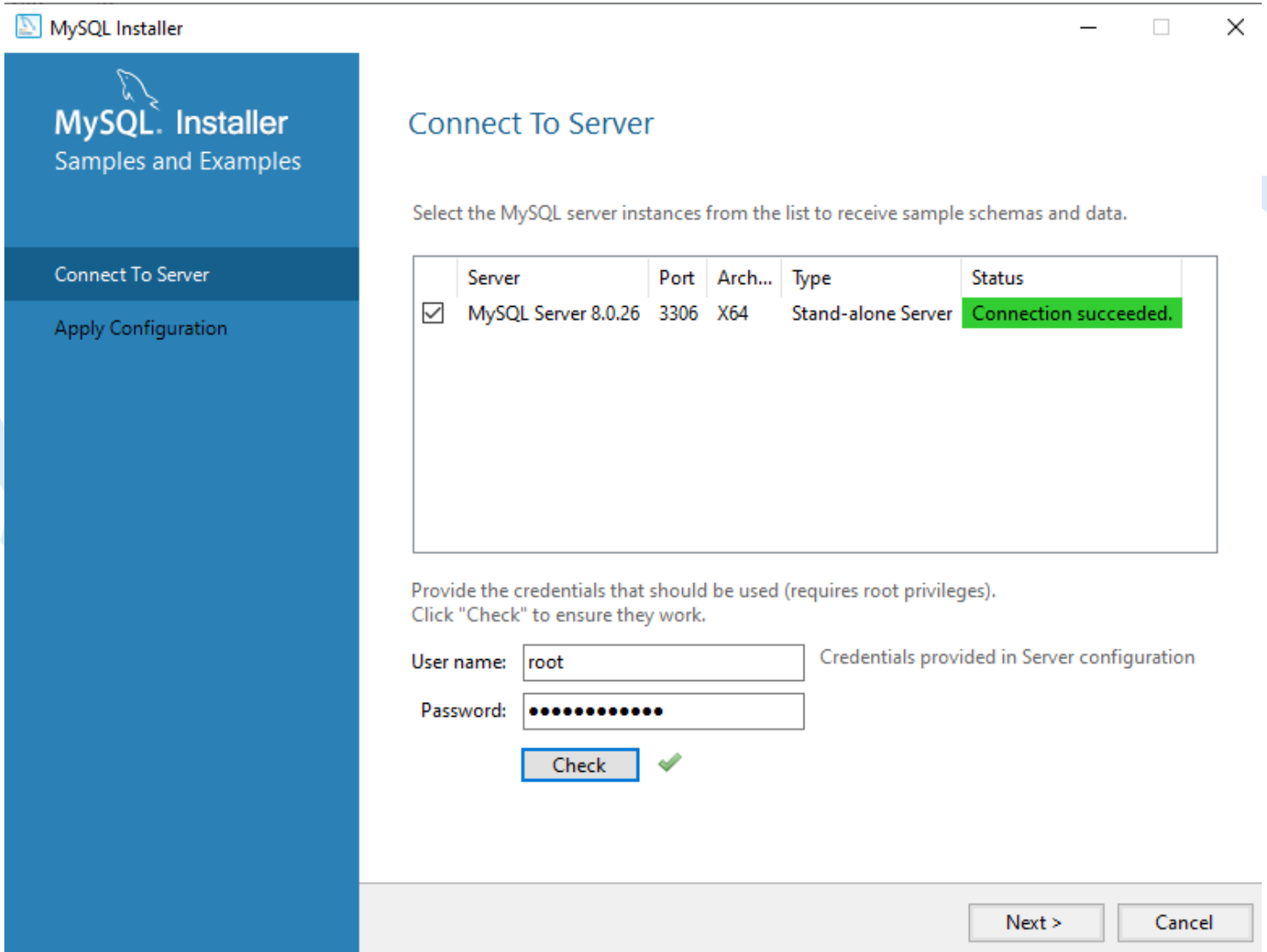
Installation

Type the same password you already entered in your Server Configuration stage



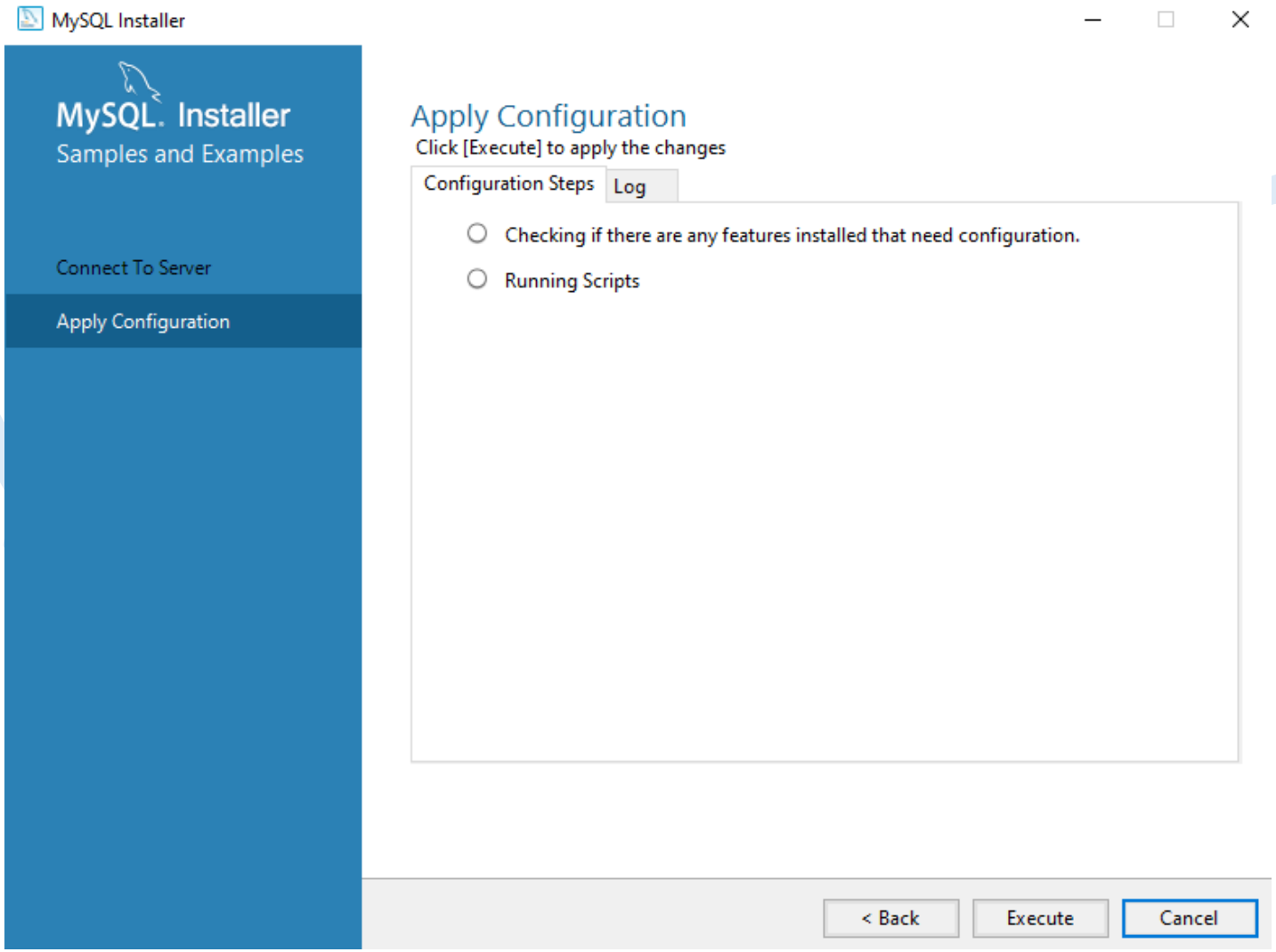
Installation

Select Check and then Next



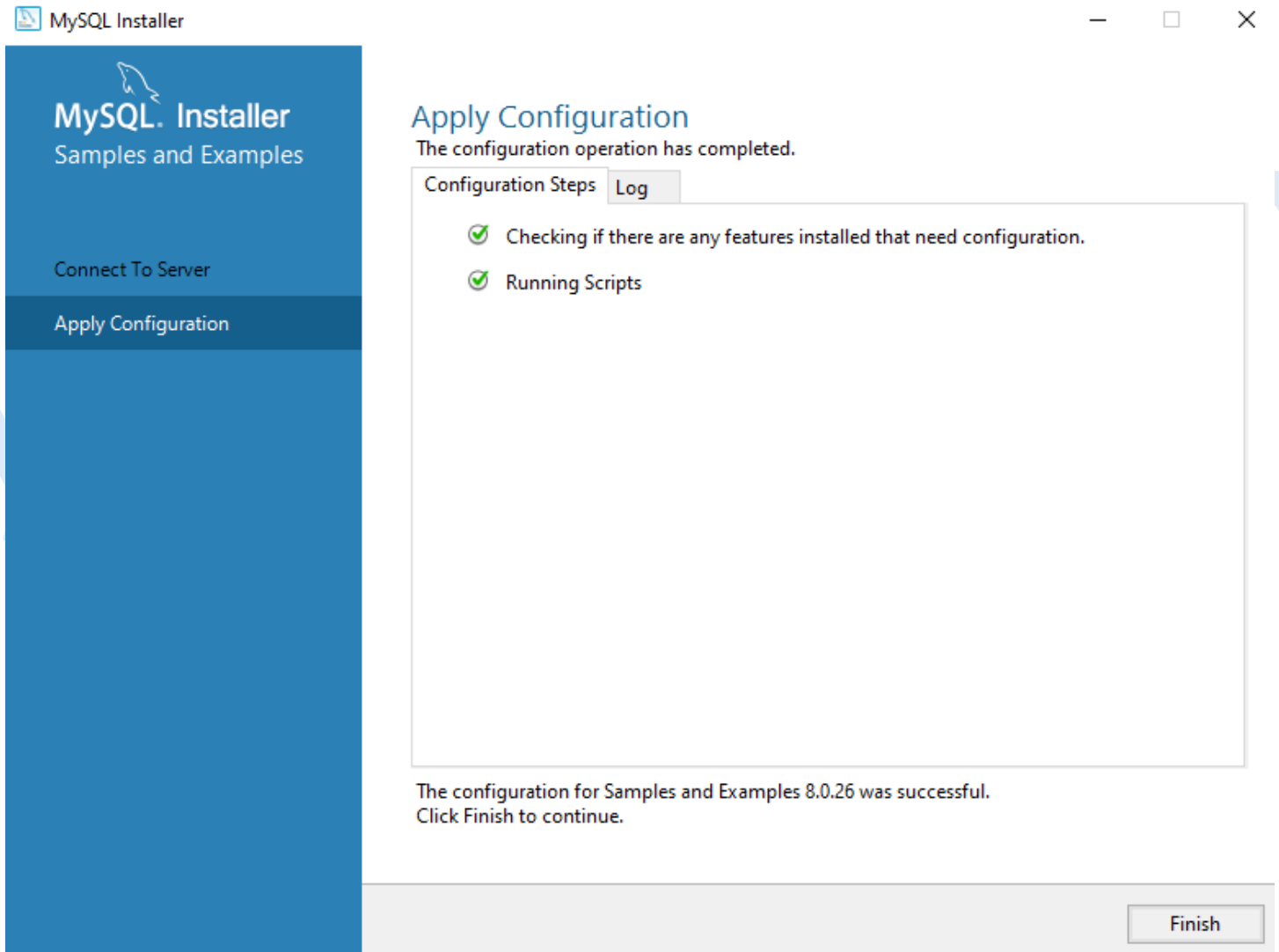
Installation

Select Execute



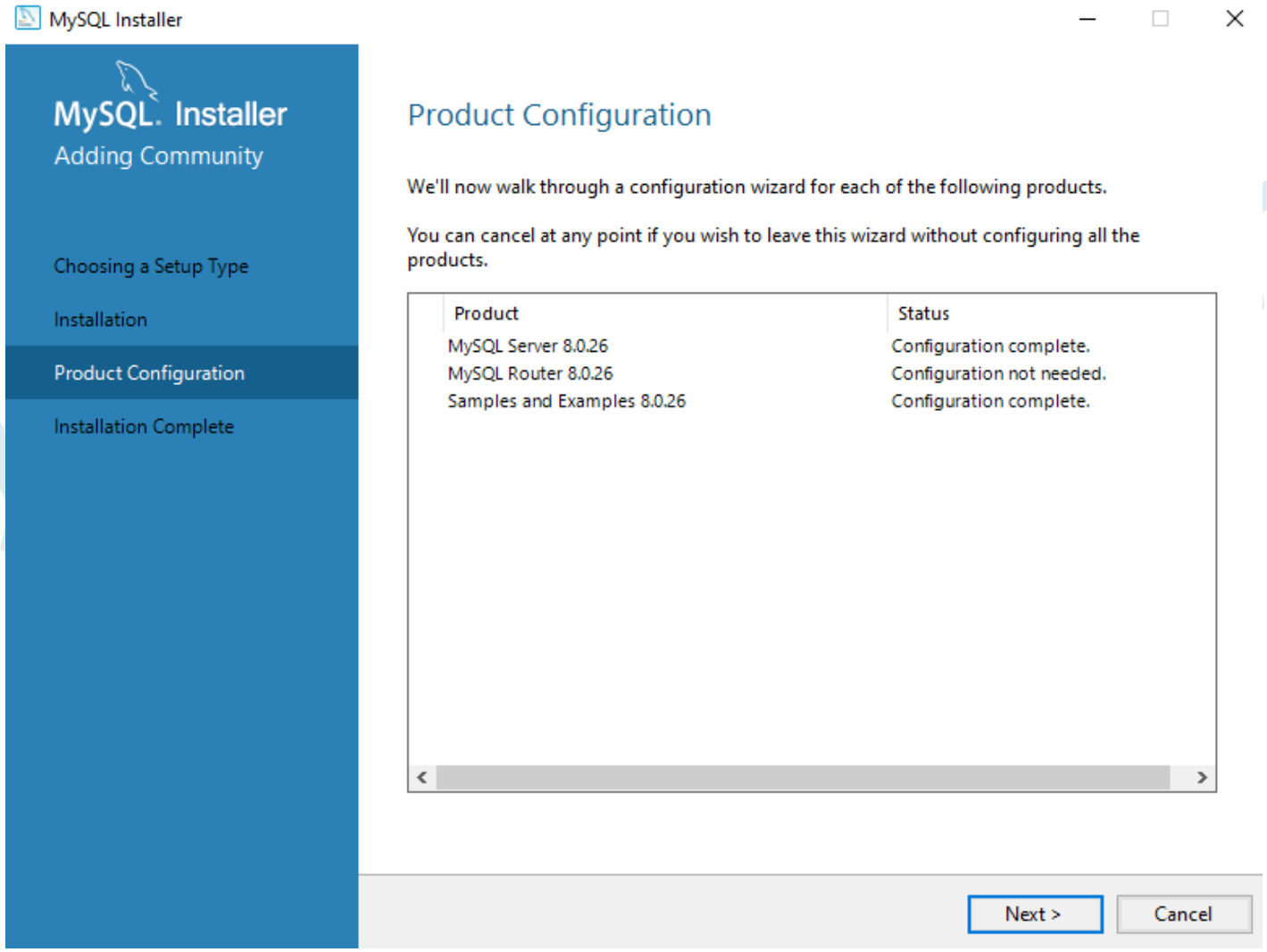
Installation

Select Finish



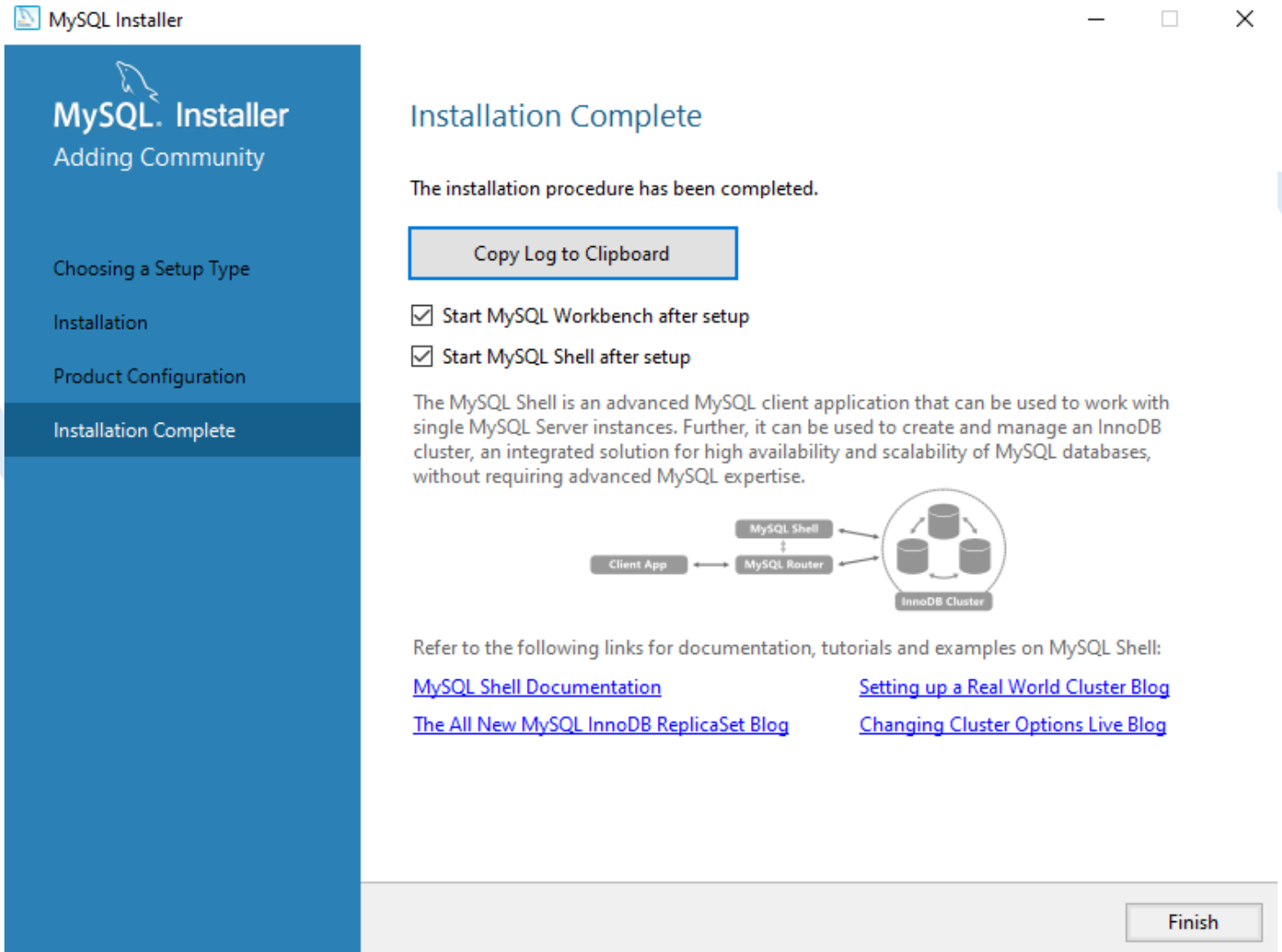
Installation

Select Next

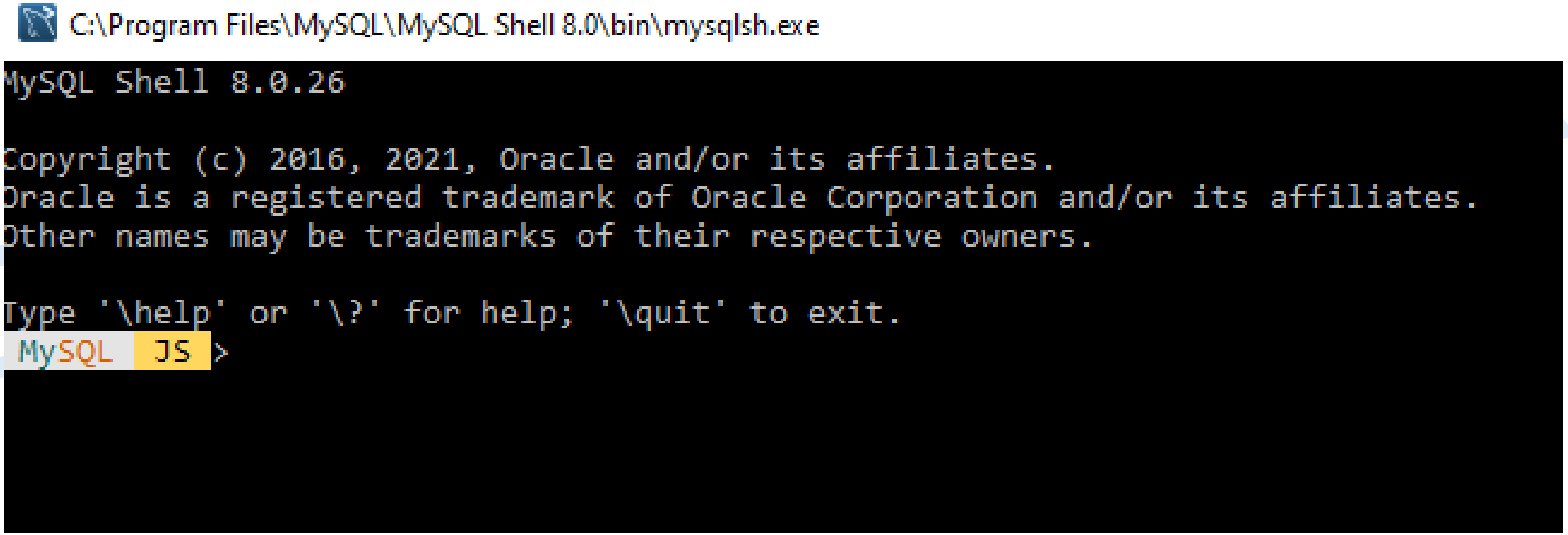


Installation

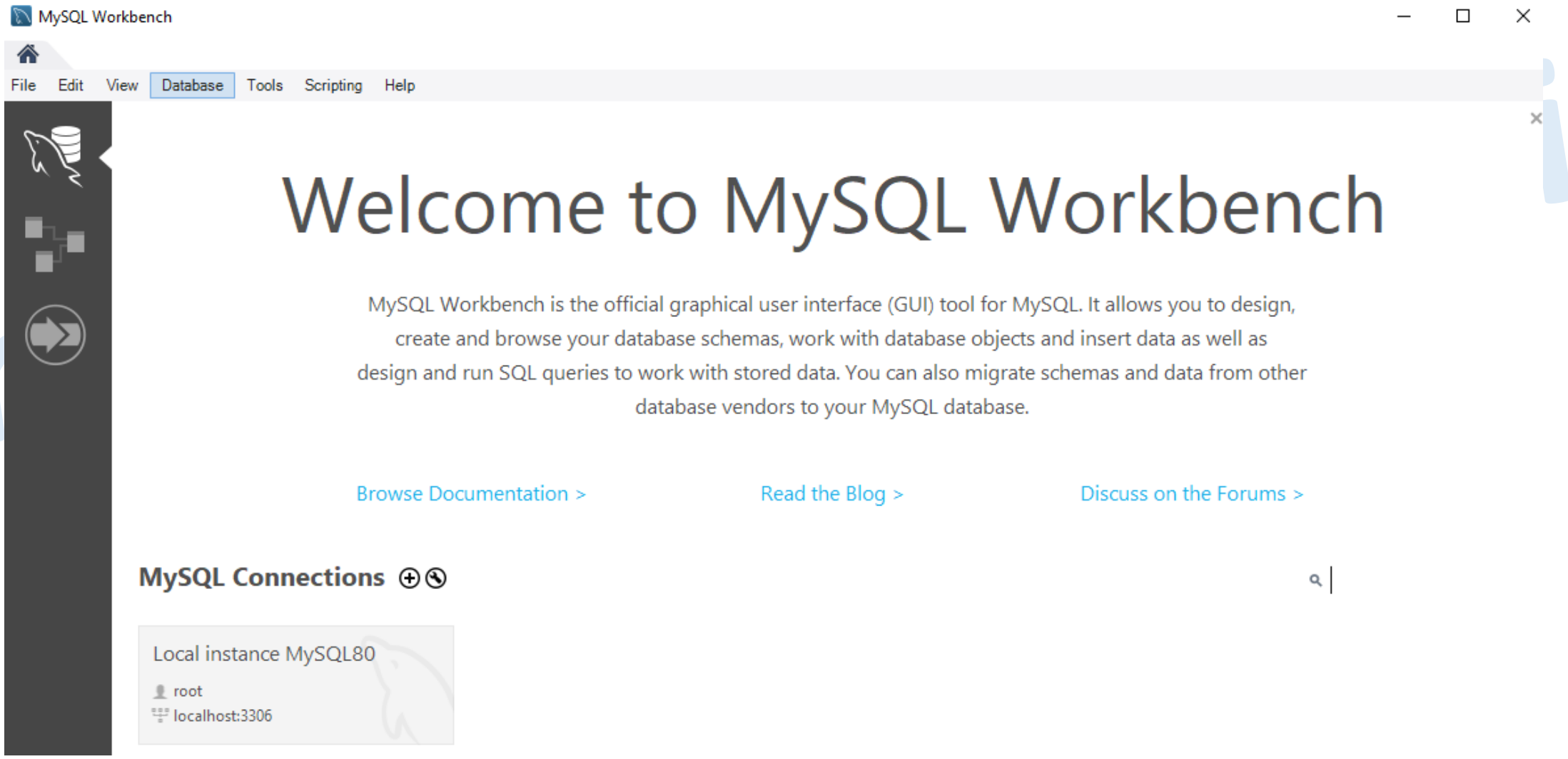
Select Finish



Installation: View of Shell



Installation: View of Workbench



SQL: Data Definition Language (DDL)

- The DDL commands in SQL are used to create database schema and to define the type and structure of the data that will be stored in a database.
- SQL DDL commands are further divided into the following major categories:
  - CREATE: The CREATE query is used to create a database or objects such as tables, views, stored procedures, etc.
  - ALTER: alters the structure of the existing database
  - DROP: delete objects from the database
  - TRUNCATE: remove all records from a table, including all spaces allocated for the records are removed

SQL: DDL: CREATE

- Database
  - `CREATE DATABASE LibraryDB;`
- Table
  - CREATE TABLE Books
    (
    Id INT (1),
    Name VARCHAR (50),
    Price INT (10)
    );

Data type

- A Data Type in SQL server is defined as the type of data that any column or variable can store.
- It is a type of data that an object holds like integer, character, string, etc.
- An SQL developer must decide what type of data that will be stored inside each column when creating a table.
- While creating any table or variable, in addition to specifying the name, you also set the Type of Data it will store.
- The data type is a guideline for SQL to understand what type of data is expected inside of each column, and it also identifies how SQL will interact with the stored data.
- In MySQL there are three main data types: string, numeric, and date and time.

Data Type: String

Data type	Description
CHAR(size)	A FIXED length string (can contain letters, numbers, and special characters). The *size* parameter specifies the column length in characters - can be from 0 to 255. Default is 1
VARCHAR(size)	A VARIABLE length string (can contain letters, numbers, and special characters). The *size* parameter specifies the maximum column length in characters - can be from 0 to 65535
BINARY(size)	Equal to CHAR(), but stores binary byte strings. The *size* parameter specifies the column length in bytes. Default is 1
VARBINARY(size)	Equal to VARCHAR(), but stores binary byte strings. The *size* parameter specifies the maximum column length in bytes.
TINYBLOB	For BLOBs (Binary Large OBjects). Max length: 255 bytes
TINYTEXT	Holds a string with a maximum length of 255 characters

Data Type: String

TEXT(size)	Holds a string with a maximum length of 65,535 bytes
BLOB(size)	For BLOBs (Binary Large OBjects). Holds up to 65,535 bytes of data
MEDIUMTEXT	Holds a string with a maximum length of 16,777,215 characters
MEDIUMBLOB	For BLOBs (Binary Large OBjects). Holds up to 16,777,215 bytes of data
LONGTEXT	Holds a string with a maximum length of 4,294,967,295 characters
LONGBLOB	For BLOBs (Binary Large OBjects). Holds up to 4,294,967,295 bytes of data
ENUM(val1, val2, val3, ...)	A string object that can have only one value, chosen from a list of possible values. You can list up to 65535 values in an ENUM list. If a value is inserted that is not in the list, a blank value will be inserted. The values are sorted in the order you enter them
SET(val1, val2, val3, ...)	A string object that can have 0 or more values, chosen from a list of possible values. You can list up to 64 values in a SET list

Data Type: Numeric

Data type	Description
BIT(*size*)	A bit-value type. The number of bits per value is specified in *size*. The *size* parameter can hold a value from 1 to 64. The default value for *size* is 1.
TINYINT(*size*)	A very small integer. Signed range is from -128 to 127. Unsigned range is from 0 to 255. The *size* parameter specifies the maximum display width (which is 255)
BOOL	Zero is considered as false, nonzero values are considered as true.
BOOLEAN	Equal to BOOL
SMALLINT(*size*)	A small integer. Signed range is from -32768 to 32767. Unsigned range is from 0 to 65535. The *size* parameter specifies the maximum display width (which is 255)
MEDIUMINT(*size*)	A medium integer. Signed range is from -8388608 to 8388607. Unsigned range is from 0 to 16777215. The *size* parameter specifies the maximum display width (which is 255)
INT(*size*)	A medium integer. Signed range is from -2147483648 to 2147483647. Unsigned range is from 0 to 4294967295. The *size* parameter specifies the maximum display width (which is 255)

Data Type: Numeric

INTEGER(*size*)	Equal to INT(size)
BIGINT(*size*)	A large integer. Signed range is from -9223372036854775808 to 9223372036854775807. Unsigned range is from 0 to 18446744073709551615. The *size* parameter specifies the maximum display width (which is 255)
FLOAT(*size*, *d*)	A floating point number. The total number of digits is specified in *size*. The number of digits after the decimal point is specified in the *d* parameter. This syntax is deprecated in MySQL 8.0.17, and it will be removed in future MySQL versions
FLOAT(*p*)	A floating point number. MySQL uses the *p* value to determine whether to use FLOAT or DOUBLE for the resulting data type. If *p* is from 0 to 24, the data type becomes FLOAT(). If *p* is from 25 to 53, the data type becomes DOUBLE()
DOUBLE(*size*, *d*)	A normal-size floating point number. The total number of digits is specified in *size*. The number of digits after the decimal point is specified in the *d* parameter
DOUBLE PRECISION(*size*, *d*)	
DECIMAL(*size*, *d*)	An exact fixed-point number. The total number of digits is specified in *size*. The number of digits after the decimal point is specified in the *d* parameter. The maximum number for *size* is 65. The maximum number for *d* is 30. The default value for *size* is 10. The default value for *d* is 0.
DEC(*size*, *d*)	Equal to DECIMAL(size,d)

Data Type: Date and Time

Data type	Description
DATE	A date. Format: YYYY-MM-DD. The supported range is from '1000-01-01' to '9999-12-31'
DATETIME(*fsp*)	A date and time combination. Format: YYYY-MM-DD hh:mm:ss. The supported range is from '1000-01-01 00:00:00' to '9999-12-31 23:59:59'. Adding DEFAULT and ON UPDATE in the column definition to get automatic initialization and updating to the current date and time
TIMESTAMP(*fsp*)	A timestamp. TIMESTAMP values are stored as the number of seconds since the Unix epoch ('1970-01-01 00:00:00' UTC). Format: YYYY-MM-DD hh:mm:ss. The supported range is from '1970-01-01 00:00:01' UTC to '2038-01-09 03:14:07' UTC. Automatic initialization and updating to the current date and time can be specified using DEFAULT CURRENT_TIMESTAMP and ON UPDATE CURRENT_TIMESTAMP in the column definition
TIME(*fsp*)	A time. Format: hh:mm:ss. The supported range is from '-838:59:59' to '838:59:59'
YEAR	A year in four-digit format. Values allowed in four-digit format: 1901 to 2155, and 0000. MySQL 8.0 does not support year in two-digit format.

Primary Key Constraint

- The PRIMARY KEY constraint uniquely identifies each record in a table.
- Primary keys must contain UNIQUE values, and cannot contain NULL values.
- A table can have only ONE primary key; and in the table, this primary key can consist of single or multiple columns (fields).

SQL: DDL: ALTER

- The ALTER command in SQL DDL is used to modify the structure of an already existing table.
- Add primary key:
  - ALTER TABLE Books
    ADD PRIMARY KEY (Id);
- Add new column:
  - ALTER TABLE Books
    ADD Year year;

```
ALTER TABLE Books
ADD Publisher varchar(50),
ADD AuthorName varchar(50);
```

- Modify the data type of a column:
  - ALTER TABLE Books
    MODIFY COLUMN Price float(10,2);

SQL: DDL: ALTER

- Modify the column name:
  - ALTER TABLE Books
    RENAME COLUMN AuthorName TO FirstName,
    ADD LastName varchar(50);
- Modify table name:
  - ALTER TABLE Books RENAME Book_Details;
- Drop a column:
  - ALTER TABLE Book_Details
    DROP COLUMN Publisher;
- Add NOT NULL constraint:
  - ALTER TABLE Book_Details
    MODIFY Name varchar(50) NOT NULL;

SQL: DDL: DROP and TRUNCATE

- Drop a column:
  - ALTER TABLE Book_Details
    DROP COLUMN Publisher;
- The DROP TABLE statement is used to drop an existing table in a database.
  - DROP TABLE Book_Details;
- Drop the database:
  - DROP DATABASE libraryDB;
- The TRUNCATE TABLE statement is used to delete the data inside a table, but not the table itself.
  - TRUNCATE TABLE Book Details;

SQL: DML

- DML is short name of **Data Manipulation Language** which deals with data manipulation and includes most common SQL statements such as
  - SELECT: Used to query or fetch selected fields or columns from a database table.
  - INSERT: Used to insert new data records or rows in the database table
  - UPDATE: Used to set the value of a field or column for a particular record to a new value
  - DELETE: Used to remove one or more rows from the database table

SQL: DML: INSERT

- Specify both the column names and the values to be inserted:
  - INSERT INTO *table_name* (*column1*, *column2*, *column3*, ...)
    VALUES (*value1*, *value2*, *value3*, ...);
- If you are adding values for all the columns of the table, you do not need to specify the column names in the SQL query.
  - INSERT INTO *table_name*
    VALUES (*value1*, *value2*, *value3*, ...);
- Insert data in specific columns.
  - INSERT INTO *table_name* (*column2*, *column5*, *column6*)
    VALUES (*value1*, *value2*, *value3*);

SQL: DML: SELECT and SELECT DISTINCT

- The field names of the table you want to select data from:
  - SELECT *column1*, *column2, ...*
    FROM *table_name*;
- If you want to select all the fields available in the table, use the following syntax:
  - SELECT * FROM *table_name*;
- We can also populate one table using another table with the help of select statement. The only condition is that the table must have the same sets of attributes.
  - Insert into table_no_first [(column1, column 2…column n)]
    select column1, column 2…column n from table_no_two;
- Used to return only distinct (different) values.
  - SELECT DISTINCT *column1, column2, ...*
    FROM *table_name*;

SQL: DML: SELECT with where

- used to extract only those records that fulfill a specified condition.
  - SELECT *column1*, *column2, …*
    FROM *table_name*
    WHERE *condition*;
- SQL requires single quotes around text values (most database systems will also allow double quotes).
- However, numeric fields should not be enclosed in quotes.
- Operators → =, >, <, <>, >=, <=, BETWEEN, LIKE, IN

SQL: DML: SELECT with AND, OR, NOT

- The AND operator displays a record if all the conditions separated by AND are TRUE.
  - SELECT *column1*, *column2, ...*
    FROM *table_name*
    WHERE *condition1* AND *condition2* AND *condition3 ...*;
- The OR operator displays a record if any of the conditions separated by OR is TRUE.
  - SELECT *column1*, *column2, ...*
    FROM *table_name*
    WHERE *condition1* OR *condition2* OR *condition3 ...*;
- The NOT operator displays a record if the condition(s) is NOT TRUE.
  - SELECT *column1*, *column2, ...*
    FROM *table_name*
    WHERE NOT *condition*;
- Combine:
  - SELECT * FROM students WHERE student_name LIKE 'r%' AND (course='C' OR roll_no between 2 and 4);

SQL: DML: SELECT with ORDER BY

- Used to sort the result-set in ascending or descending order.
- Sorts the records in ascending order by default. To sort the records in descending order, use the DESC keyword.
  - SELECT *column1*, *column2, ...*
    FROM *table_name*
    ORDER BY *column1, column2, ...* ASC|DESC;
- ORDER BY Several Columns: Means that it orders by column1, but if some rows have the same column1 value, it orders them by column2:
  - SELECT * FROM *table_name*
    ORDER BY *column1, column2*;
  - SELECT * FROM *table_name*
    ORDER BY *column1* ASC, *column2* DESC;

SQL: DML: SELECT with GROUP BY

- groups rows that have the same values into summary rows
  - ```
    SELECT column_name(s)
    FROM table_name
    WHERE condition
    GROUP BY column_name(s)
    ORDER BY column_name(s);
    ```
  - ```
    SELECT COUNT(roll_no), course
    FROM students
    GROUP BY course; #lists the number of students in each course
    ```
  - ```
    SELECT COUNT(roll_no), course
    FROM students
    GROUP BY course
    ORDER BY COUNT(roll_no) DESC; #number of students in each course, sorted high to low
    ```
    ```

SQL: DML: SELECT with NULL and NOT NULL

- SELECT *column_names*
  FROM *table_name*
  WHERE *column_name* IS NULL;

- SELECT *column_names*
  FROM *table_name*
  WHERE *column_name* IS NOT NULL;

SQL: DML: SELECT with HAVING

- The HAVING clause was added to SQL because the WHERE keyword cannot be used with aggregate functions.
  - ```
    SELECT column_name(s)
    FROM table_name
    WHERE condition
    GROUP BY column_name(s)
    HAVING condition
    ORDER BY column_name(s);
    ```
  - ```
    SELECT COUNT(roll_no), course
    FROM students
    GROUP BY course
    HAVING COUNT(roll_no) > 1; #lists the number of students in each course with more than 1 student
    ```

SQL: DML: SELECT with Aggregate Functions

- MIN() Syntax: returns the smallest value of the selected column
  - SELECT MIN(*column_name*) FROM *table_name* WHERE *condition*;
- MAX() Syntax: returns the largest value of the selected column
  - SELECT MAX(*column_name*) FROM *table_name* WHERE *condition*;
- COUNT() Syntax: returns the number of rows that matches a specified criterion
  - SELECT COUNT(*column_name*) FROM *table_name* WHERE *condition*;
- AVG() Syntax: returns the average value of a numeric column
  - SELECT AVG(*column_name*) FROM *table_name* WHERE *condition*;
- SUM() Syntax: returns the total sum of a numeric column
  - SELECT SUM(*column_name*) FROM *table_name* WHERE *condition*;

SQL: DML: UPDATE

- UPDATE *table_name*
  SET *column1 = value1, column2 = value2, ...*
  WHERE *condition*;
- UPDATE Multiple Records: update the column1 to value1 for all records where the condition is true
  - UPDATE *table_name*
    SET *column1 = value1*
    WHERE *condition*;
- Warning
  - Be careful when updating records. If you omit the WHERE clause, ALL records will be updated!
  - UPDATE *table_name*
    SET *column1 = value1*;

SQL: DML: DELETE

- `DELETE FROM table_name WHERE condition;`
- `DELETE FROM table_name;` [to delete all records]