MySQL Installation and Usage on Windows 10

Author: Sri Adilakshmi M

Table of Contents

This document outlines the steps needed to install MySQL and access various MySQL utilities on Windows Operating system.

1. Overview:

MySQL is the most popular open-source relational database management system (RDBMS) that uses Structured Query Language (SQL) for managing and manipulating data. It is known for its speed, reliability, scalability and ease of use, widely used in global services and major web applications like Facebook, Netflix, Uber, etc.

MySQL is named after co-founder Monty Widenius's daughter, **My** and initially developed by the Swedish company named **MySQL AB** which was bought by **Sun Microsystem**s. Later, in 2010, when Oracle Corporation acquired Sun Microsystems, the MySQL is being sponsored by **Oracle** since then.

MySQL works on a client-server architecture such that a **client** (a web application or command line tool) sends a request to the **MySQL server** which processes the request, optimizes the query, interacts with the storage engine to read data from disk and generates result after which the server sends the results back to the client application.

MySQL is a popular choice due to its key features:

- MySQL is free and open source allowing modification of MySQL software under GPL (General Public License).
- It uses a standard programming language for database called **SQL**, allowing users to work on data effectively.
- It is written in C and C++ language and runs on multiple operating systems including Windows, Linux, and macOS, and works with various programming languages such as Java, PHP, Perl, Python, etc.
- It provides transactional and non-transactional storage engines like InnoDB (default), MyISAM for flexible use.
- When using the InnoDB storage engine, it supports full ACID (Atomicity, Consistency, Isolation, Durability) properties to ensure data integrity, reliability and accurate data processing.
- It handles high traffic and supports large databases *(more than 50 million records in a table with default file size limit of 4 GB which can be extended to 8 million TB, supporting 64 indexes per table with each index consists of up to 16 columns)* using features like partitioning, replication and clustering to ensure high performance and availability.
- It includes several client programs including **MySQL Workbench** GUI, and command-line utilities like `mysql`, `mysqlsh`, `mysqladmin`, `mysqldump`, `mysqlimport`, `mysqlshow`, `mysqlcheck`, etc.

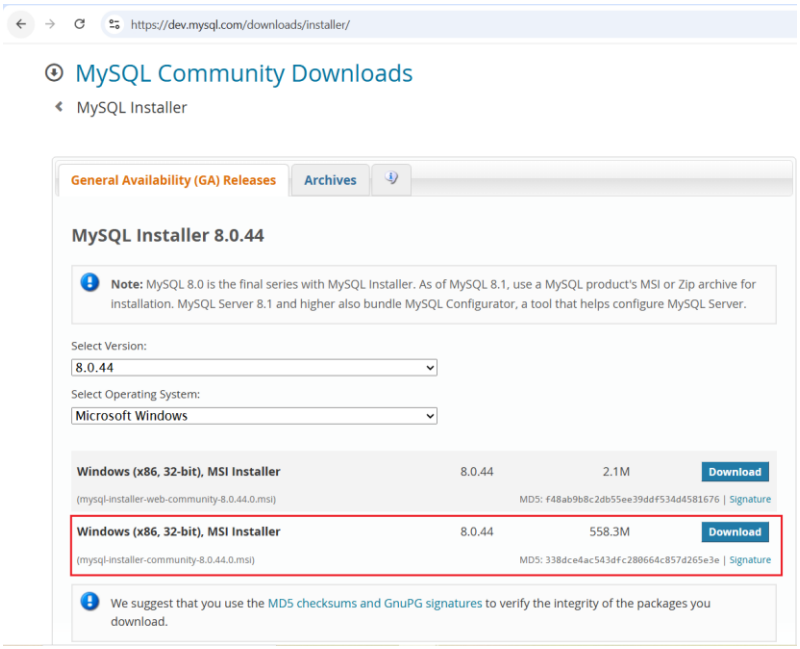
Common use cases of MySQL include:

- **Web Applications:** MySQL is widely used as backend in many websites and online services to manage user data, content and session information.
- **Content Management Systems (CMS):** MySQL is a core component of the popular LAMP (Linux, Apache, MySQL, PHP/Python/Perl) software stack, forming the backend for content management systems like WordPress, Joomla, and Drupal.
- **E-commerce & Social Platforms:** Large-scale transactional systems and social networks rely on MySQL for managing product catalogs, customer details, user profiles, content, and financial data.
- **Data Analytics & Reporting:** Organizations use MySQL to store and analyze large datasets to gain business insights.

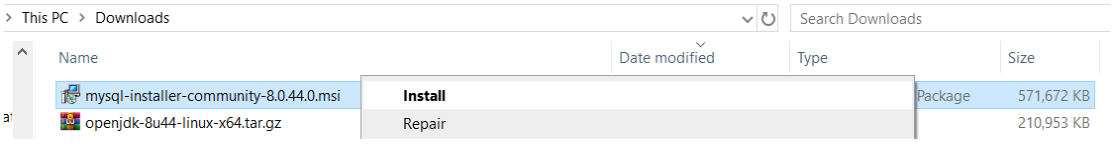
Considering the heavy usage of MySQL, many cloud providers offer managed services of MySQL such as **Amazon RDS** (offered by Amazon Web Services), **Google Cloud SQL** (offered by Google Cloud Platform), **Azure Database** (offered by Microsoft cloud), **MySQL Heatwave** (offered by Oracle cloud) etc. to simply database management by automating patching, backups and scaling.

2. Install MySQL Server:

Go to MySQL Installer Downloads website, select the latest version *(at the time of this document writing, the latest version was 8.0.44)* and click on **Download** button to download the MSI Installer community file which gets saved into your **Downloads** folder.

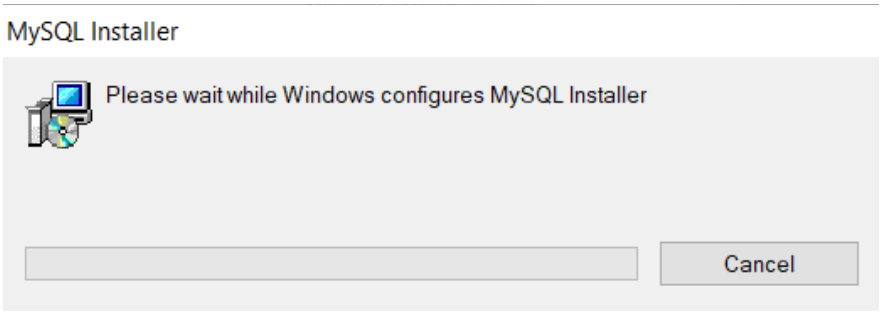


After the file is downloaded, go to your **Downloads** folder and right click on `mysql-installer-community-*.msi` and select **Install** option.

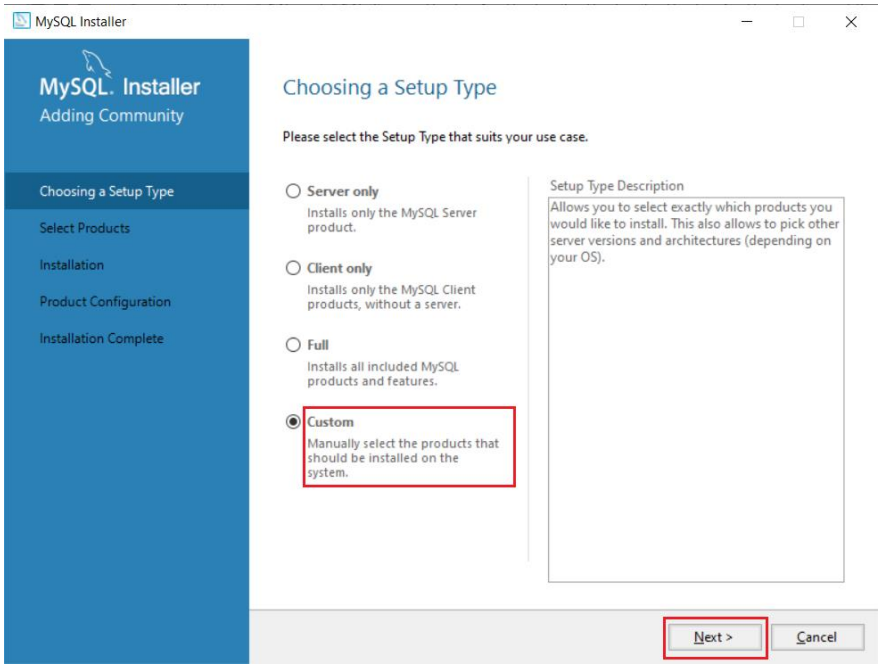


It starts with preparing for installation.

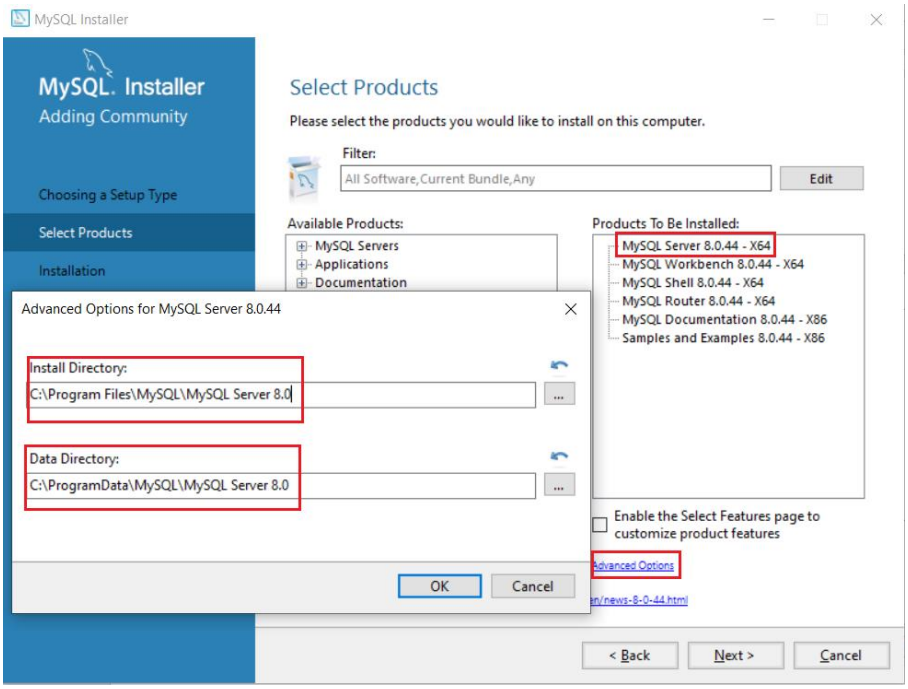
Then it configures MySQL Installer to open up.



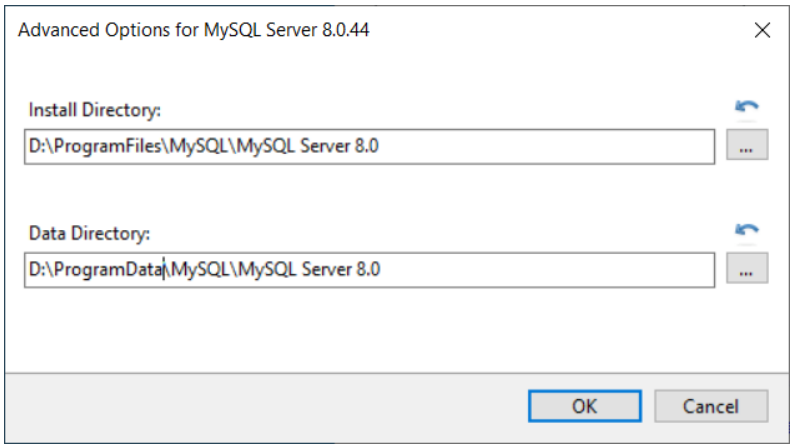
On the **MySQL Installer** wizard, choose the **Setup Type** as **Custom** to install all MySQL products. Then press **Next** to continue.



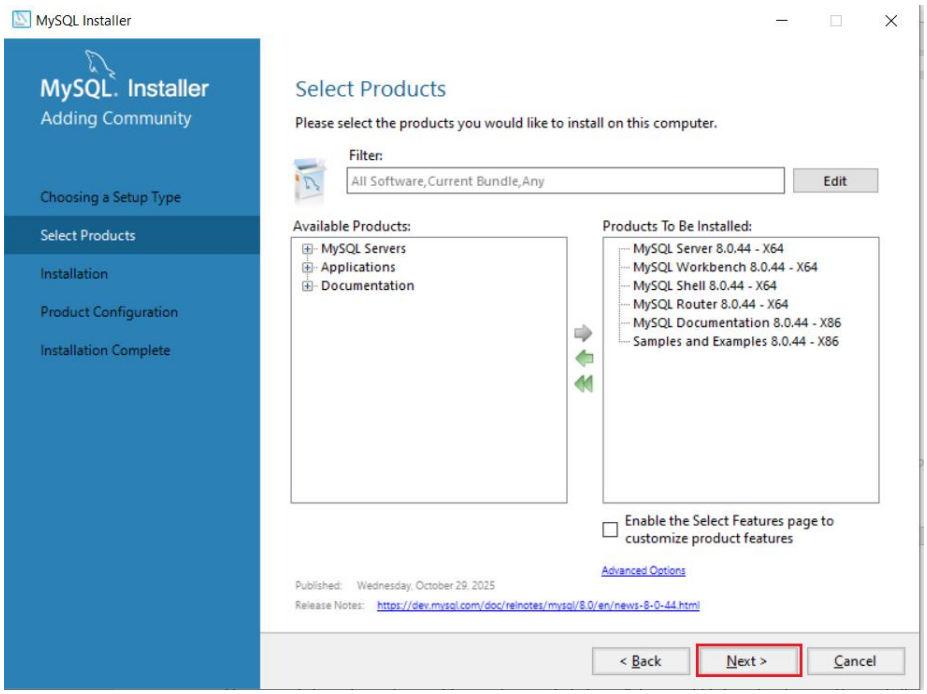
On the **Select Products** tab, it displays the list of products like MySQL Server, MySQL Workbench, MySQL Shell, etc. to be installed. Select **MySQL Server** product and click on **Advanced options** where you see the default **Install Directory** pointing to `C:\Program Files\MySQL\MySQL Server 8.0` and **Data Directory** pointing to `C:\ProgramData\MySQL\MySQL Server 8.0`.



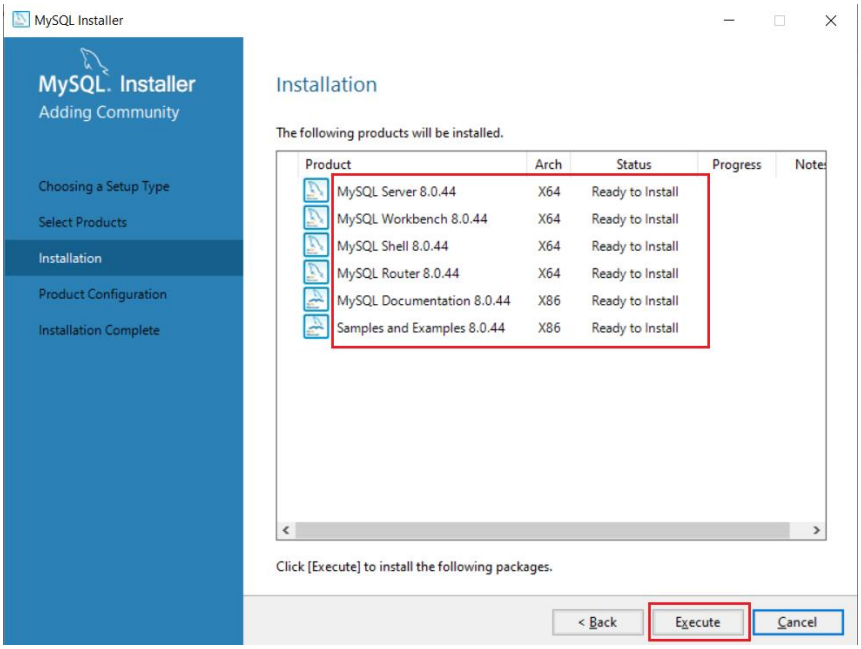
Change the default paths if needed. Here, I am changing from `C:\Program Files` to `D:\ProgramFiles` and `C:\ProgramData` to `D:\ProgramData` path to install in D drive instead of C drive.



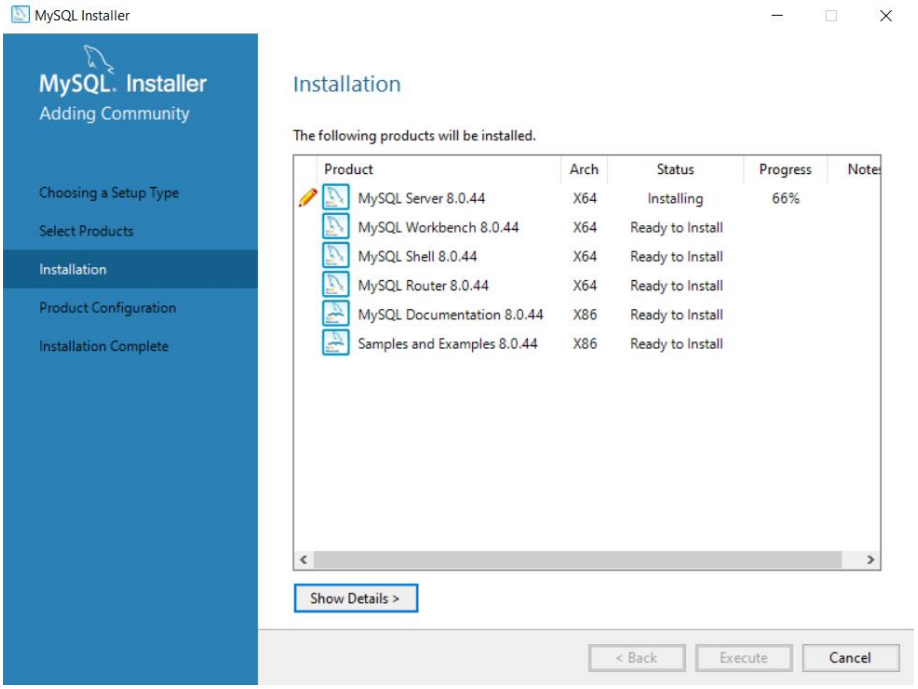
Similarly, select other products like MySQL Workbench, MySQL Shell, MySQL Router, MySQL Documentation, Samples and Examples, and click on **Advanced Options** to change the install directory if needed. Then, press on **Next** to continue.



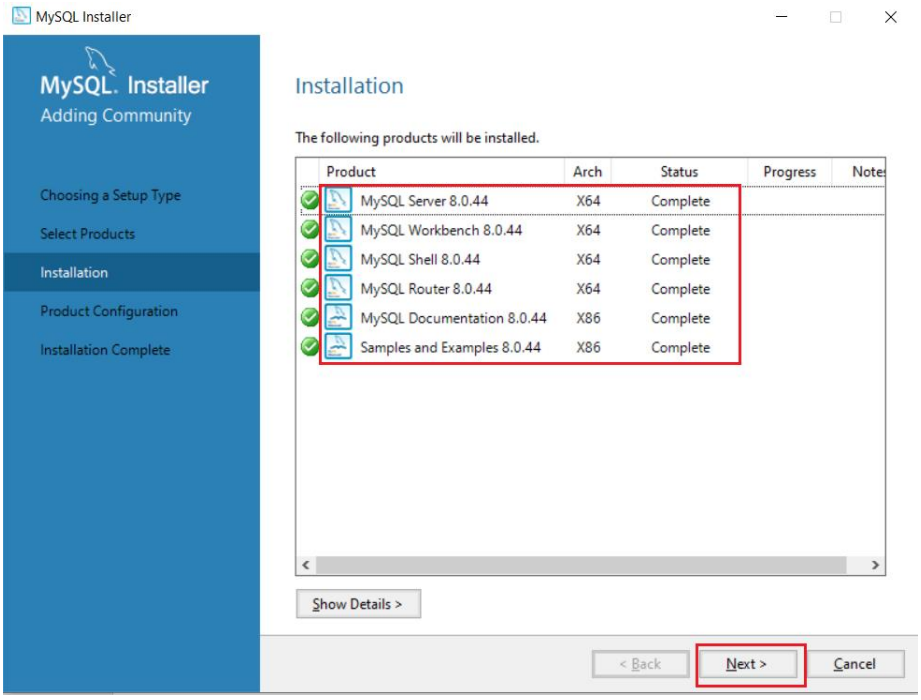
On the **Installation** tab, it displays the list of products to be installed along with their status *(which is set to **Ready to install** initially)*. Press **Execute** to start the installation.



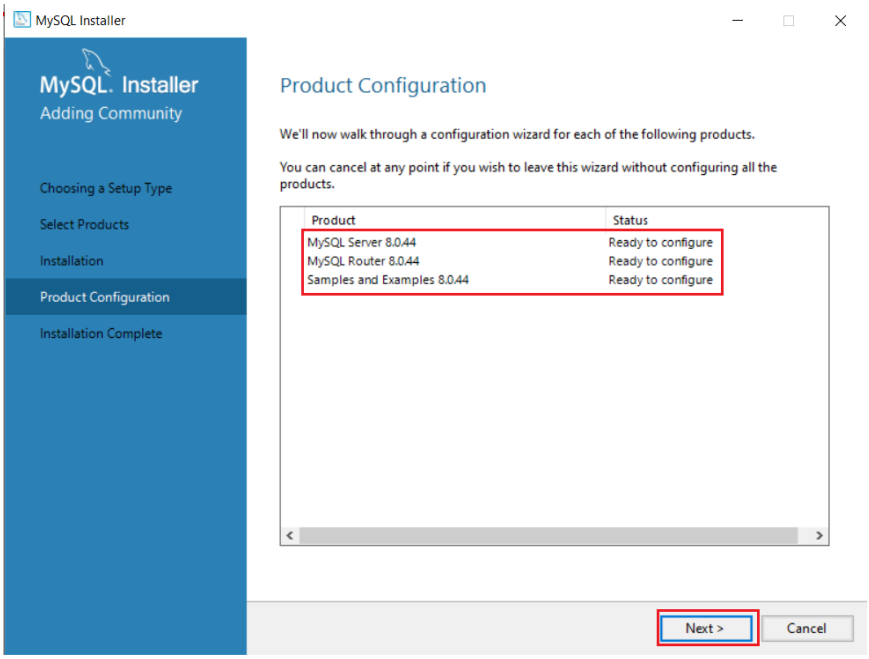
It begins with the installation of **MySQL Server**.



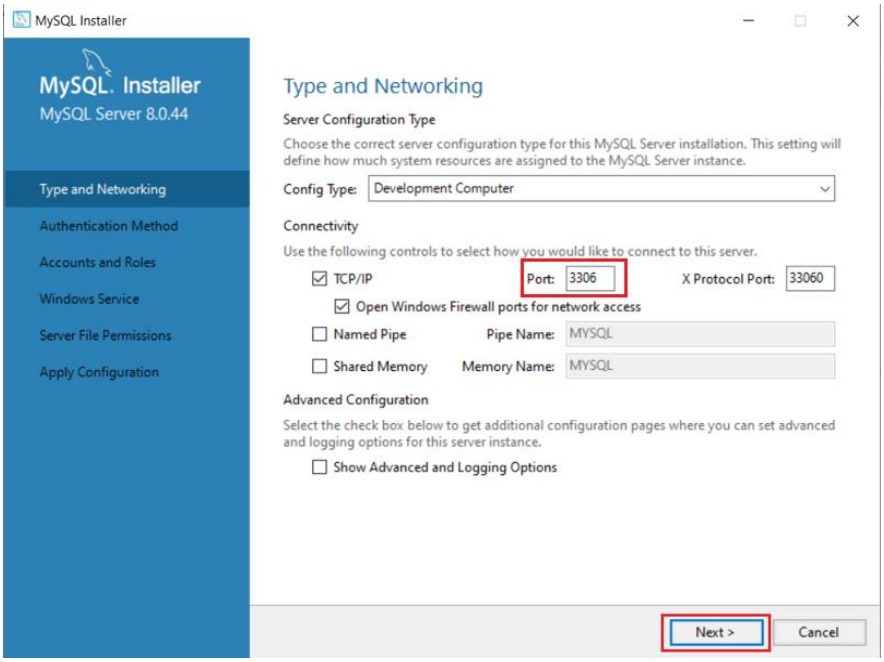
In few minutes, status of all products changes to **Complete**. Press **Next** to continue.



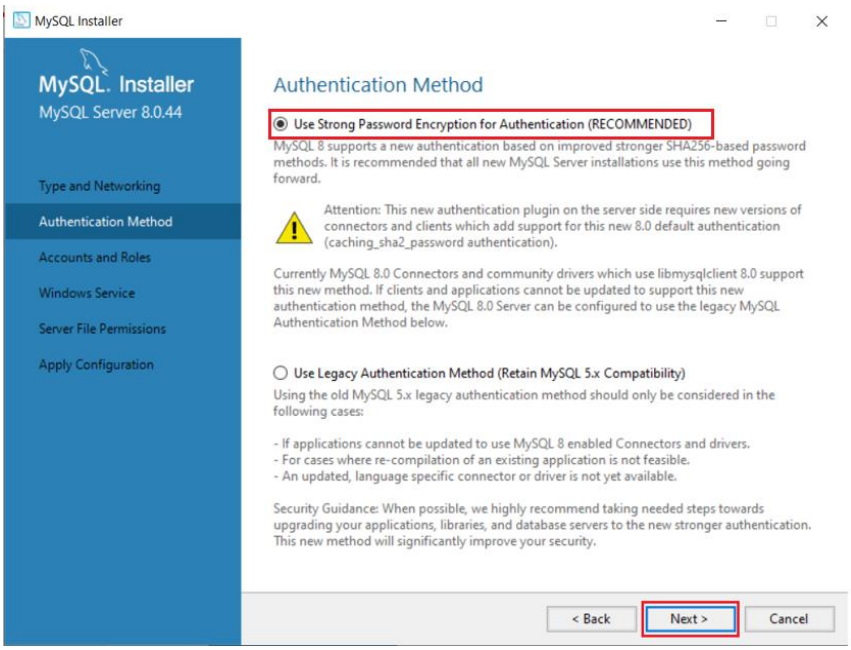
On the **Product Configuration** tab, it displays the list of products to be configured along with their status *(which is set to **Ready to configure** initially)*. Press **Next** to continue with MySQL Server configuration.



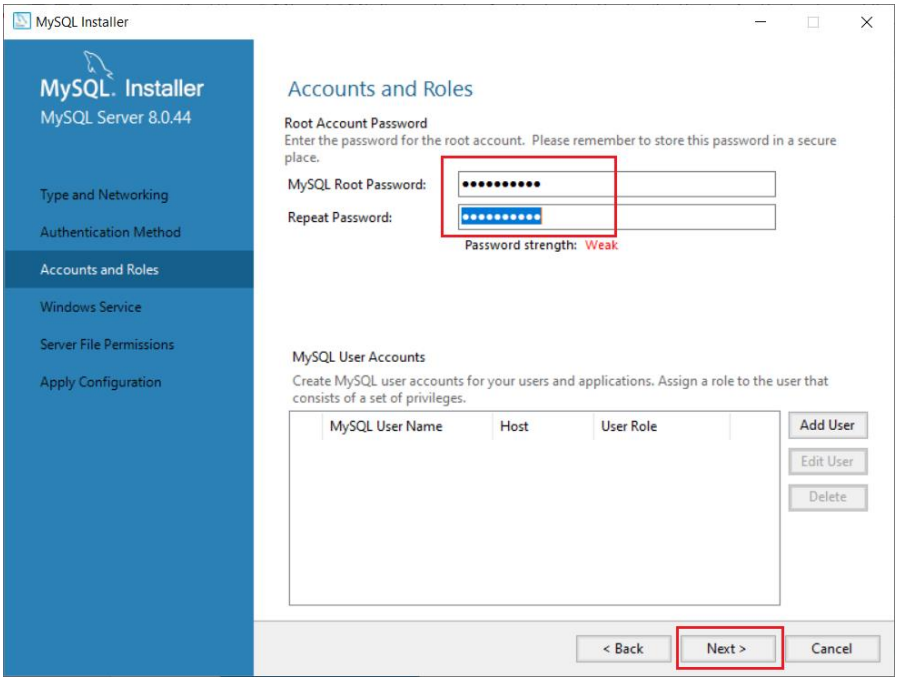
On the **Type and Networking** tab, it displays the network port for each configuration type (the default port for MySQL server is **3306**). Accept the default values or change port number as needed and press **Next** to continue.



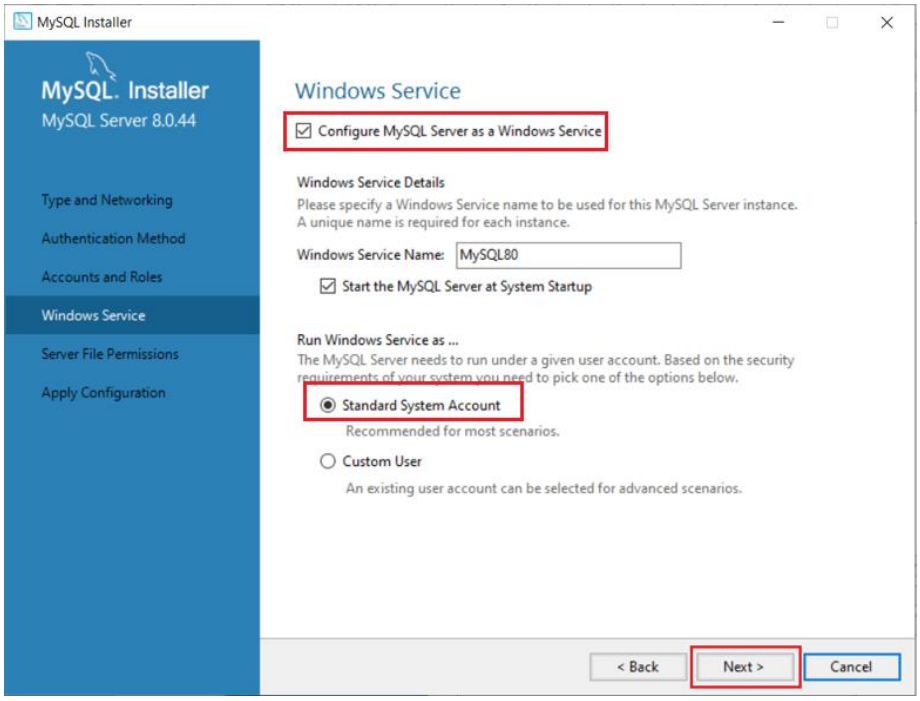
On the **Authentication Method** tab, chose the type of authentication or accept the default method which is **Use Strong Password Encryption for Authentication** and press **Next** to continue.



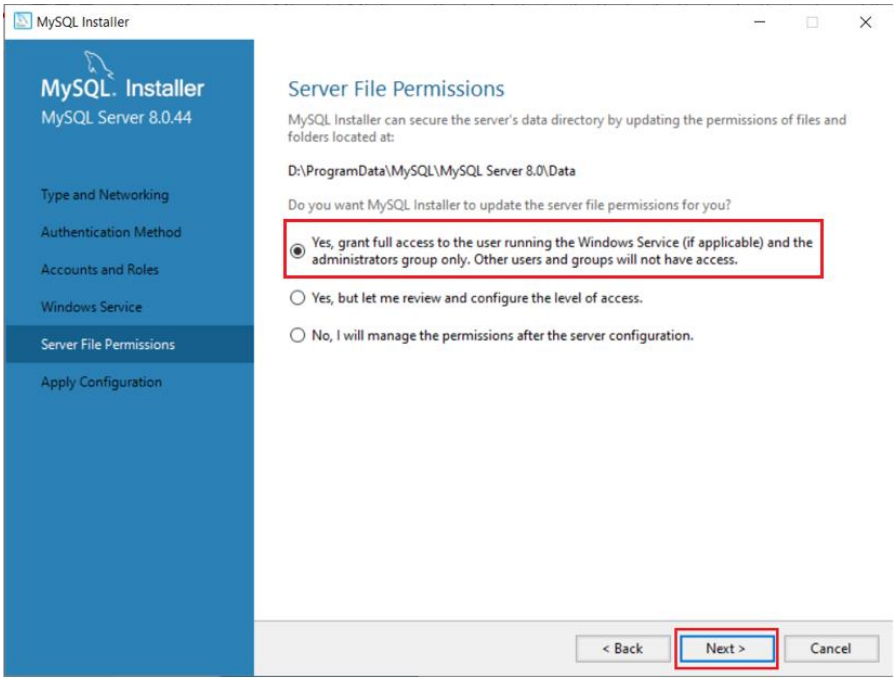
On the **Accounts and Roles** tab, provide **MySQL Root Password**. If needed, provide MySQL user accounts to be created at the time of installation (for now leave this) and press **Next** to continue.



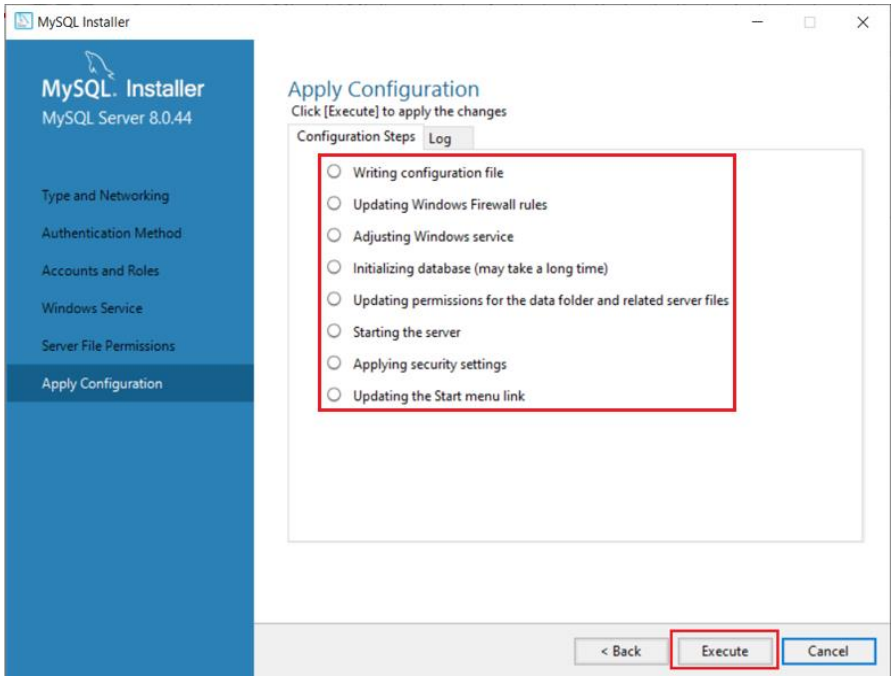
On the **Windows Service** tab, choose the option to **Configure MySQL Server as a Windows Service** and press **Next** to continue.



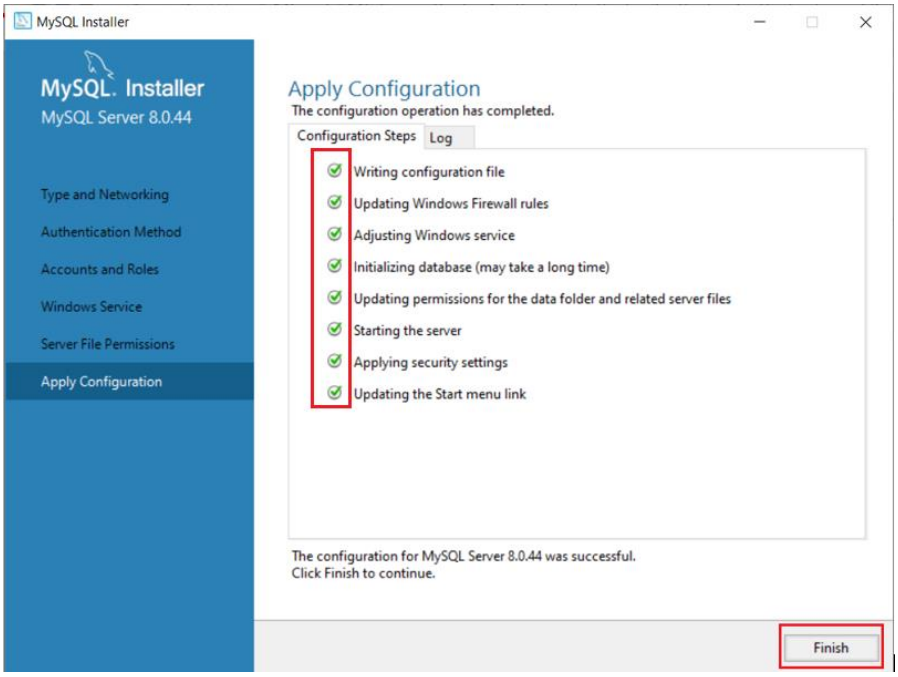
On the **Server File Permissions** tab, select the permissions type or access the default option which is **grant full access to the user running the Windows service** and press **Next** to continue.



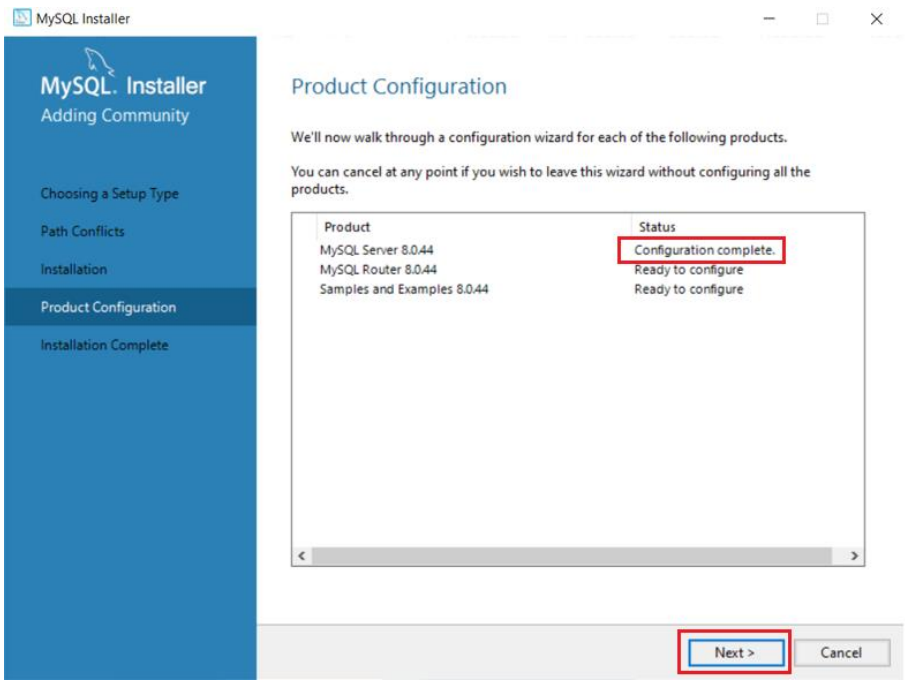
On the **Apply Configuration** tab, review the list of configuration steps that it follows and press **Execute** to continue.



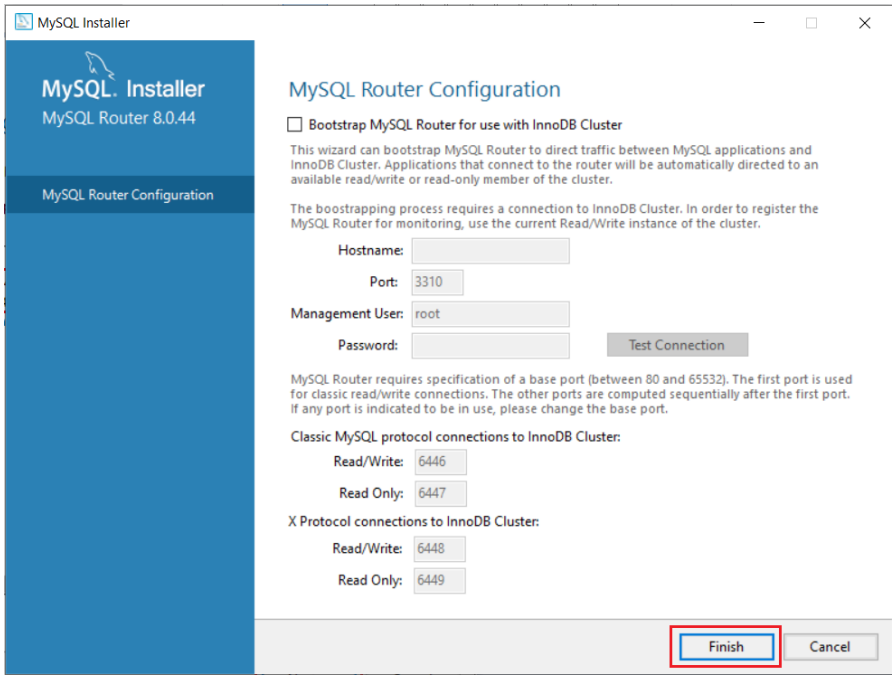
In few minutes, the status of all configuration steps changes to green which indicates the successful configuration. Click **Finish** to continue.



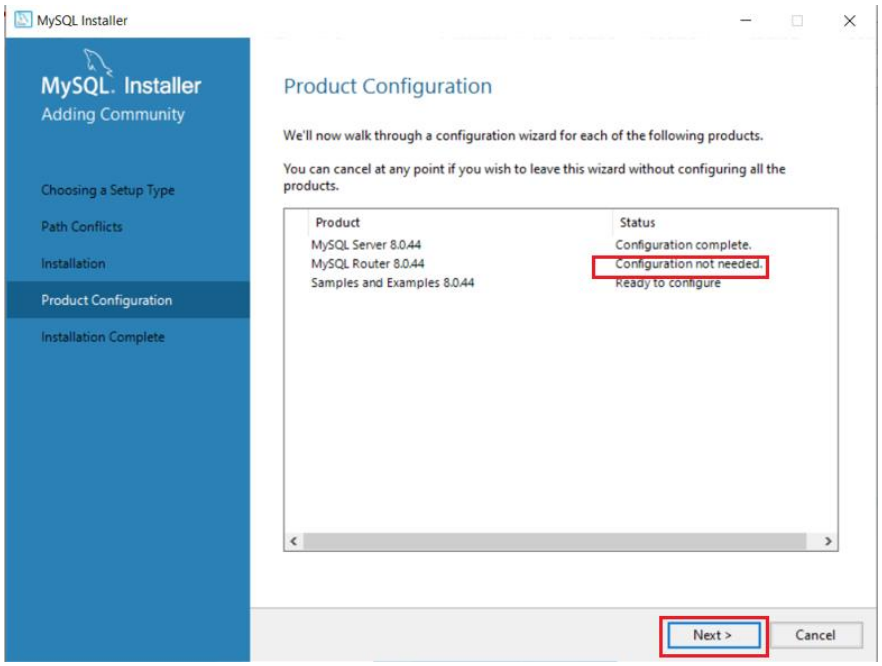
It navigates back to **Production Configuration** tab where you see the status of **MySQL Server** changes to **Configuration complete**. Click **Next** to continue with **MySQL Router** configuration.



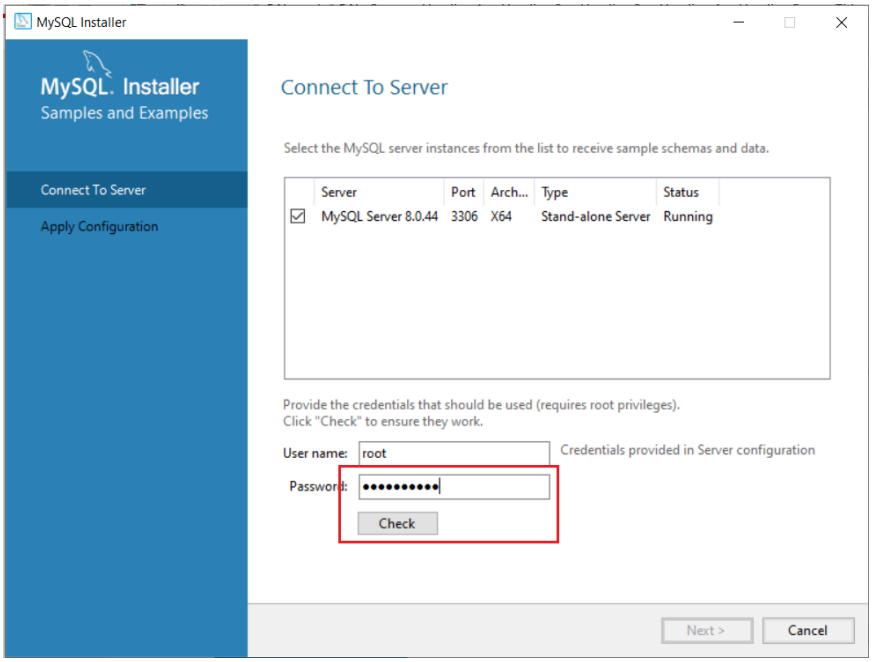
On the **MySQL Router Configuration** tab, it displays the default configuration to connect to InnoDB. Click **Finish** to continue.



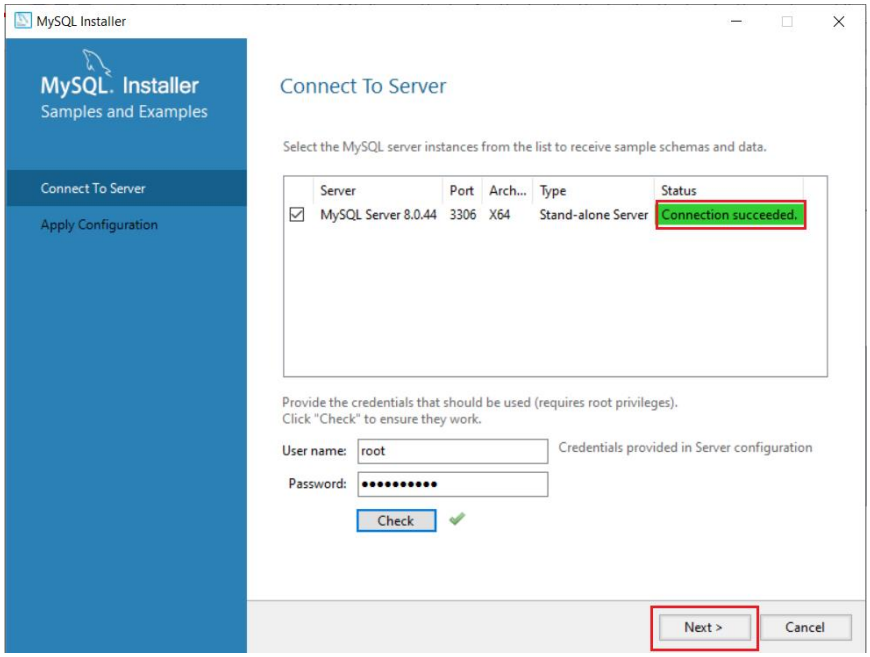
It navigates back to **Production Configuration** tab where you see the status of **MySQL Router** changes to **Configuration not needed**. Click **Next** to continue with **Samples and Examples** configuration.



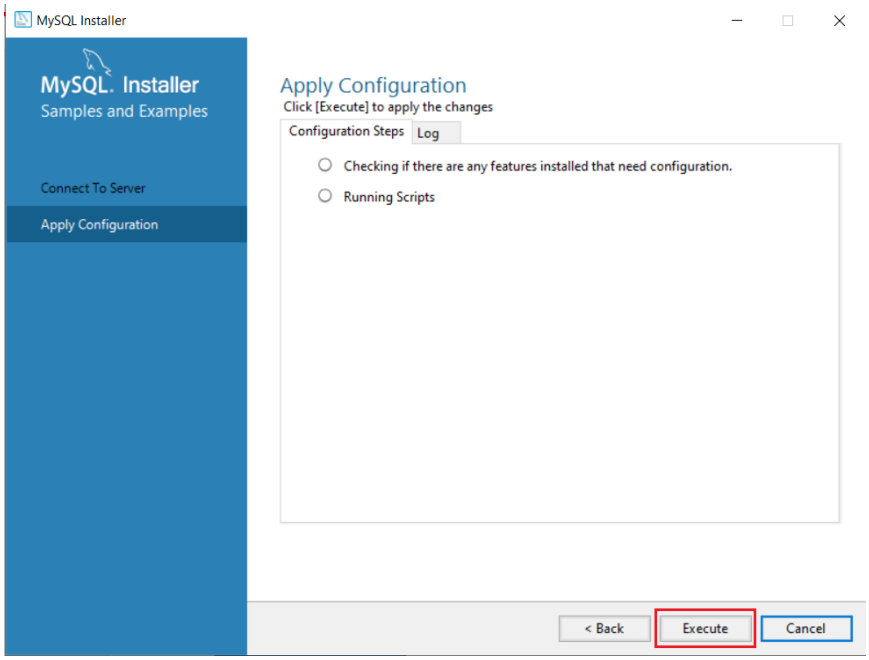
On the **Connect To Server** tab, provide the root password and click on **Check** button to verify the connectivity.



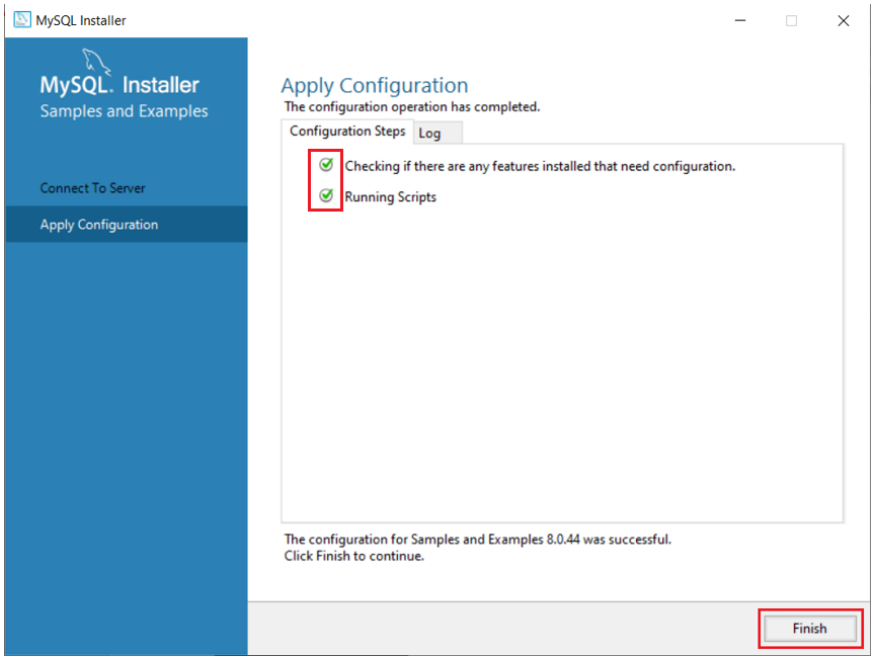
Once the connection is successful, click **Next** to continue.



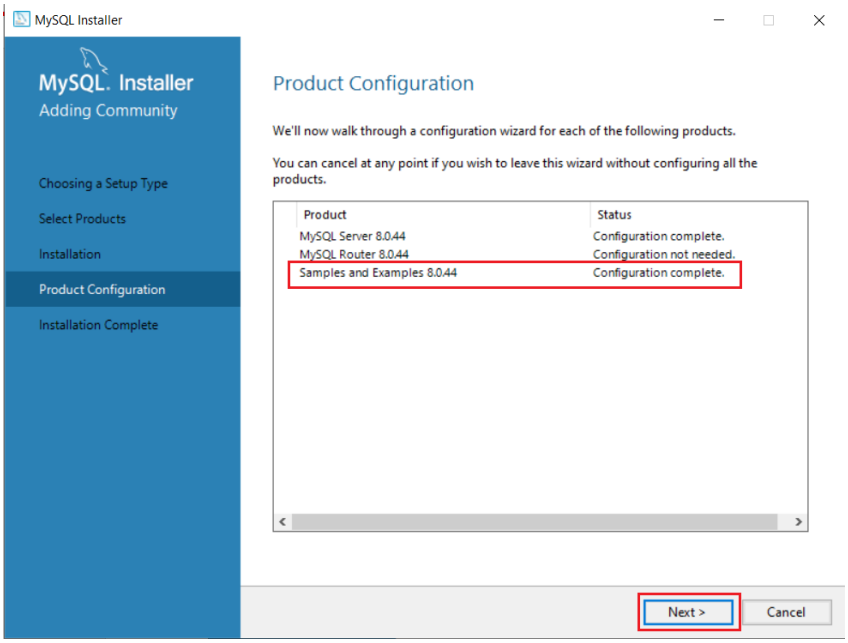
On the **Apply Configuration** tab, review the configuration steps and click on **Execute**.



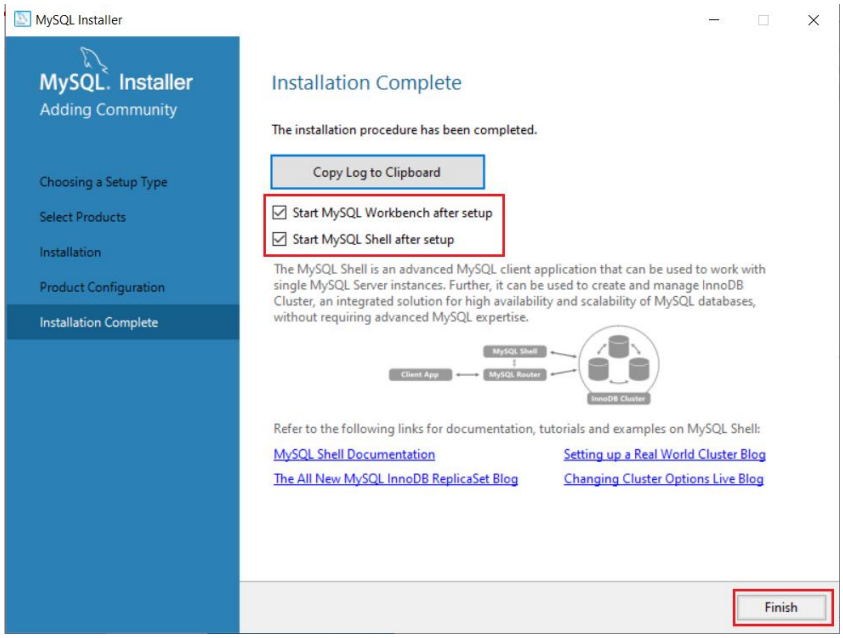
In few minutes, the status of all configuration steps changes to green which indicates the successful configuration. Click **Finish** to continue.



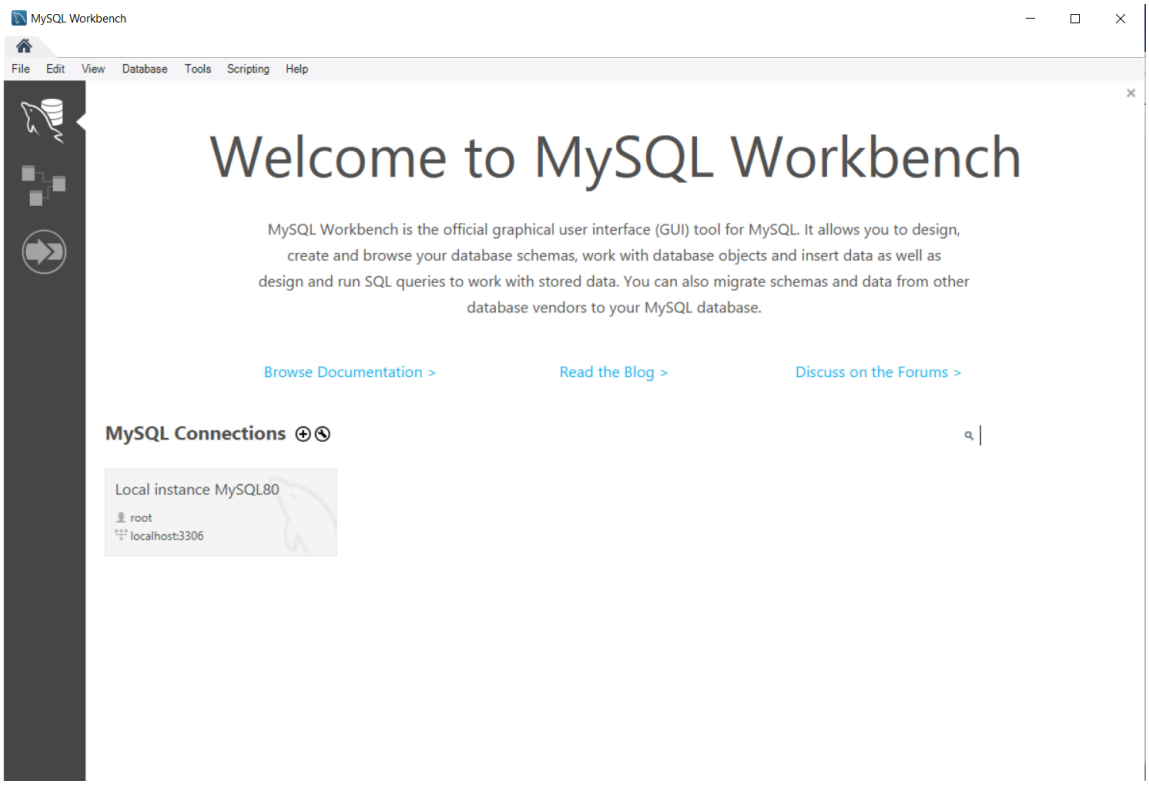
It navigates back to **Production Configuration** tab where you see the status of **Samples and Examples** changes to **Configuration complete**. Click **Next** to continue.

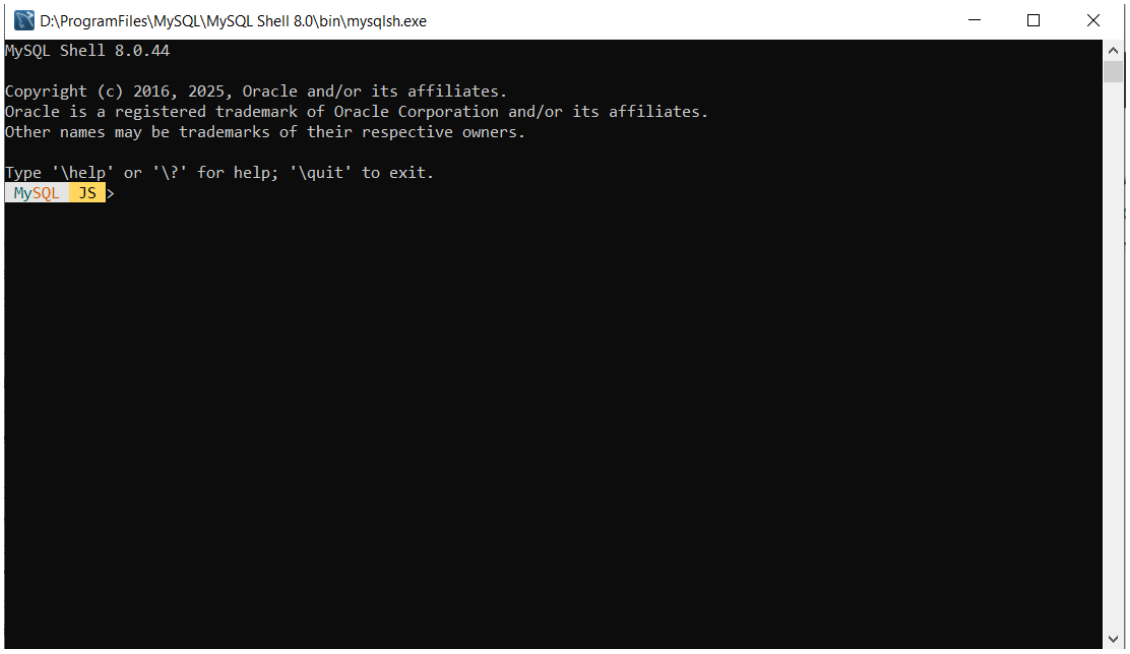


Then, it displays **Installation Complete** message. By default, it starts **MySQL Workbench** and **MySQL Shell** after setup. Click **Finish** to exit the wizard.



Once finished, it opens up **MySQL Workbench** and **MySQL Shell** applications parallely



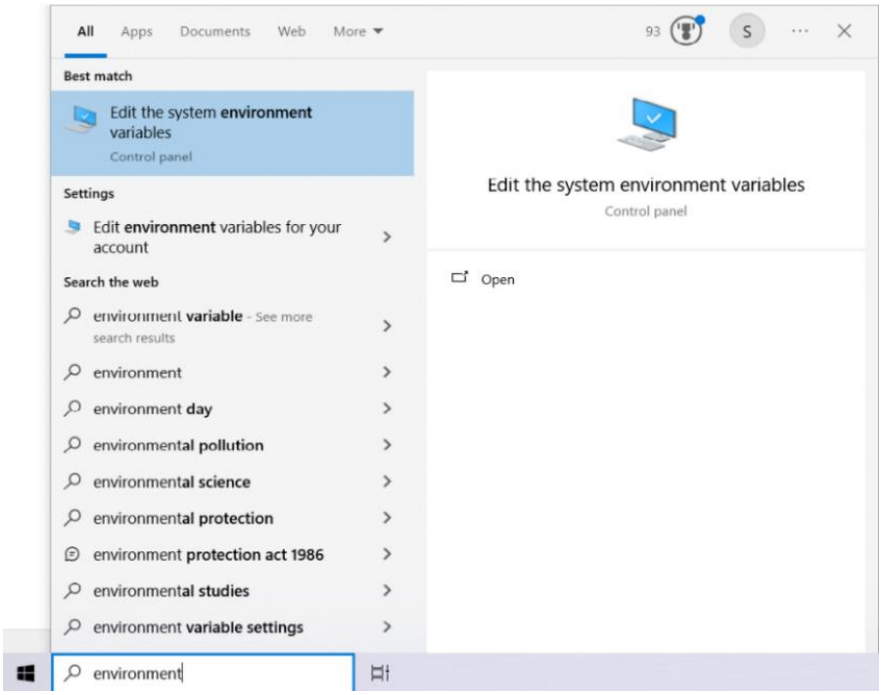


3. Set up Environment Variables:

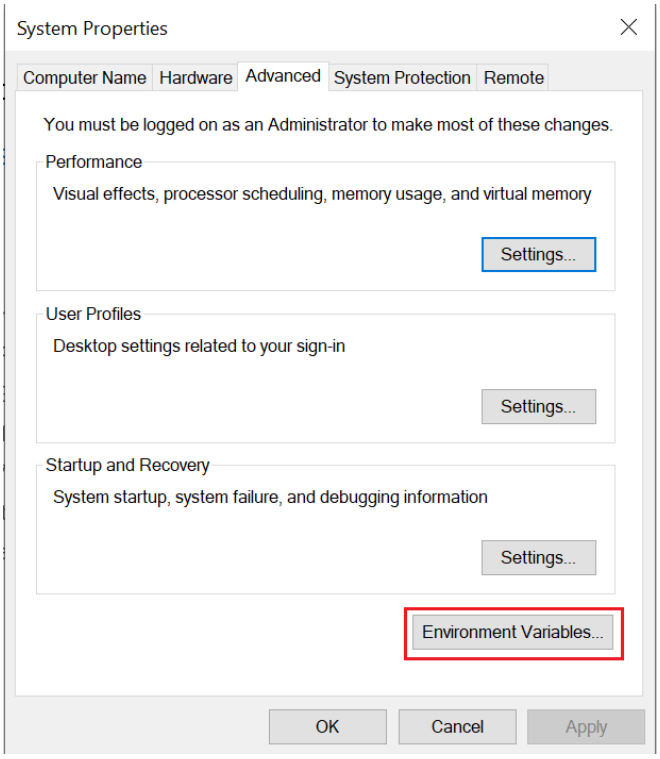
After installing MySQL Server, we should configure the `PATH` environment variable defining MySQL installation path. This variable must be configured under **User environment variables** or **System environment variables** depending on MySQL configuration needed **for a single user** or **for multiple users**.

Here, we will add User environment variables since we are configuring MySQL for a single user. If you would like to configure it for multiple users, then define System environment variables.

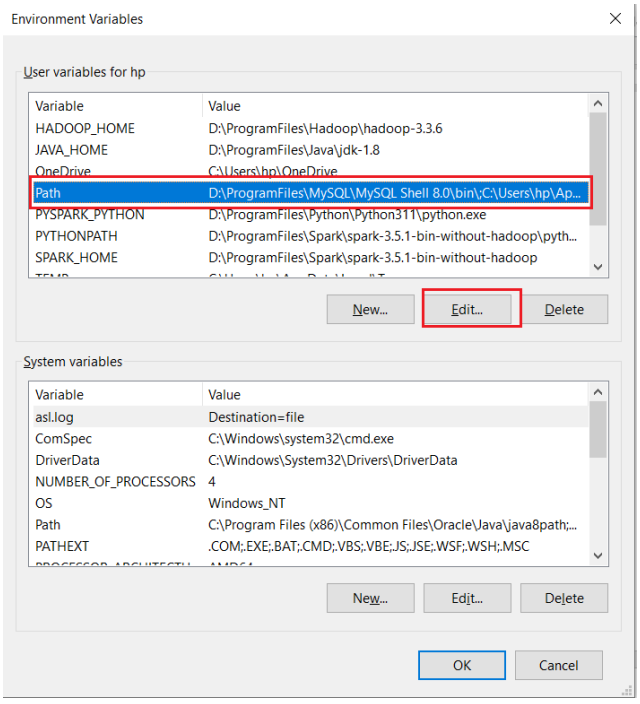
In the Windows search bar, start typing “environment variables” and select the first match which opens up **System Properties** dialog.



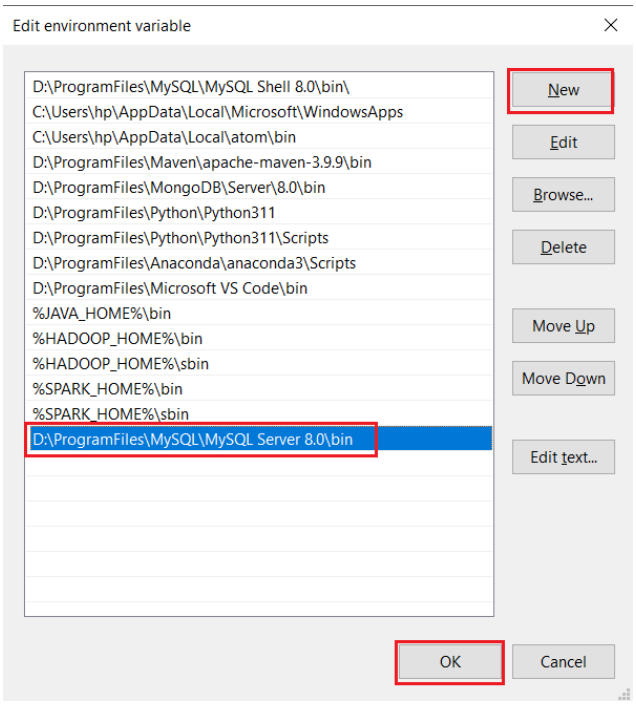
On the **System Properties** window, press **Environment Variables** button.



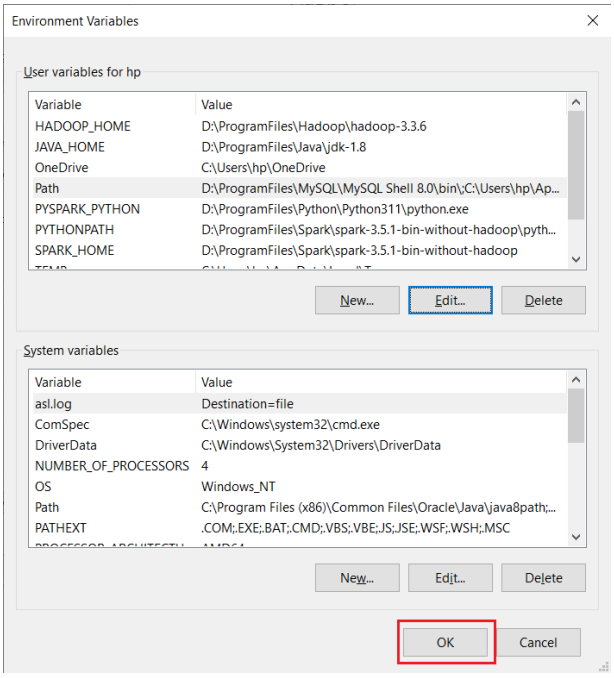
On the **Environment Variables** dialog, select `PATH` variable under **User variables** and press **Edit** button.



Press **New** and add your MySQL Server installation path *(in my case it is* `D:\ProgramFiles\MySQL\MySQL Server 8.0\bin`*)* and press **OK**.



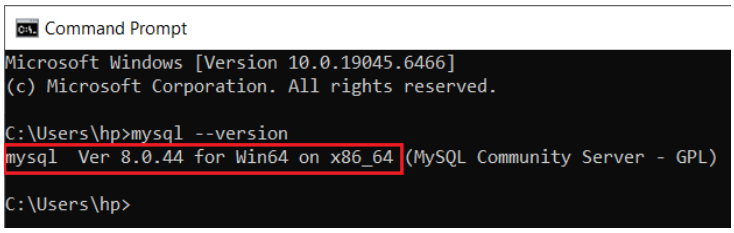
Press **OK** to apply environment variable changes and close window.



4. Verify MySQL:

Open new **Command Prompt** and run the following command to get the installed MySQL version:

```
mysql --version
```

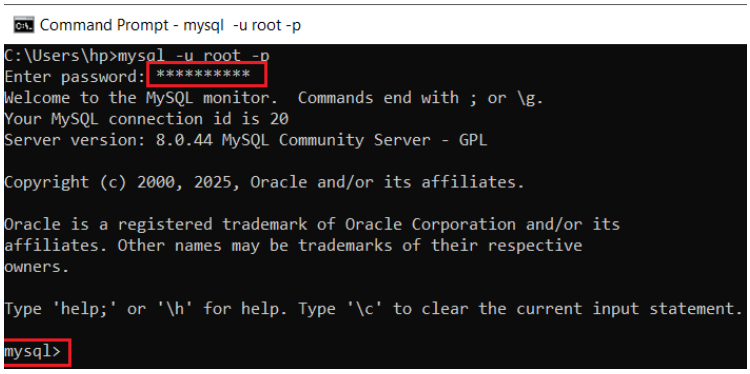


Here, it displays the `mysql` version 8.0.44 version that was installed on the Windows system.

5. Connect MySQL Server:

Run the following command to connect to MySQL server and provide the root password *(given at the time of installation)* when prompted:

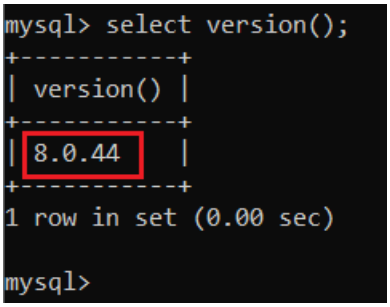
```
mysql -u root -p
```



Once it is successfully connected, it displays `mysql>` prompt where all SQL commands can be executed.

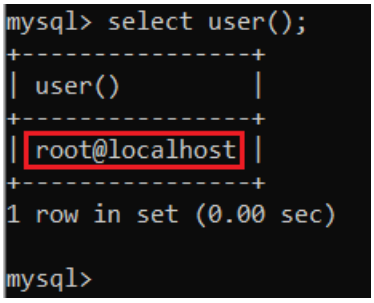
For example, run the following command to get the current MySQL version:

```
select version();
```



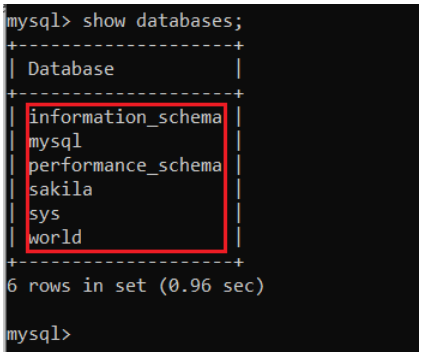
Run the following command to get the current user that has connected to MySQL:

```
select user();
```



Run the following command to display list of databases available in the current MySQL server:

```
show databases;
```



As you can see, it displays the default databases like `information_schema`, `mysql`, `performance_schema`, `sakila`, `sys` and `world`. Here, `sakila` and `world` are sample databases created at the time of installation.

6. Run MySQL Commands:

Follow the official MySQL documentation to understand the syntax for the SQL statements supported by MySQL.

Let us execute some SQL commands to create database, table and insert some data into it.

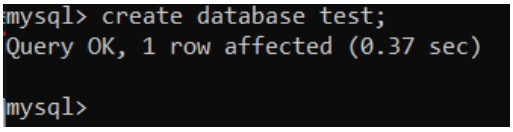
6.1. Create Database:

To create a database, MySQL follows specific syntax as below:

```
CREATE DATABASE database_name;
```

Run the following command to create a database named `test`:

```
create database test;
```



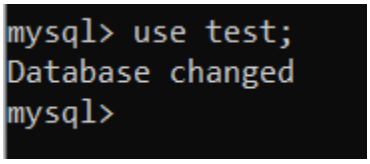
6.2. Select Database:

To select a database, MySQL follows specific syntax as below:

```
USE database_name;
```

Run the following command to use `test` database:

```
use test;
```



Use the following to verify the current database name:

```
select database();
```

```
mysql> select database();
+------------+
| database() |
+------------+
| test       |
+------------+
1 row in set (0.00 sec)

mysql>
```

6.3. Create Table:

To create a table in a database, MySQL follows specific syntax as below:

```
CREATE TABLE table_name (column_name1 data_type
<optional_constraint>, column_name2 data_type <optional_constraint>);
```

Run the following command to create a table named `employee` with columns like `employee_id`, `first_name`, `last_name`, `email`, `phone_number`, etc.:

```
create table employee (employee_id int not null, first_name
varchar(100), last_name varchar(100), email varchar(100),
phone_number varchar(20), hire_date date, salary int, manager_id int,
department_id int);
```

```
mysql> create table employee (employee_id int not null, first_name varchar(100), last_name varchar(100), email varchar(1
00), phone_number varchar(20), hire_date date, salary int, manager_id int, department_id int);
Query OK, 0 rows affected (0.96 sec)

mysql>
```

6.4. Show Columns:

Use the following command to see the list of columns along with their datatypes and constraints in the `employee` table:

```
show columns from employee;
```

```
mysql> show columns from employee;
+---------------+-------------+------+-----+---------+-------+
| Field         | Type        | Null | Key | Default | Extra |
+---------------+-------------+------+-----+---------+-------+
| employee_id   | int         | NO   |     | NULL    |       |
| first_name    | varchar(100)| YES  |     | NULL    |       |
| last_name     | varchar(100)| YES  |     | NULL    |       |
| email         | varchar(100)| YES  |     | NULL    |       |
| phone_number  | varchar(20) | YES  |     | NULL    |       |
| hire_date     | date        | YES  |     | NULL    |       |
| salary        | int         | YES  |     | NULL    |       |
| manager_id    | int         | YES  |     | NULL    |       |
| department_id | int         | YES  |     | NULL    |       |
+---------------+-------------+------+-----+---------+-------+
9 rows in set (0.10 sec)

mysql>
```

6.5. Insert Data:

To insert data into a table, MySQL follows specific syntax as below:

```
INSERT INTO TABLE table_name (column_name1, column_name2) VALUES
(value1, value2);
```

Run the following command to insert 4 records into `employee` table:

```
insert into employee (employee_id, first_name, last_name, email,
phone_number, hire_date, salary, manager_id, department_id)
values(100, 'Steven', 'King', 'SKING@gmail.com','515.123.4567',
'2023-06-17', 24000, 0, 20);
insert into employee (employee_id, first_name, last_name, email,
phone_number, hire_date, salary, manager_id, department_id)
values(101, 'Neena', 'Kochhar', 'NKOCHHAR@gmail.com', '515.123.4568',
'2025-09-21', 17000, 100, 10);
insert into employee (employee_id, first_name, last_name, email,
phone_number, hire_date, salary, manager_id, department_id)
values(102, 'Lex', 'De Haan', 'LDEHAAN@yahoo.com', '515.123.4569',
'2021-01-13', 17000, 100, 10);
insert into employee (employee_id, first_name, last_name, email,
phone_number, hire_date, salary, manager_id, department_id)
values(103, 'Alexander', 'Hunold', 'AHUNOLD@orbit.com',
'590.423.4567', '2022-01-03', 9000, 102, 30);
```

```
mysql> insert into employee (employee_id, first_name, last_name, email, phone_number, hire_date, salary, manager_id, dep
artment_id) values(100, 'Steven', 'King', 'SKING@gmail.com','515.123.4567', '2023-06-17', 24000, 0, 20);
Query OK, 1 row affected (0.11 sec)

mysql> insert into employee (employee_id, first_name, last_name, email, phone_number, hire_date, salary, manager_id, dep
artment_id) values(101, 'Neena', 'Kochhar', 'NKOCHHAR@gmail.com', '515.123.4568', '2025-09-21', 17000, 100, 10);
Query OK, 1 row affected (0.09 sec)

mysql> insert into employee (employee_id, first_name, last_name, email, phone_number, hire_date, salary, manager_id, dep
artment_id) values(102, 'Lex', 'De Haan', 'LDEHAAN@yahoo.com', '515.123.4569', '2021-01-13', 17000, 100, 10);
Query OK, 1 row affected (0.09 sec)

mysql> insert into employee (employee_id, first_name, last_name, email, phone_number, hire_date, salary, manager_id, dep
artment_id) values(103, 'Alexander', 'Hunold', 'AHUNOLD@orbit.com', '590.423.4567', '2022-01-03', 9000, 102, 30);
Query OK, 1 row affected (0.12 sec)

mysql>
```

6.6. Select Data:

To select data from a table, MySQL follows specific syntax as below:

```
SELECT column_names FROM table_name WHERE <condition>;
```

Run the following command to select department 10 employees list:

```
select * from employee where department_id=10;
```

```
mysql> select * from employee where department_id=10;
+-------------+------------+-----------+--------------------+--------------+------------+--------+------------+---------
------+
| employee_id | first_name | last_name | email              | phone_number | hire_date  | salary | manager_id | departme
nt_id |
+-------------+------------+-----------+--------------------+--------------+------------+--------+------------+---------
------+
|         101 | Neena      | Kochhar   | NKOCHHAR@gmail.com | 515.123.4568 | 2025-09-21 |  17000 |        100 |
   10 |
|         102 | Lex        | De Haan   | LDEHAAN@yahoo.com  | 515.123.4569 | 2021-01-13 |  17000 |        100 |
   10 |
+-------------+------------+-----------+--------------------+--------------+------------+--------+------------+---------
------+
2 rows in set (0.00 sec)

mysql>
```

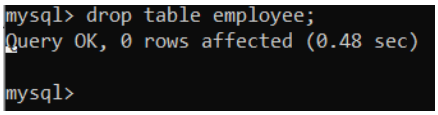
6.7. Drop Table:

To drop a specific table, MySQL follows specific syntax as below:

```
DROP TABLE table_name;
```

Run the following command to drop `employee` table:

```
drop table employee;
```



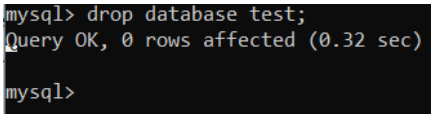
6.8. Drop Database:

To drop a specific database, MySQL follows specific syntax as below:

```
DROP DATABASE database_name;
```

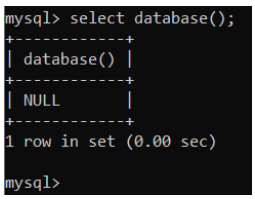
Run the following command to drop `test` database:

```
drop database test;
```



Run the following commands to verify if the current test database has been removed:

```
select database();
```



7. MySQL Server CLI Utilities:

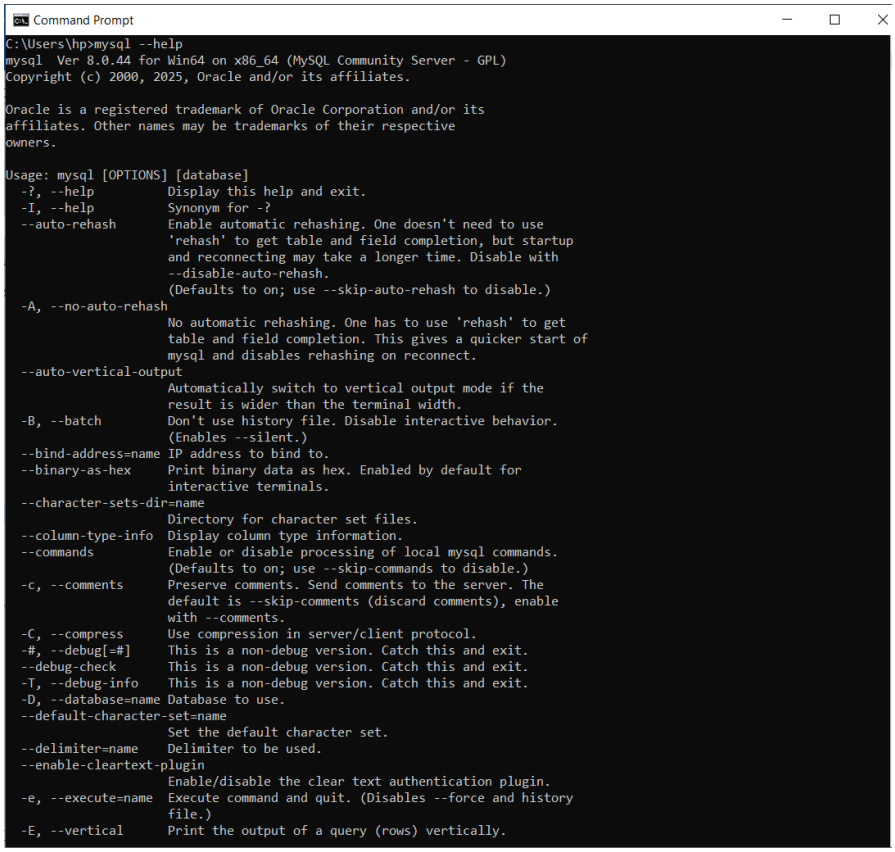
MySQL provides a bunch of command line utilities to perform various activities.

7.1. mysql:

`mysql` is a simple SQL shell with input line editing capabilities and it works in both interactive and noninteractive mode. `mysql` client is used to execute SQL commands.

`mysql` supports various options which can be specified in the command line. Use the following command to get the list of options supported by `mysql`.

```
mysql --help
```



For example, use the following commands to get the current `mysql` client version:

```
mysql -V
```

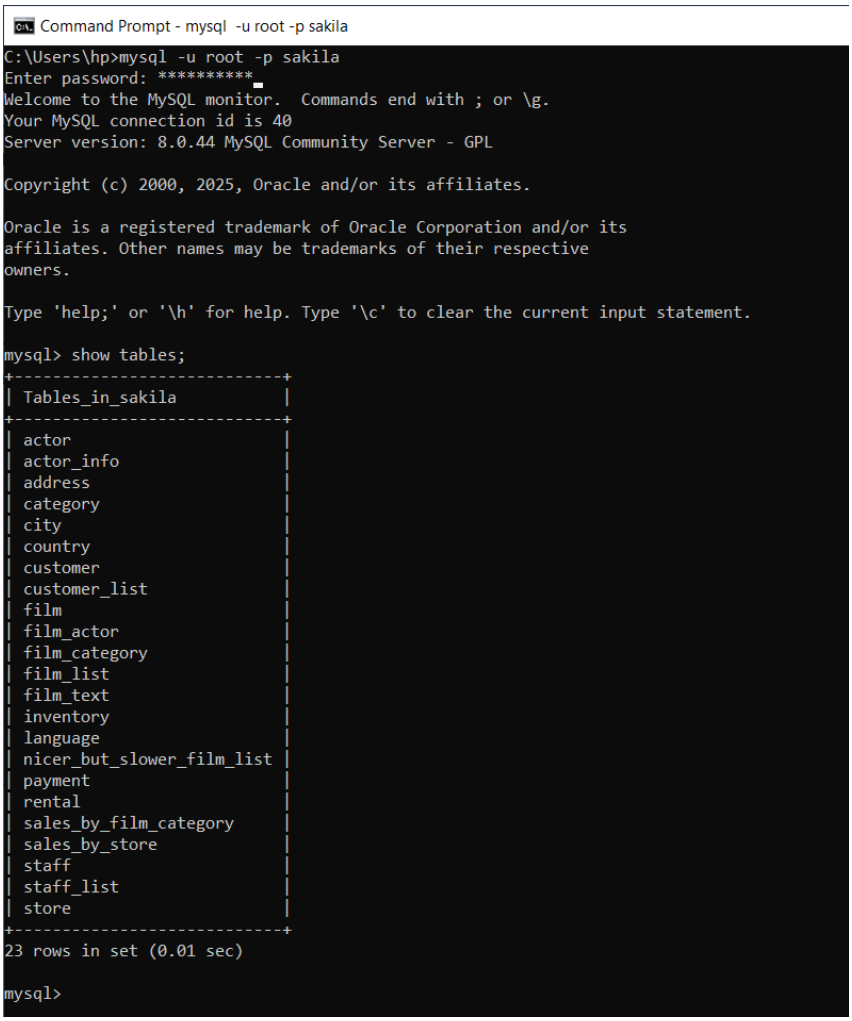
or

```
mysql --version
```

```
C:\Users\hp>mysql -V
mysql  Ver 8.0.44 for Win64 on x86_64 (MySQL Community Server - GPL)

C:\Users\hp>
```

Use the following command to connect to specific database like `sakila` directly and enter `root` user password when prompted:

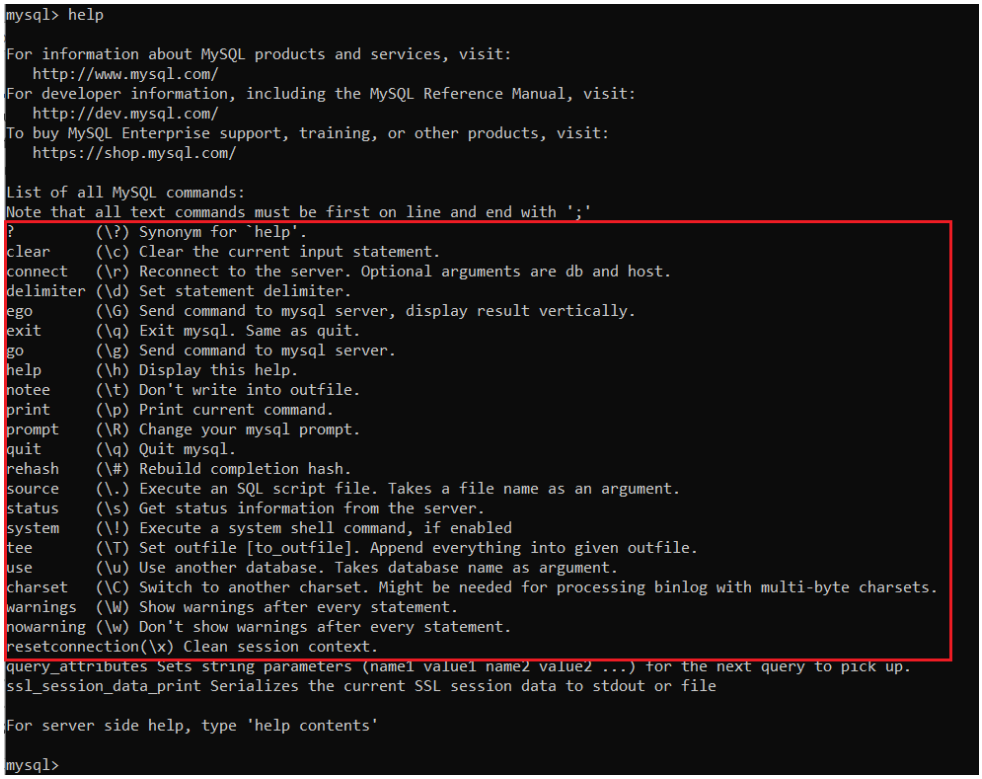
```
mysql -u root -p sakila
```



`mysql` provides many built-in commands to perform tasks like connecting to a MySQL instance, sending command to MySQL server, executing a script file, etc.

Run the following command in `mysql>` prompt to get the list of those commands:

```
help
```



Following are some common commands being used:

- `?` or `\?` or `\h` or `help` or `\help`: Displays general help about `mysql` CLI.
- `\q` or `quit` or `exit`: Exits `mysql` CLI.
- `\c` or `clear`: Clears current input statement.
- `\r` or `connect`: Reconnects to the same MySQL instance.
- `\p` or `print`: Prints current input statement without executing it.
- `\R` or `prompt`: Changes the `mysql` prompt.
- `\g` or `go`: Sends command to the MySQL instance.
- `\s` or `status`: Displays the status information of MySQL server.
- `\.` or `source`: Executes script file using the active language.
- `!` or `system`: Executes the specified shell command.
- `\u` or `use`: Allows to specify schema to use.

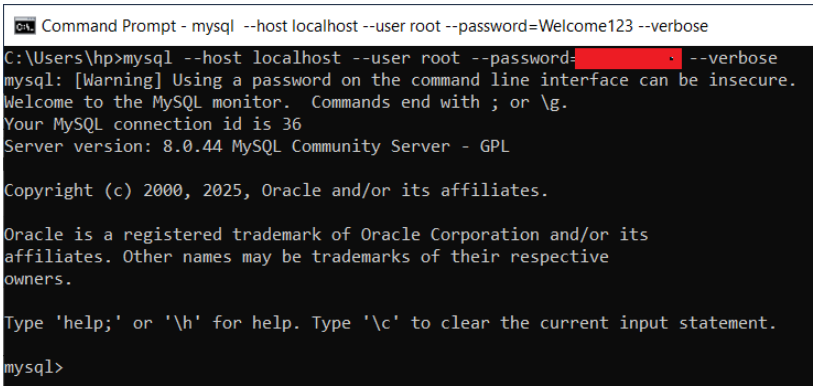
Use the following commands to connect to local MySQL instance with `root` user and password *(replace `<root_password>` with your root password given at the time of installation)* in verbose *(which displays SQL command in output)* mode.

```
mysql -h localhost -u root -p=<root_password> -v
```

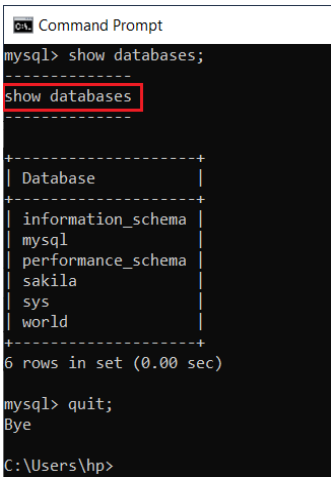
or

```
mysql --host localhost --user root --password=<root_password> --
verbose
```

Note that `-v` option can be given multiple times to produce more and more output such as `-v -v -v` produces table output format even in batch mode.

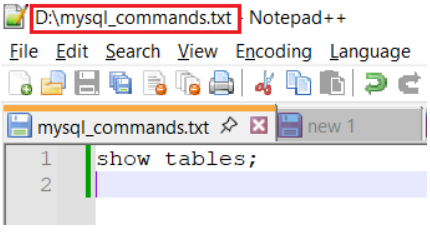


Once connected in verbose mode, `mysql` displays what the command does.
For example, run `show databases` command and it displays the command in the output:



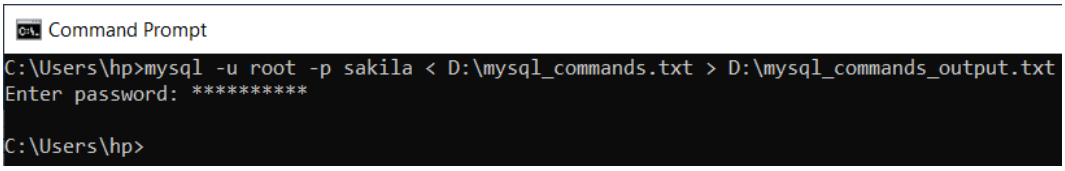
`mysql` tool also allows to execute SQL commands from the text file.
First, create a text file named `mysql_commands.txt` in D drive and enter the following commands in the file:

```
show tables;
```

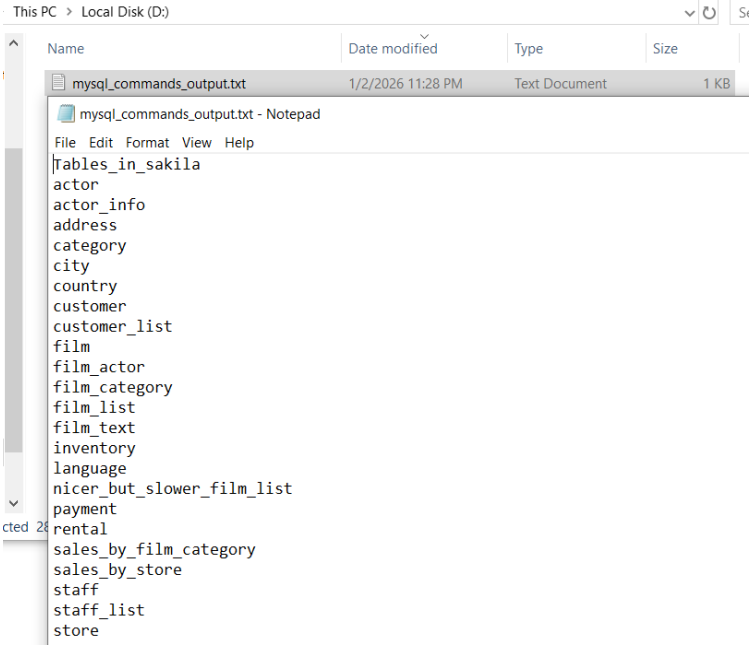


Now, run the following command to connect to `sakila` database and execute `mysql_commands.txt` file and save the output into output file named `mysql_commands_output.txt`:

```
mysql -u root -p sakila < D:\mysql_commands.txt >
D:\mysql_commands_output.txt
```

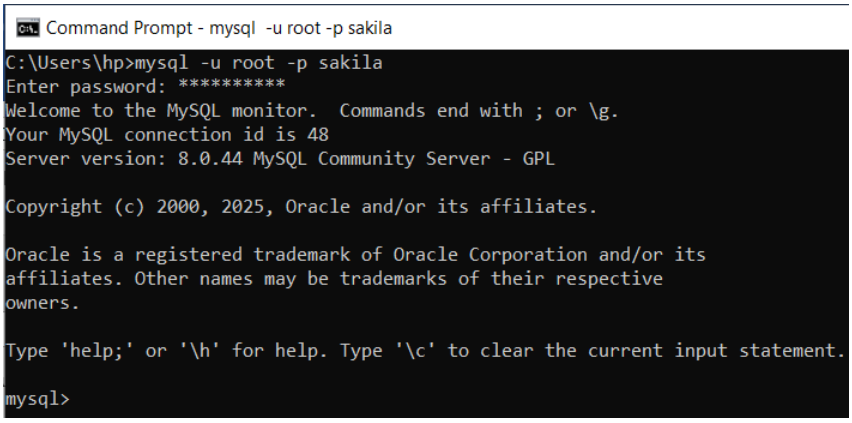


Open `D:\mysql_commands_output.txt` file where you can see the list of tables available in `sakila`.



SQL file can also be executed after connecting to MySQL server.
First, run the following command to connect to `sakila` database. Enter `root` user password when prompted:

```
mysql -u root -p sakila
```

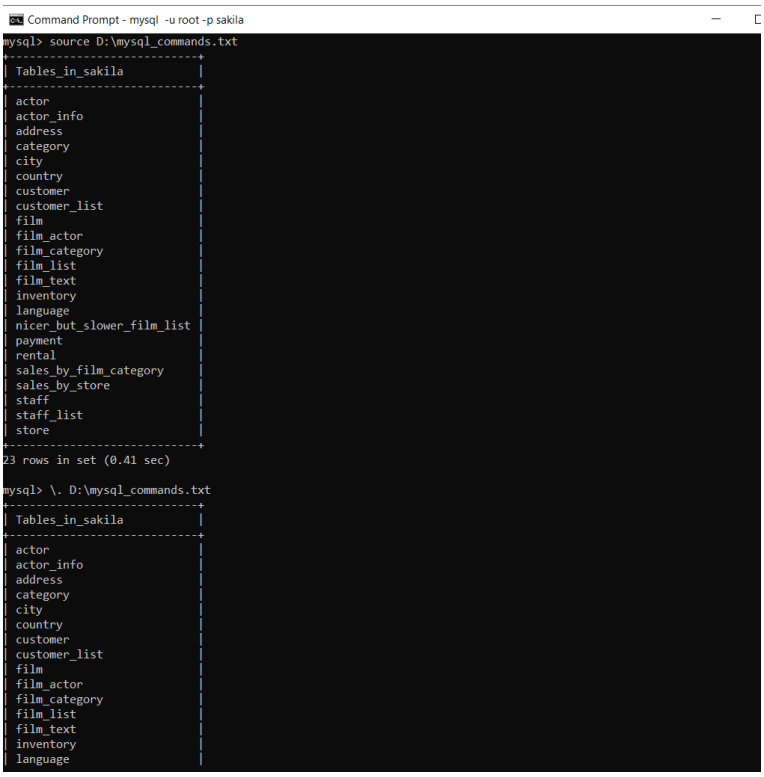


Then run the following SQL command to execute `D:\mysql_commands.txt` file:

```
source D:\mysql_commands.txt
```

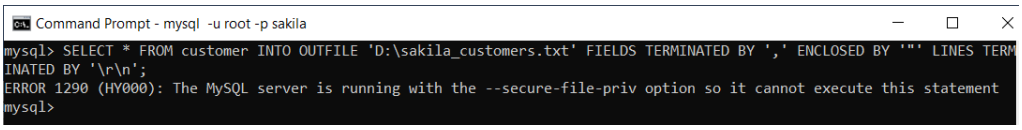
or

```
\. D:\mysql_commands.txt
```



To export a specific table data into a file, use the following query:

```
SELECT * FROM customer INTO OUTFILE 'D:\sakila_customers.txt' FIELDS
TERMINATED BY ',' ENCLOSED BY '"' LINES TERMINATED BY '\r\n';
```



To import data from a file to a specific table, use the following command:

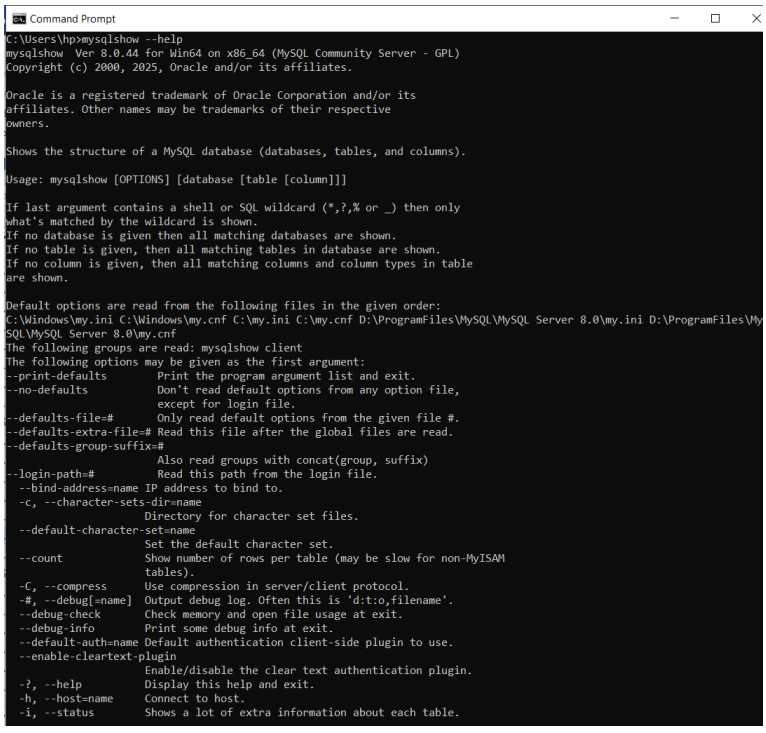
```
LOAD DATA LOCAL INFILE 'D:\sakila_customers.txt' INTO TABLE
customers_dump FIELDS TERMINATED BY ',' LINES TERMINATED BY '\r\n';
```

7.2. mysqlshow:

`mysqlshow` client can be used to quickly check the existence of databases, tables, columns and indexes.

`mysqlshow` supports various options which can be specified in the command line. Use the following command to get the list of options supported by `mysqlshow` and commands to be executed:

```
mysqlshow --help
```

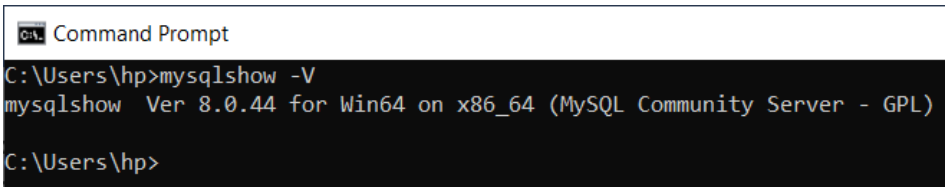


For example, use the following commands to get the current `mysqlshow` client version:

```
mysqlshow -V
```

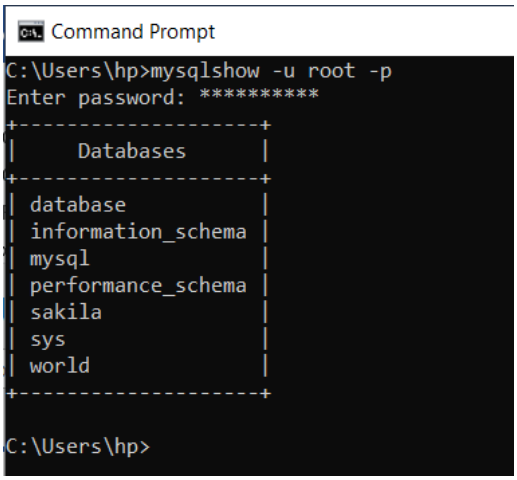
or

```
mysqlshow --version
```



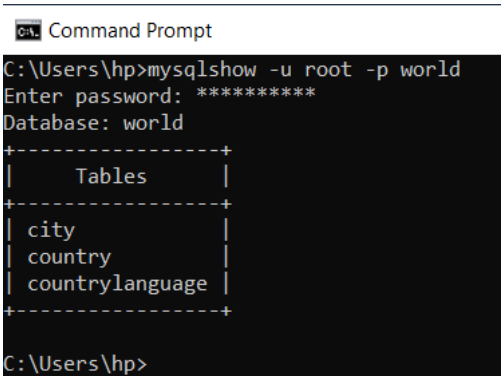
Use the following command to get the list of databases. Enter `root` user password when prompted:

```
mysqlshow -u root -p
```



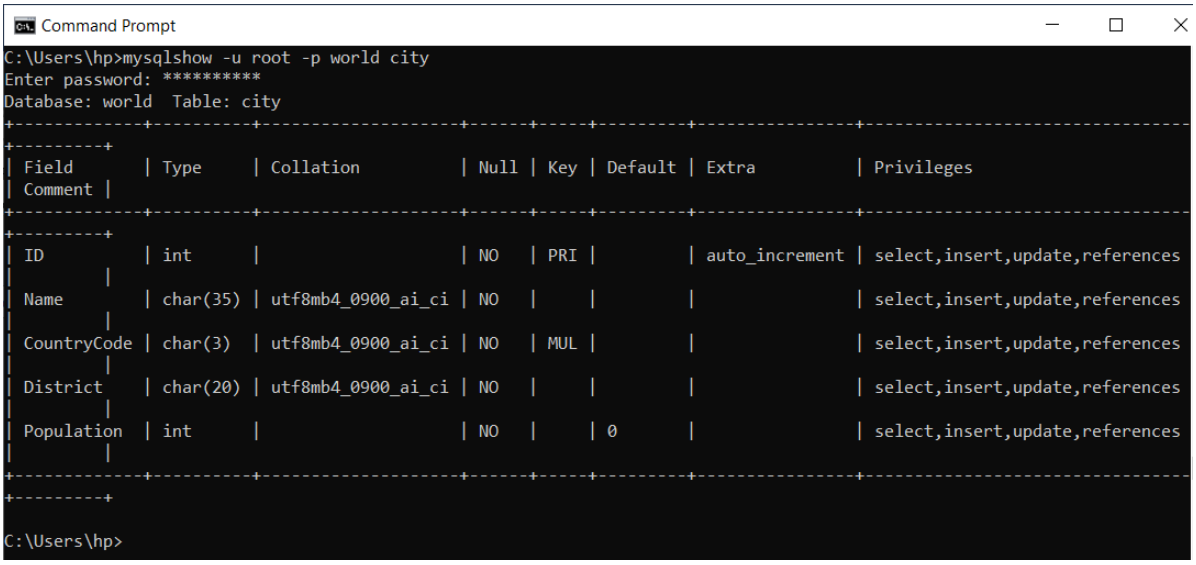
Use the following command to get the list of tables in `test` database. Enter `root` user password when prompted:

```
mysqlshow -u root -p world
```



Use the following command to get the list of columns in `city` table in `world` database. Enter `root` user password when prompted:

```
mysqlshow -u root -p world city
```

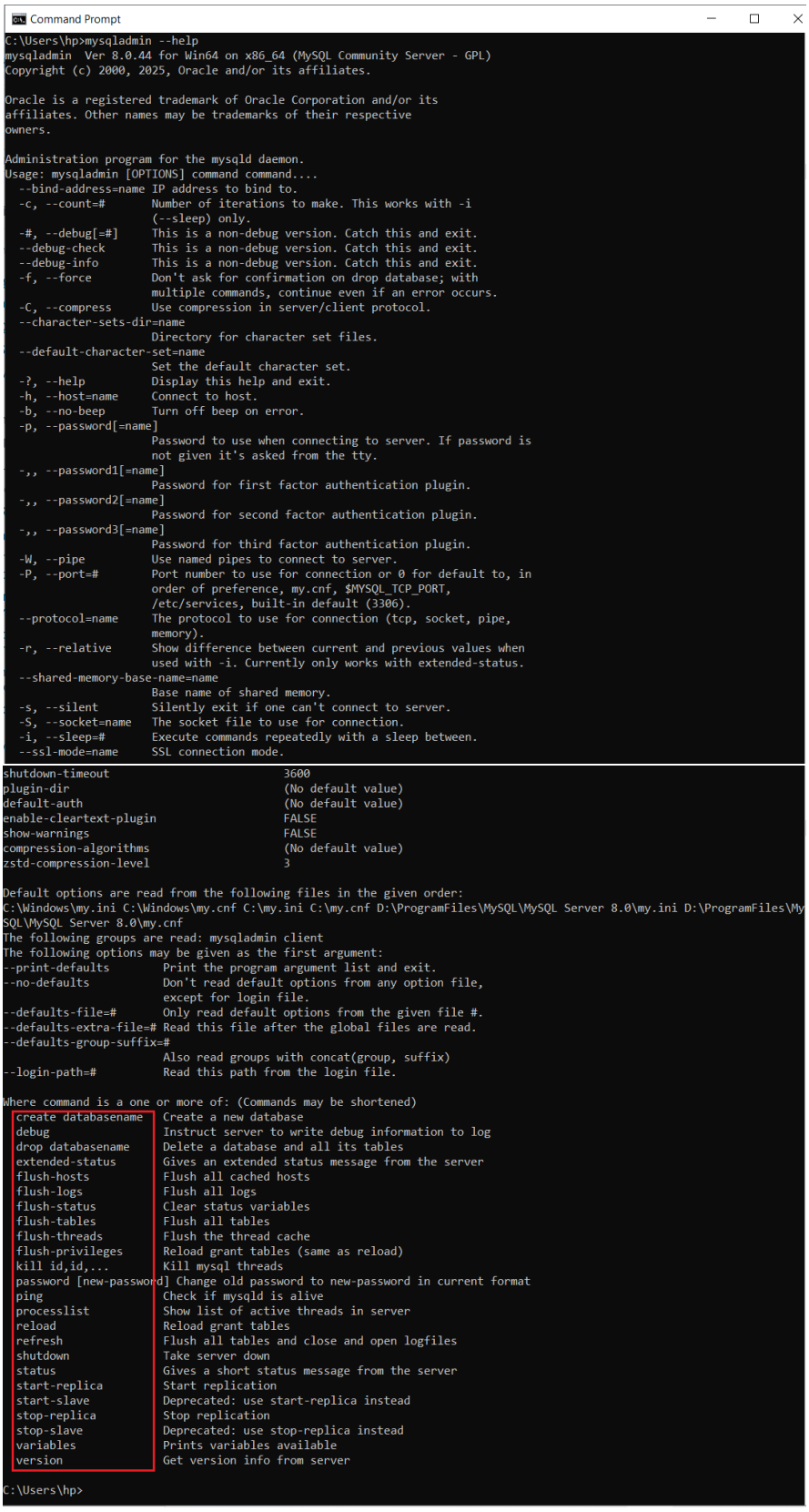


7.3. mysqladmin:

`mysqladmin` client can be used to perform various administrative activities such as check current server status, verify configuration, create and drop databases, stop server, etc.

`mysqladmin` supports various options which can be specified in the command line. Use the following command to get the list of options supported by `mysqladmin` and commands to be executed:

```
mysqladmin --help
```

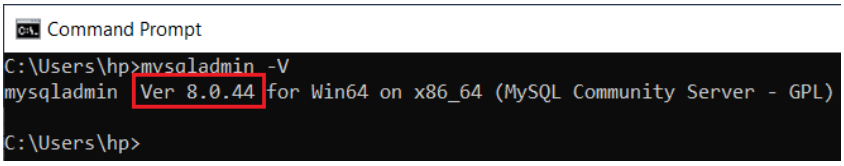

Command Prompt
C:\Users\hp>mysqladmin --help
mysqladmin Ver 8.0.44 for Win64 on x86_64 (MySQL Community Server - GPL)
Copyright (c) 2000, 2025, Oracle and/or its affiliates.
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.
Administration program for the mysqld daemon.
Usage: mysqladmin [OPTIONS] command command....
--bind-address=name IP address to bind to.
-c, --count=# Number of iterations to make. This works with -i
(--sleep) only.
-#, --debug[=#] This is a non-debug version. Catch this and exit.
--debug-check This is a non-debug version. Catch this and exit.
--debug-info This is a non-debug version. Catch this and exit.
-f, --force Don't ask for confirmation on drop database; with
multiple commands, continue even if an error occurs.
-C, --compress Use compression in server/client protocol.
--character-sets-dir=name
Directory for character set files.
--default-character-set=name
Set the default character set.
-?, --help Display this help and exit.
-h, --host=name Connect to host.
-b, --no-beep Turn off beep on error.
-p, --password[=name]
Password to use when connecting to server. If password is
not given it's asked from the tty.
-,, --password1[=name]
Password for first factor authentication plugin.
-,, --password2[=name]
Password for second factor authentication plugin.
-,, --password3[=name]
Password for third factor authentication plugin.
-W, --pipe Use named pipes to connect to server.
-P, --port=# Port number to use for connection or 0 for default to, in
order of preference, my.cnf, $MYSQL_TCP_PORT,
/etc/services, built-in default (3306).
--protocol=name The protocol to use for connection (tcp, socket, pipe,
memory).
-r, --relative Show difference between current and previous values when
used with -i. Currently only works with extended-status.
--shared-memory-base-name=name
Base name of shared memory.
-s, --silent Silently exit if one can't connect to server.
-S, --socket=name The socket file to use for connection.
-i, --sleep=# Execute commands repeatedly with a sleep between.
--ssl-mode=name SSL connection mode.
shutdown-timeout 3600
plugin-dir (No default value)
default-auth (No default value)
enable-cleartext-plugin FALSE
show-warnings FALSE
compression-algorithms (No default value)
zstd-compression-level 3
Default options are read from the following files in the given order:
C:\Windows\my.ini C:\Windows\my.cnf C:\my.ini C:\my.cnf D:\ProgramFiles\MySQL\MySQL Server 8.0\my.ini D:\ProgramFiles\My
SQL\MySQL Server 8.0\my.cnf
The following groups are read: mysqladmin client
The following options may be given as the first argument:
--print-defaults Print the program argument list and exit.
--no-defaults Don't read default options from any option file,
except for login file.
--defaults-file=# Only read default options from the given file #.
--defaults-extra-file=# Read this file after the global files are read.
--defaults-group-suffix=#
Also read groups with concat(group, suffix)
--login-path=# Read this path from the login file.
Where command is a one or more of: (Commands may be shortened)
create databasename Create a new database
debug Instruct server to write debug information to log
drop databasename Delete a database and all its tables
extended-status Gives an extended status message from the server
flush-hosts Flush all cached hosts
flush-logs Flush all logs
flush-status Clear status variables
flush-tables Flush all tables
flush-threads Flush the thread cache
flush-privileges Reload grant tables (same as reload)
kill id,id,... Kill mysql threads
password [new-password] Change old password to new-password in current format
ping Check if mysqld is alive
processlist Show list of active threads in server
reload Reload grant tables
refresh Flush all tables and close and open logfiles
shutdown Take server down
status Gives a short status message from the server
start-replica Start replication
start-slave Deprecated: use start-replica instead
stop-replica Stop replication
stop-slave Deprecated: use stop-replica instead
variables Prints variables available
version Get version info from server
C:\Users\hp>

For example, use the following commands to get the current `mysqladmin` client version:

```
mysqladmin -V
```

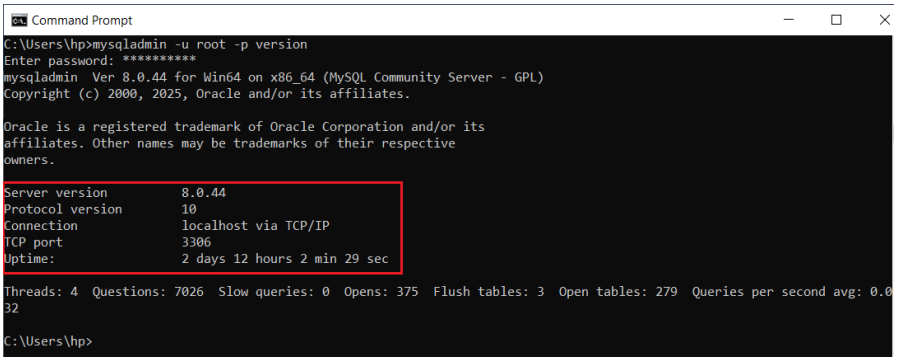
or

```
mysqladmin --version
```



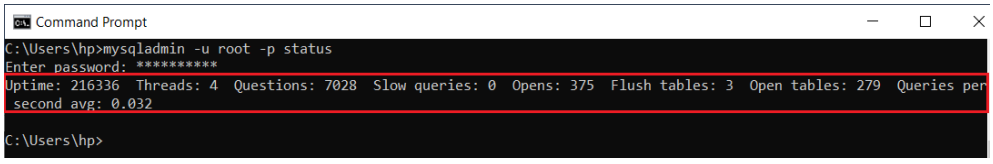
Use the following command to connect to MySQL server and get the server version and uptime. Enter `root` user password when prompted:

```
mysqladmin -u root -p version
```



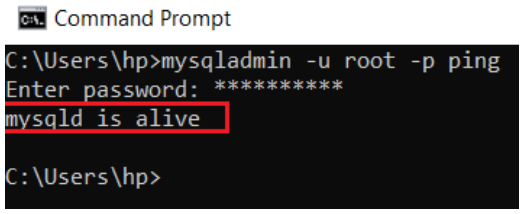
Use the following command to get the short status of MySQL server. Enter `root` user password when prompted:

```
mysqladmin -u root -p status
```



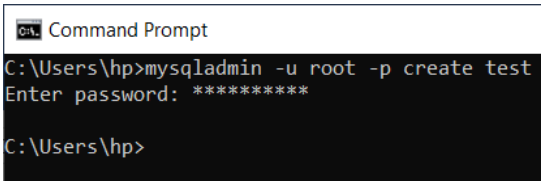
Use the following command to verify if `mysqld` (MySQL server) is alive. Enter `root` user password when prompted:

```
mysqladmin -u root -p status
```



Use the following command to create database named `test`. Enter `root` user password when prompted:

```
mysqladmin -u root -p create test
```

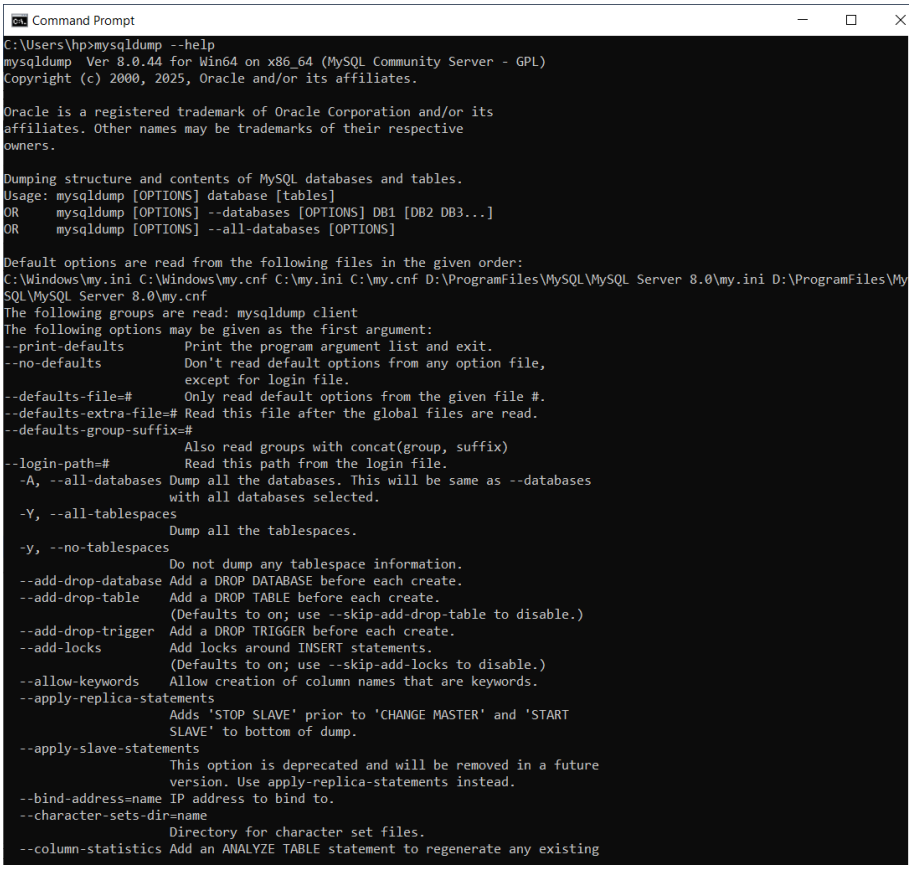


7.4. mysqldump:

`mysqldump` client can help to perform logical backups by producing a set of SQL commands that can be executed to reproduce original database and its data. It is also used to generate table output in CSV, XML or any delimited text file.

`mysqldump` supports various options which can be specified in the command line. Use the following command to get the list of options supported by `mysqldump` and commands to be executed:

```
mysqldump --help
```

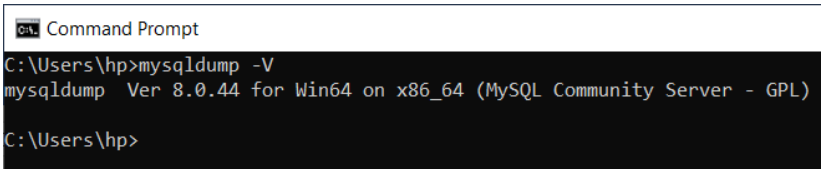


For example, use the following commands to get the current `mysqldump` client version:

```
mysqldump -V
```

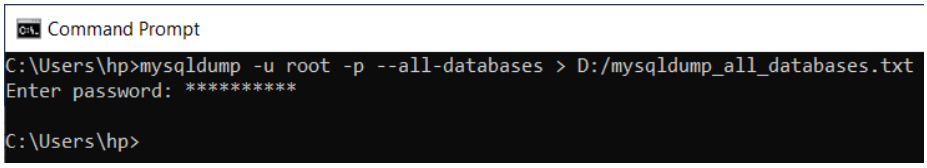
or

```
mysqldump --version
```

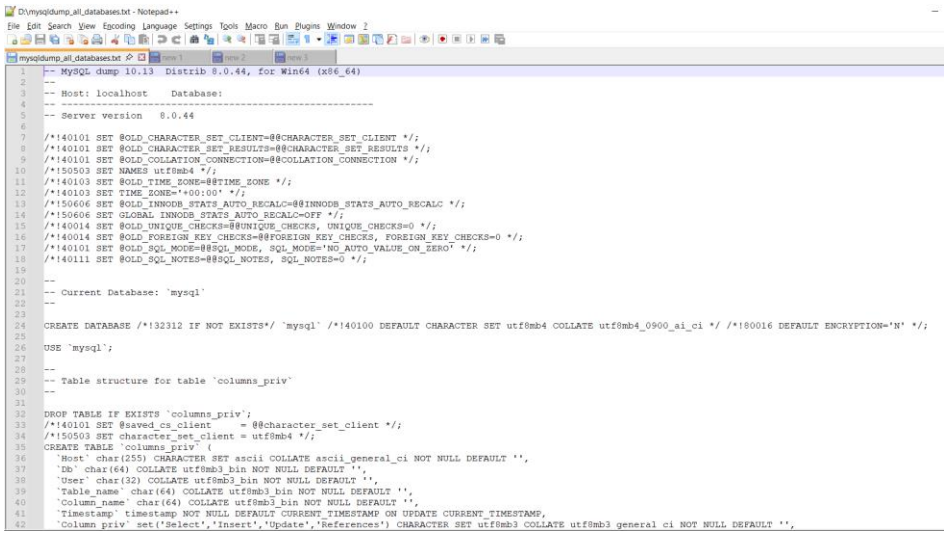


Use the following command to export all databases to `D:\mysqldump_all_databases.txt` file in SQL format. Enter `root` user password when prompted:

```
mysqldump -u root -p --all-databases > D:/mysqldump_all_databases.txt
```

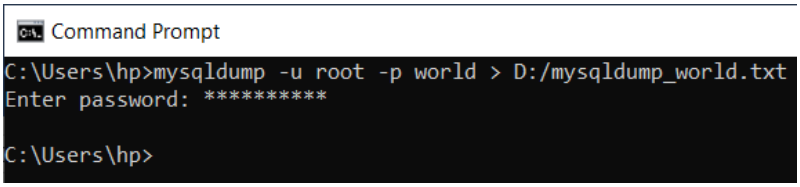


Open `D:\mysqldump_all_databases.txt` file where you can see the create and insert scripts for all databases.

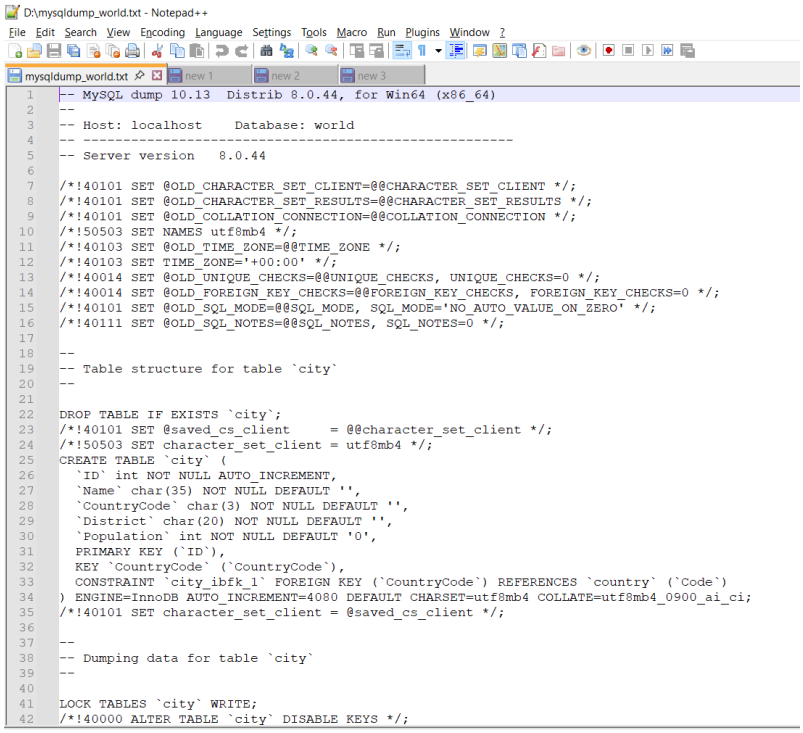


Use the following command to export the all tables in `world` database to `D:\mysqldump_world.txt` file in SQL format. Enter `root` user password when prompted:

```
mysqldump -u root -p world > D:/mysqldump_world.txt
```

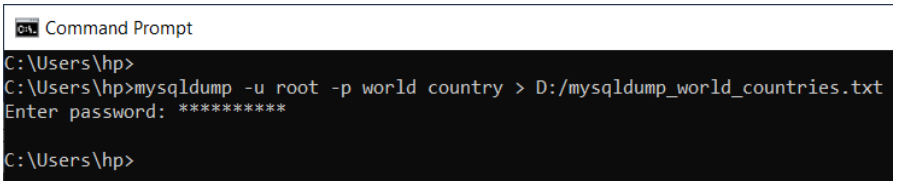


Open `D:\mysqldump_world.txt` file where you can see the create and insert scripts for all tables in `world` database.

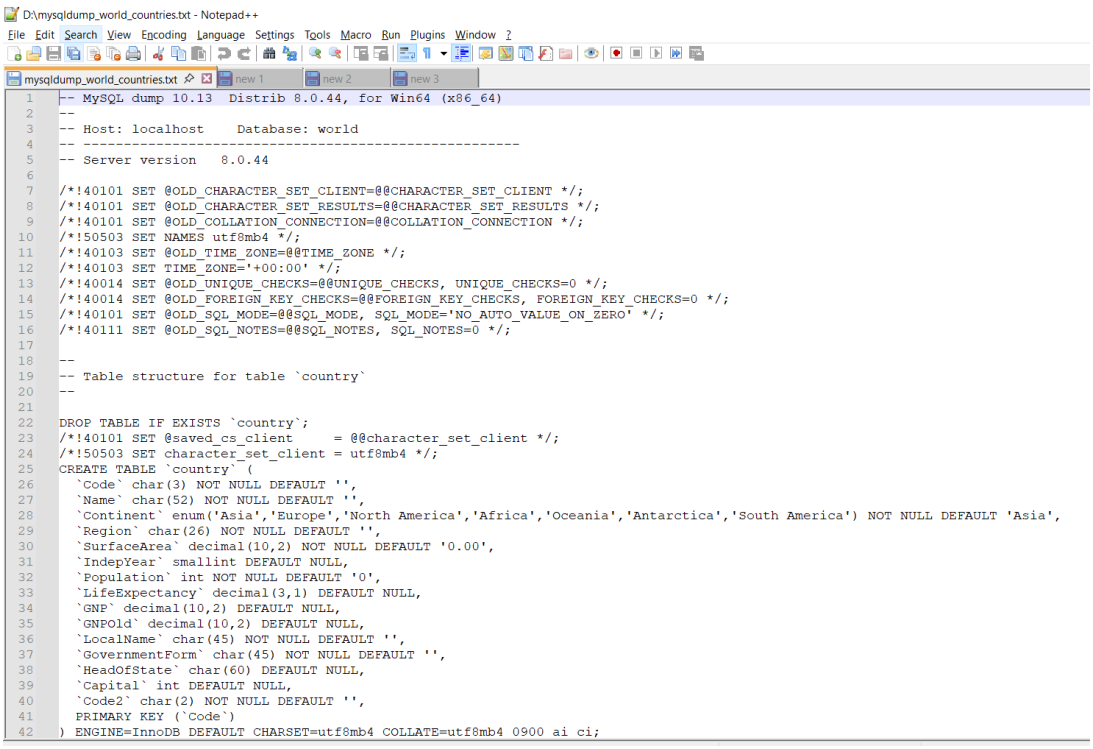


Use the following command to export only `country` table in `world` database to `D:/mysqldump_world_countries.txt` file in SQL format. Enter `root` user password when prompted:

```
mysqldump -u root -p world country > D:/mysqldump_world_countries.txt
```

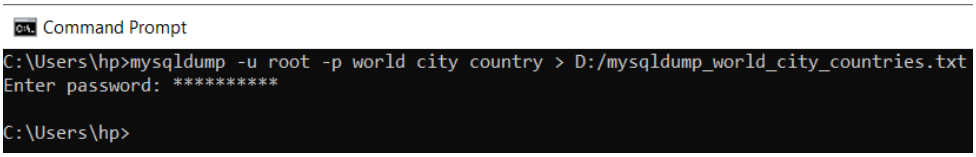


Open `D:\mysqldump_world_countries.txt` file where you can see the create and insert scripts for the `country` table.

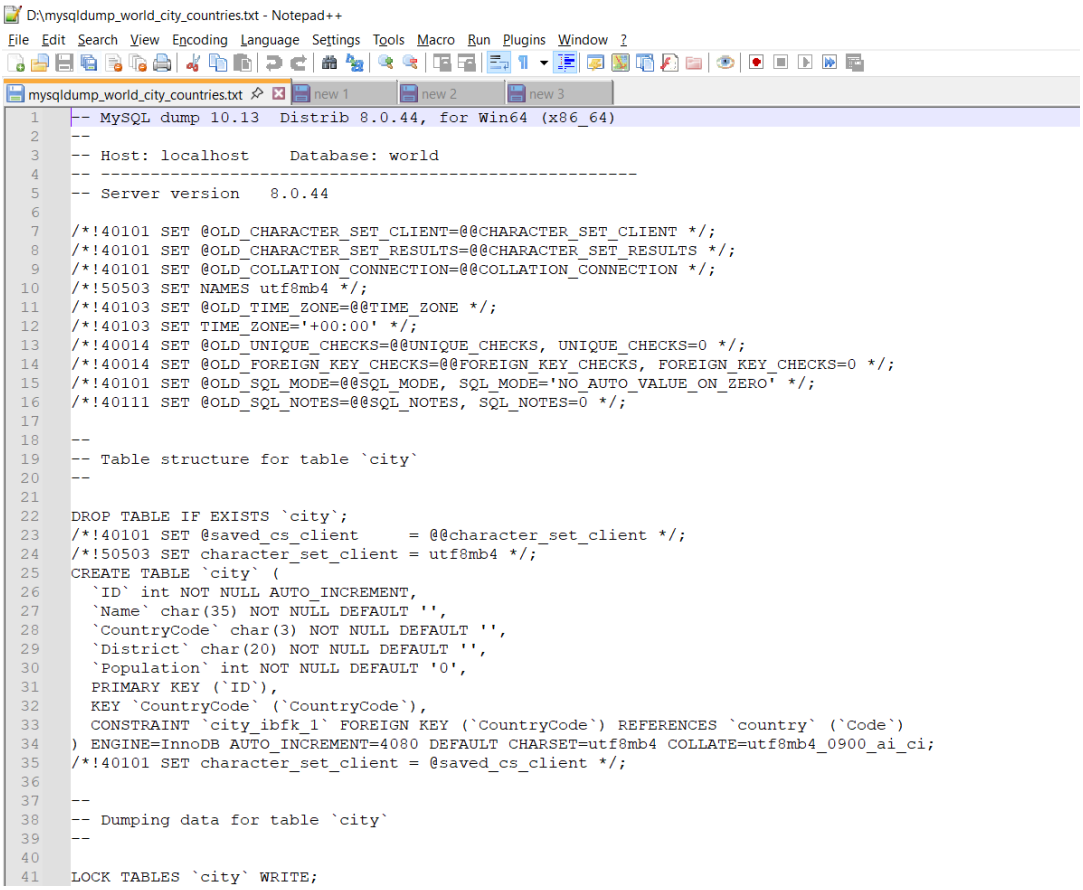


Use the following command to export `city` and `country` tables in `world` database to `D:\mysqldump_world_city_countries.txt` file in SQL format. Enter `root` user password when prompted:

```
mysqldump -u root -p world city country >
D:/mysqldump_world_city_countries.txt
```

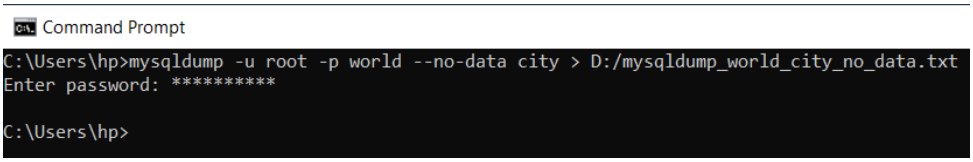


Open `D:\mysqldump_world_city_countries.txt` file where you can see the create and insert scripts for the `city` and `country` tables.

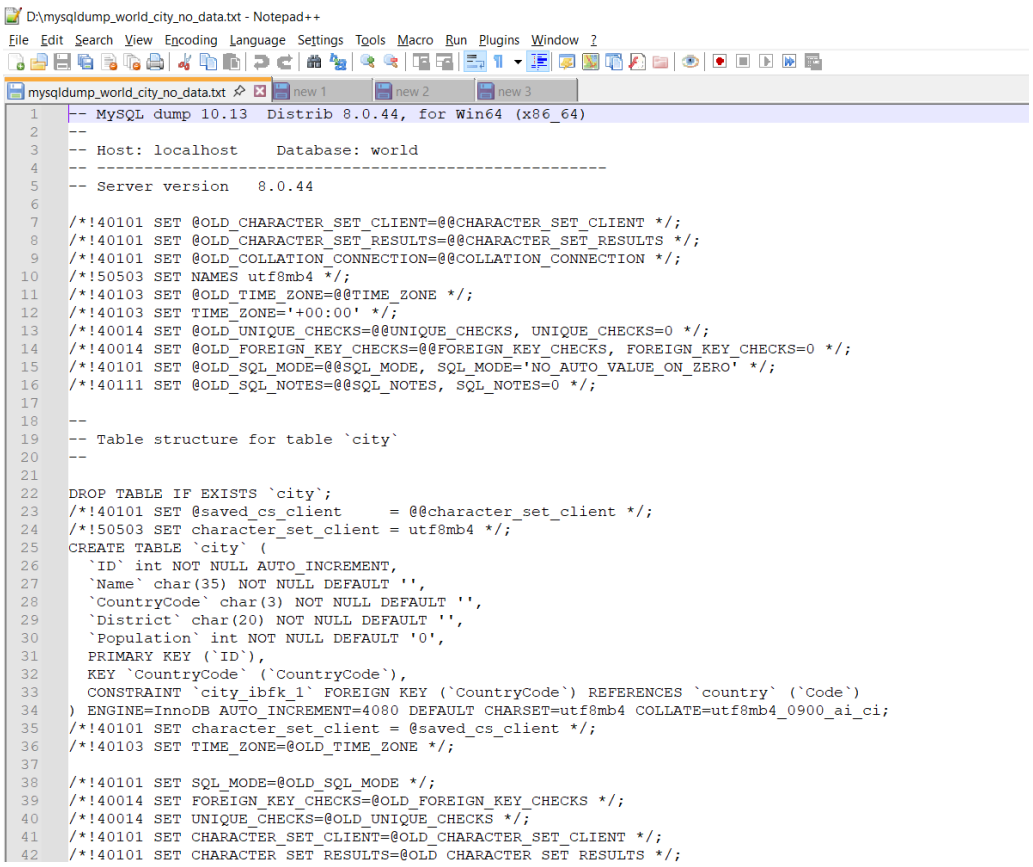


Use the following command to export `city` table in `world` database with CREATE script and no INSERT scripts. Enter `root` password when prompted:

```
mysqldump -u root -p world --no-data city >
D:/mysqldump_world_city_no_data.txt
```

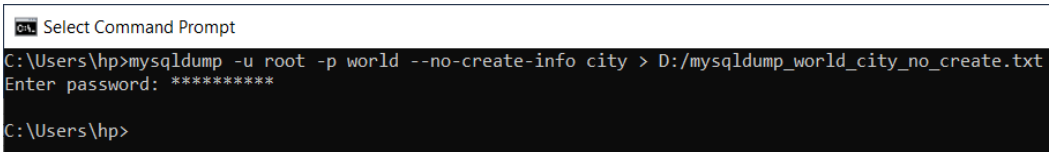


Open `D:\mysqldump_world_city_no_data.txt` file where you can see the create script for the `city` table.

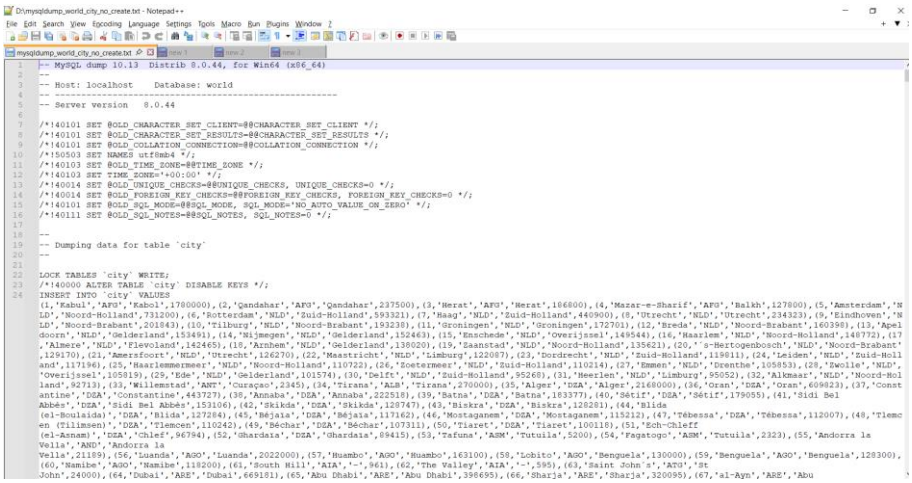


Use the following command to export `city` table in `world` database with only INSERT scripts. Enter `root` user password when prompted:

```
mysqldump -u root -p world --no-create-info city >
D:/mysqldump_world_city_no_create.txt
```



Open `D:\mysqldump_world_city_no_create.txt` file where you can see the insert scripts for the `city` table.

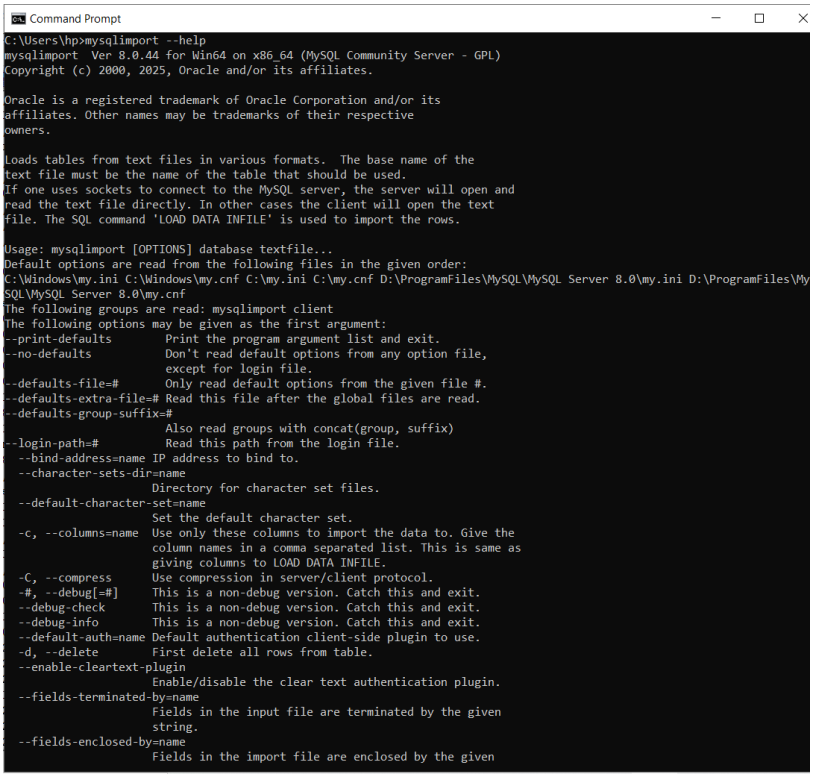


7.5. mysqlimport:

`mysqlimport` client can help to import data from a file into a table. It works as an alternative to `LOAD DATA` SQL statement.

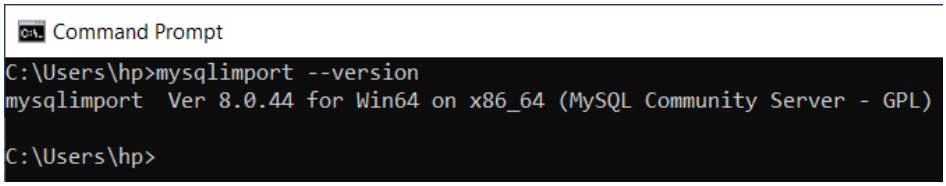
`mysqlimport` supports various options which can be specified in the command line. Use the following command to get the list of options supported by `mysqlimport` and commands to be executed:

```
mysqlimport --help
```



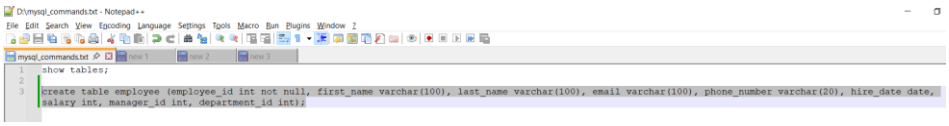
For example, use the following commands to get the current `mysqlimport` client version:

```
mysqlimport -V
```
or
```
mysqlimport --version
```



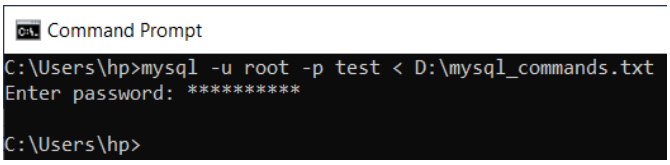
Before importing data from a file, first create the respective table in the database. For this, open `D:\mysql_commands.txt` file and enter the following commands in the file:

```
create table employee (employee_id int not null, first_name varchar(100),
last_name varchar(100), email varchar(100), phone_number varchar(20),
hire_date date, salary int, manager_id int, department_id int);
```



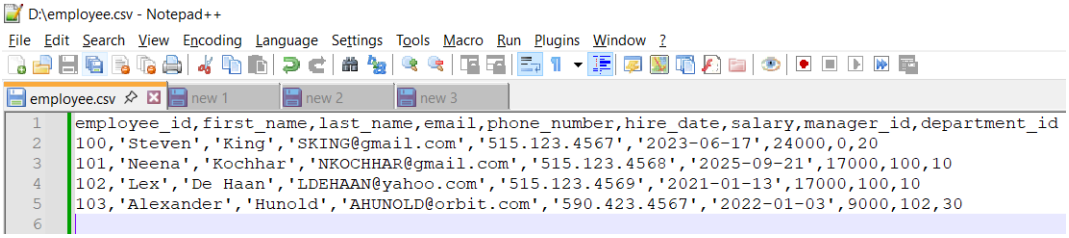
Then run the below `mysql` command to create table in `test` database. Enter `root` user password when prompted:

```
mysql -u root -p test < D:\mysql_commands.txt
```



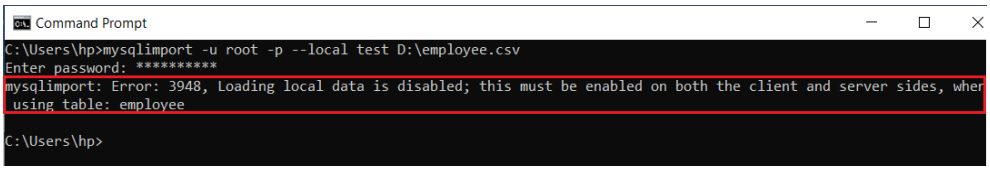
Now, create a sample file named `employee.csv` file in `D:\` and enter the following contents:

```
employee_id,first_name,last_name,email,phone_number,hire_date,salary,manager
_id,department_id
100,'Steven','King','SKING@gmail.com','515.123.4567','2023-06-17',24000,0,20
101,'Neena','Kochhar','NKOCHHAR@gmail.com','515.123.4568','2025-09-
21',17000,100,10
102,'Lex','De Haan','LDEHAAN@yahoo.com','515.123.4569','2021-01-
13',17000,100,10
103,'Alexander','Hunold','AHUNOLD@orbit.com','590.423.4567','2022-01-
03',9000,102,30
```



Use the following command to import data from `D:\employee.csv` file into `test` database. Enter `root` user password when prompted:

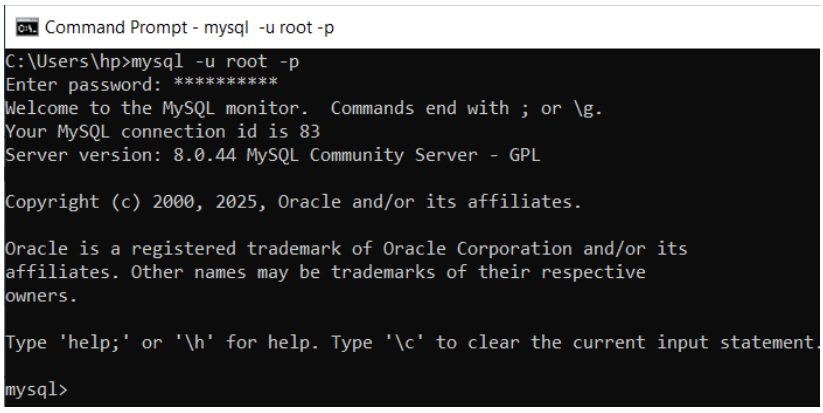
```
mysqlimport -u root -p --local test D:\employee.csv
```



When the above command is executed, it might throw ***mysqlimport: Error: 3948, Loading local data is disabled; this must be enabled on both the client and server sides, when using table: employee*****.** This is because `local_infile` variable is disabled by default.

First run the following command to connect to `mysql` client. Enter `root` user password when prompted:

```
mysql -u root -p
```



Then, run the following SQL command to verify the status of `local_infile` variable:

```
show global variables like 'local_infile';
```

```
mysql> show global variables like 'local_infile';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| local_infile  | OFF   |
+---------------+-------+
1 row in set (0.01 sec)

mysql>
```

Here, it displays the value OFF which means `local_infile` is disabled.

Use the following command to enable it:

```
set global local_infile=true;
```

```
mysql> set global local_infile=true;
Query OK, 0 rows affected (0.00 sec)

mysql> show global variables like 'local_infile';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| local_infile  | ON    |
+---------------+-------+
1 row in set (0.00 sec)

mysql>
```

Then exit `mysql` client and run the following `mysqlimport` command. Enter `root` user password when prompted:

```
mysqlimport -u root -p --local test D:\employee.csv
```

```
Command Prompt
C:\Users\hp>mysqlimport -u root -p --local test D:\employee.csv
Enter password: **********
test.employee: Records: 5  Deleted: 0  Skipped: 0  Warnings: 45

C:\Users\hp>
```

Use the following `mysqlimport` command to import data from file using specific options. Enter `root` user password when prompted:

```
mysqlimport -u root -p --local --fields-terminated-by="," --lines-
terminated-by="\n" test D:\employee.csv
```

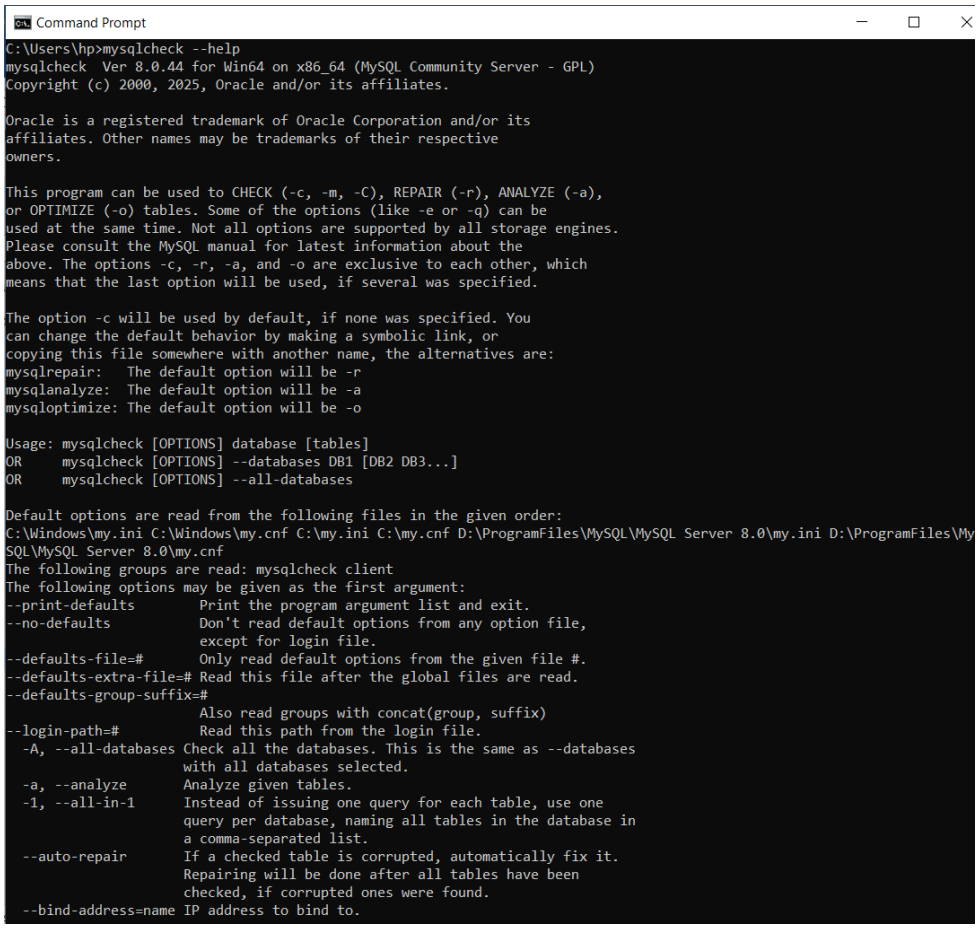
```
Command Prompt
C:\Users\hp>mysqlimport -u root -p --local --fields-terminated-by="," --lines-terminated-by="\n" test D:\employee.csv
Enter password: **********
test.employee: Records: 5  Deleted: 0  Skipped: 0  Warnings: 13

C:\Users\hp>
```

7.6. mysqlcheck:

`mysqlcheck` client is useful to for table maintenance operations such as checking table, repairing, analyzing and optimizing tables.

`mysqlcheck` supports various options which can be specified in the command line. Use the following command to get the list of options supported by `mysqlcheck` and commands to be executed:

```
mysqlcheck --help
```



For example, use the following commands to get the current `mysqlcheck` client version:

```
mysqlcheck -V
```

or

```
mysqlcheck --version
```

```
Command Prompt
C:\Users\hp>mysqlcheck -V
mysqlcheck  Ver 8.0.44 for Win64 on x86_64 (MySQL Community Server - GPL)

C:\Users\hp>
```

Use the following command to check the `employee` table in `test` database for any errors. Enter `root` user password when prompted:

```
mysqlcheck -u root -p test employee
```

```
Command Prompt
C:\Users\hp>mysqlcheck -u root -p test employee
Enter password: **********
test.employee                                     OK

C:\Users\hp>
```

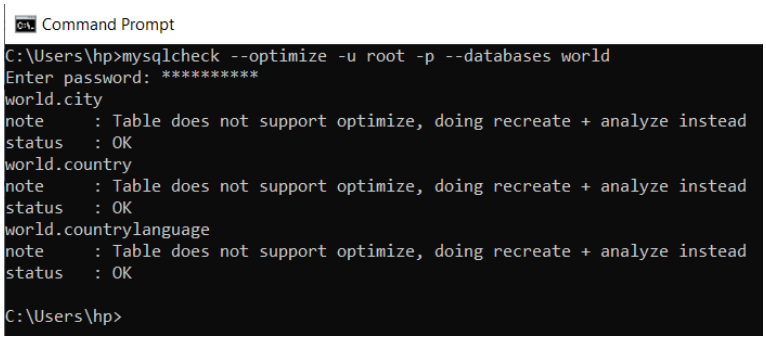
Use the following command to analyze tables in all databases. Enter `root` user password when prompted:

```
mysqlcheck --analyze -u root -p --all-databases
```

```
Command Prompt
C:\Users\hp>mysqlcheck --analyze -u root -p --all-databases
Enter password: **********
mysql.columns_priv                                OK
mysql.component                                   OK
mysql.db                                          OK
mysql.default_roles                               OK
mysql.engine_cost                                 OK
mysql.func                                        OK
mysql.general_log
note     : The storage engine for the table doesn't support analyze
mysql.global_grants                               OK
mysql.gtid_executed                               OK
mysql.help_category                               OK
mysql.help_keyword                                OK
mysql.help_relation                               OK
mysql.help_topic                                  OK
mysql.innodb_index_stats                          OK
mysql.innodb_table_stats                          OK
mysql.ndb_binlog_index                            OK
mysql.password_history                            OK
mysql.plugin                                      OK
mysql.procs_priv                                  OK
mysql.proxies_priv                                OK
mysql.replication_asynchronous_connection_failover OK
mysql.replication_asynchronous_connection_failover_managed OK
mysql.replication_group_configuration_version     OK
mysql.replication_group_member_actions            OK
mysql.role_edges                                  OK
mysql.server_cost                                 OK
mysql.servers                                     OK
mysql.slave_master_info                           OK
mysql.slave_relay_log_info                        OK
mysql.slave_worker_info                           OK
mysql.slow_log
note     : The storage engine for the table doesn't support analyze
mysql.tables_priv                                 OK
mysql.time_zone                                   OK
mysql.time_zone_leap_second                       OK
mysql.time_zone_name                              OK
mysql.time_zone_transition                        OK
mysql.time_zone_transition_type                   OK
mysql.user                                        OK
sakila.actor                                      OK
sakila.address                                    OK
sakila.category                                   OK
sakila.city                                       OK
sakila.country                                    OK
sakila.customer                                   OK
sakila.film                                       OK
sakila.film_actor                                 OK
sakila.film_category                              OK
sakila.film_text                                  OK
sakila.inventory                                  OK
sakila.language                                   OK
sakila.payment                                    OK
sakila.rental                                     OK
sakila.staff                                      OK
sakila.store                                      OK
sys.sys_config                                    OK
test.employee                                     OK
world.city                                        OK
world.country                                     OK
world.countrylanguage                             OK

C:\Users\hp>
```

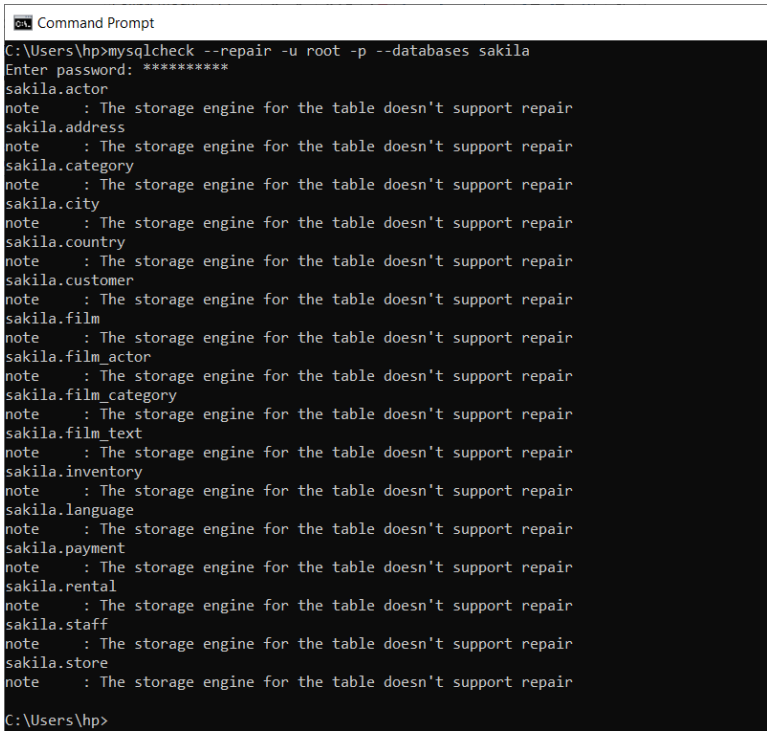
Use the following command to optimize tables in `world` database. Enter `root` user password when prompted:

```
mysqlcheck --optimize -u root -p --databases world
```



Use the following command to repair tables in `sakila` database. Enter `root` user password when prompted:

```
mysqlcheck --repair -u root -p --databases sakila
```



8. MySQL Shell:

MySQL Shell (`mysqlsh`) is an advanced command line interface to work with MySQL instances. It works similar to `mysql` to run **SQL** commands but it also offers scripting capabilities for **JavaScript** and **Python** languages. In addition to traditional database management, MySQL Shell provides APIs such as the **X DevAPI** for developing applications in JavaScript or Python that treat MySQL as both a relational and document store, and the **AdminAPI** for administering MySQL instances and configuring high-availability solutions like InnoDB Cluster, InnoDB ClusterSet, and InnoDB ReplicaSet by facilitating automated setup, provisioning, monitoring, etc.

MySQL Shell can execute code in three modes – JavaScript or Python or SQL – and process code either interactively or non-interactively (batch processing). By default, it processes the code in JavaScript mode which can be changed in two ways:

- Using `--sql` (for SQL mode) or `--py` (for Python mode) or `--js` (for JavaScript mode) options while starting `mysqlsh`.
- Using `\sql` or `\py` or `\js` commands after entering into MySQL Shell.

While MySQL Shell is installed along with MySQL Server, it can also be installed as a standalone tool on client machines from the MySQL Community Downloads page.

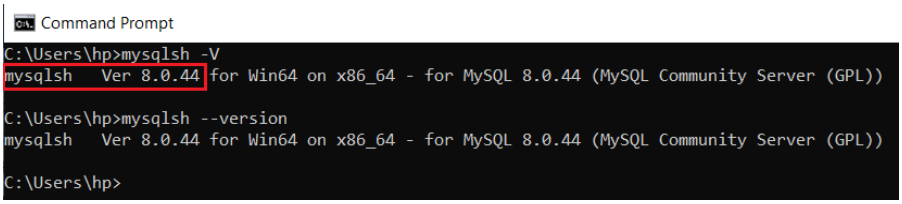
8.1. Verify MySQL Shell Installation:

Verify if `mysqlsh` has been installed successfully using the following command:

```
mysqlsh -V
```

or

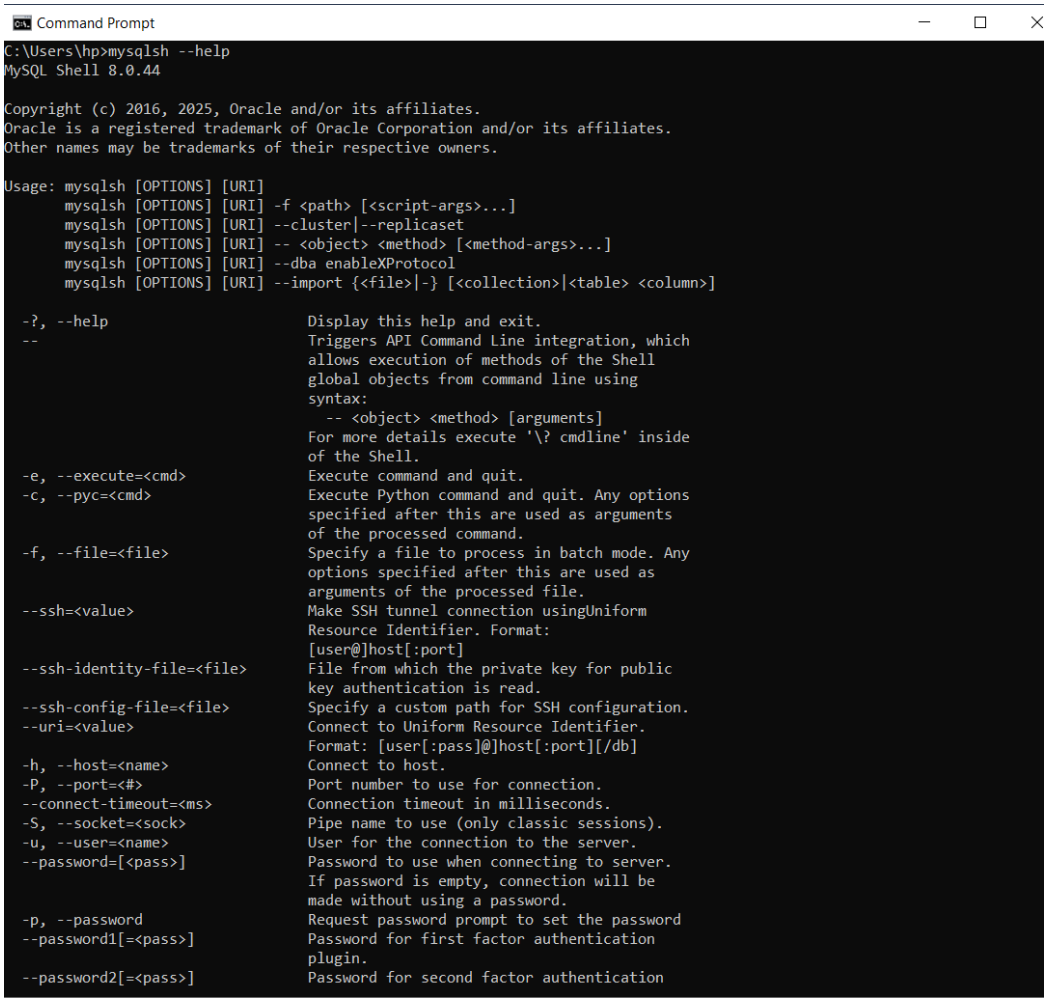
```
mysqlsh --version
```



Here, it displays the `mysqlsh` version 8.0.44 version that was installed on the Windows system.

`mysqlsh` supports various options which can be specified in the command line. Use the following command to get the list of options supported by `mysqlsh`.

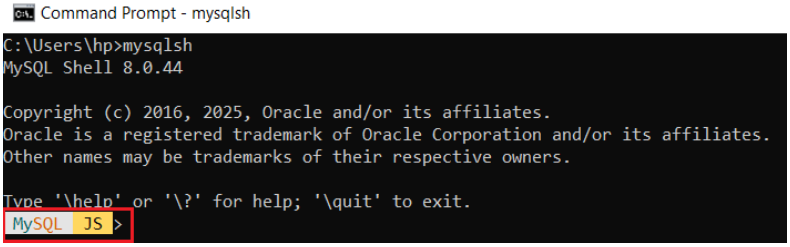
```
mysql --help
```



8.2. Start MySQL Shell:

Run the following command to start MySQL Shell:

```
mysqlsh
```



Once MySQL Shell is successfully connected, it displays **`MySQL JS >`** prompt which executes JavaScript code in interactive mode.

8.3. MySQL Shell Commands:

MySQL Shell provides many built-in commands to perform tasks like connecting to a MySQL instance, switching current scripting language, running reports, utilizing various utilities, etc. To ensure these commands operate independently of current scripting execution mode, they must always be prefixed with the backslash (`\`) escape character.

Use the following basic command to get more help on `mysqlsh` usage and available shell commands:

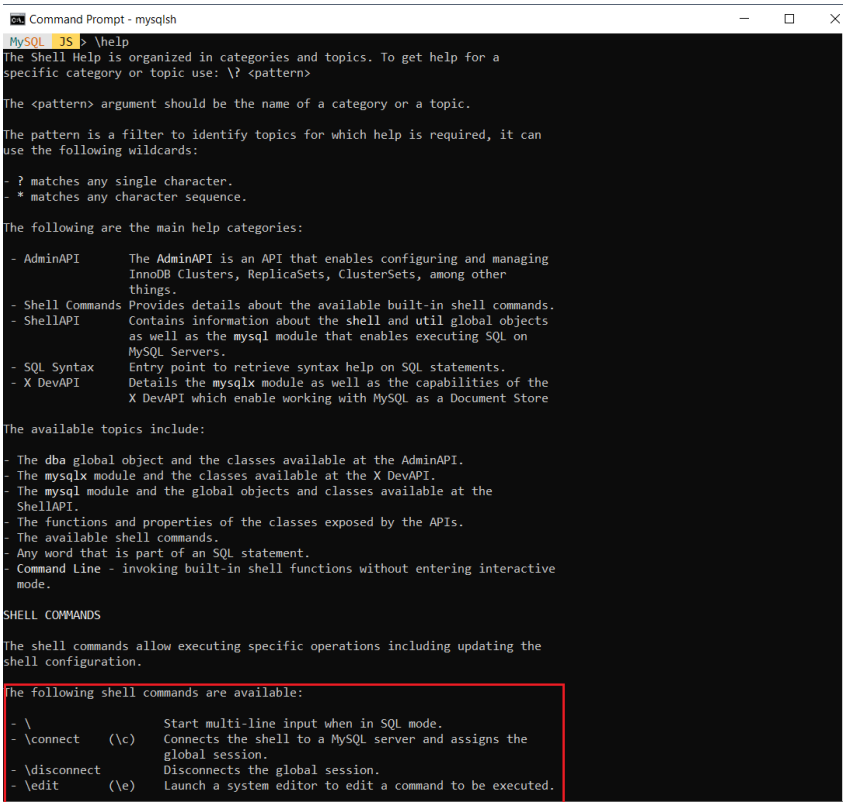
```
\?
```

or

```
\h
```

or

```
\help
```



Following are some common commands being used:

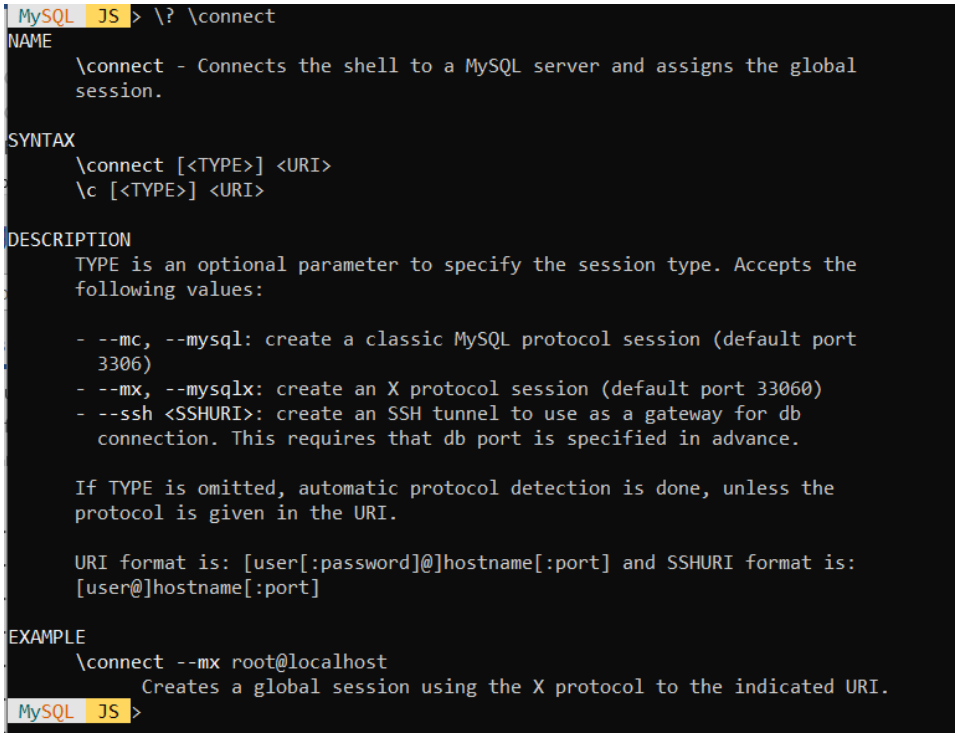
- `\?` or `\h` or `\help`: Displays general help about MySQL Shell, or allows to search the online documentation.
- `\q` or `\quit` or `\exit`: Exits MySQL Shell.
- `\c` or `\connect`: Connects to a MySQL instance.
- `\reconnect`: Reconnects to the same MySQL instance.

- `\disconnect`: Disconnects the MySQL instance.
- `\s` or `\status`: Displays the current MySQL Shell status.
- `\js`: Switches execution mode to JavaScript.
- `\py`: Switches execution mode to Python.
- `\sql`: Switches execution mode to SQL.
- `\`: Allows to write multi-line code in SQL mode.
- `\u` or `\use`: Allows to specify schema to use.
- `\.` or `\source`: Executes script file using the active language.
- `\history`: Allows to view and edit command line history.
- `\option`: Modifies MySQL Shell configuration options.
- `\show`: Runs the specified report using the provided options and arguments.
- `!` or `\system`: Executes the specified shell command.

To get more help of each of these commands, prefix the command name with `\?` or `\h` or `\help`.

For example, use the following command to get more help on `\connect` command:

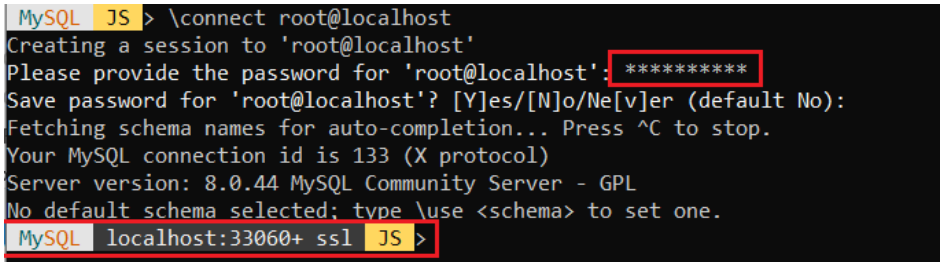
```
\? \connect
```



8.4. Connect to MySQL Instance:

In MySQL Shell, run the following command to connect to a MySQL instance. Enter `root` user password when prompted after which it asks to save password for the user *(Enter Y or N if you wish to save or not)*:

```
\connect root@localhost
```



After successfully connected, the prompt will change to `MySQL localhost:33060+ ssl JS >` to run commands.

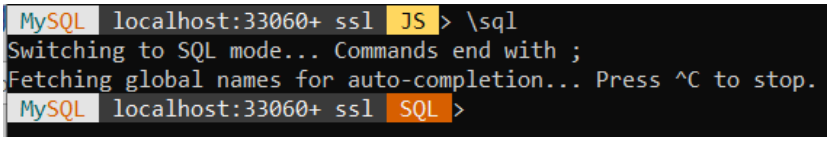
Note that `mysqlsh` also allows to connect to a MySQL instance before starting the shell. For example, use the below command to connect to MySQL and switch to SQL mode:

```
mysqlsh -h localhost -u root --sql
```

8.5. Run SQL Commands:

In MySQL Shell, first run the following command to switch to SQL execution mode:

```
\sql
```



Now the prompt changes to `MySQL localhost:33060+ ssl SQL >` to run SQL commands terminating with semi-colon.

Run the following command to list the available databases:

```
show databases;
```

```
MySQL  localhost:33060+ ssl  SQL > show databases;
+--------------------+
| Database           |
+--------------------+
| database           |
| information_schema |
| mysql              |
| performance_schema |
| sakila             |
| sys                |
| test               |
| world              |
+--------------------+
8 rows in set (0.0018 sec)
MySQL  localhost:33060+ ssl  SQL >
```

Run the following command to see the list of tables in the `test` database:

```
show tables from test;
```

```
MySQL  localhost:33060+ ssl  SQL > show tables from test;
+----------------+
| Tables_in_test |
+----------------+
| employee       |
+----------------+
1 row in set (0.0025 sec)
MySQL  localhost:33060+ ssl  SQL >
```

Run the following command to see the list of columns in the `employee` table in `test` database:

```
show columns from test.employee;
```

```
MySQL  localhost:33060+ ssl  SQL > show columns from test.employee;
+---------------+-------------+------+-----+---------+-------+
| Field         | Type        | Null | Key | Default | Extra |
+---------------+-------------+------+-----+---------+-------+
| employee_id   | int         | NO   |     | NULL    |       |
| first_name    | varchar(100)| YES  |     | NULL    |       |
| last_name     | varchar(100)| YES  |     | NULL    |       |
| email         | varchar(100)| YES  |     | NULL    |       |
| phone_number  | varchar(20) | YES  |     | NULL    |       |
| hire_date     | date        | YES  |     | NULL    |       |
| salary        | int         | YES  |     | NULL    |       |
| manager_id    | int         | YES  |     | NULL    |       |
| department_id | int         | YES  |     | NULL    |       |
+---------------+-------------+------+-----+---------+-------+
9 rows in set (0.0026 sec)
MySQL  localhost:33060+ ssl  SQL >
```

8.6. Run Python Code:

In MySQL Shell, first run the following command to switch to Python execution mode:

```
\python
```

```
MySQL  localhost:33060+ ssl  SQL > \py
Switching to Python mode...
MySQL  localhost:33060+ ssl  Py >
```

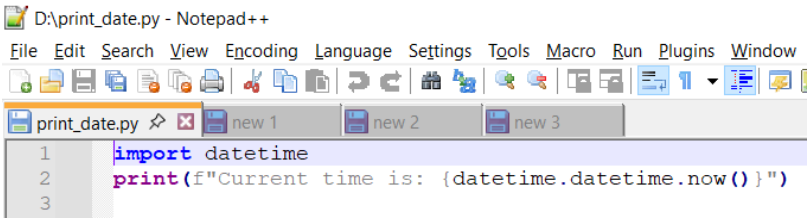
Now the prompt changes to `MySQL  localhost:33060+ ssl  Py >` to run Python code.

Run the following code to display current date and time:

```
import datetime
print(f"Current time is: {datetime.datetime.now()}")
```

```
MySQL  localhost:33060+ ssl  Py > import datetime
MySQL  localhost:33060+ ssl  Py > print(f"Current time is: {datetime.datetime.now()}")
Current time is: 2026-01-06 11:49:23.905700
MySQL  localhost:33060+ ssl  Py >
```

Instead of executing Python code line by line, MySQL Shell can execute the Python file directly. First save the above code in `print_date.py` file in **D** drive.



Run the following command in MySQL Shell to execute the `D:\print_date.py` file:

```
\. D:\print_date.py
```

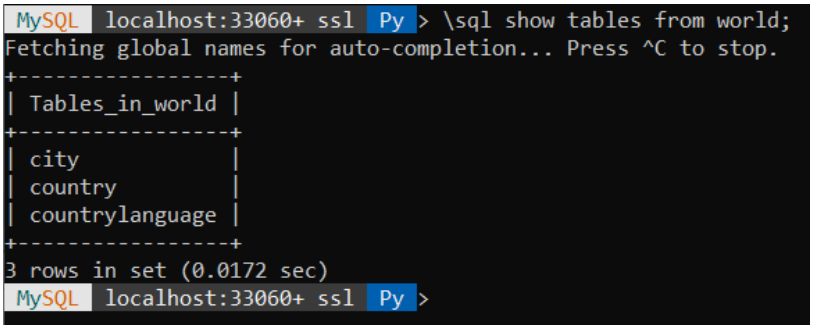
```
MySQL  localhost:33060+ ssl  Py > \. D:\print_date.py
Current time is: 2026-01-06 12:11:49.167570
MySQL  localhost:33060+ ssl  Py >
```

Note that `mysqlsh` also allows to execute Python script without starting shell.
For example, use any of the below commands to run `D:\print_date.py` file using `mysqlsh`:

```
mysqlsh --py < D:\print_date.py
```
or
```
mysqlsh --py --file D:\print_date.py
```

While remaining in Python mode in MySQL Shell, single SQL query can be executed by prefixing with `\sql` before the SQL command as below:

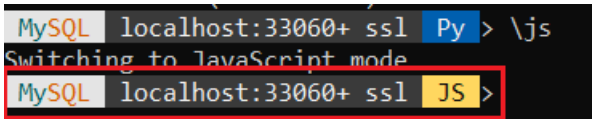
```
\sql show tables from world;
```



8.7. Run JavaScript Code:

In MySQL Shell, first run the following command to switch to JavaScript execution mode:

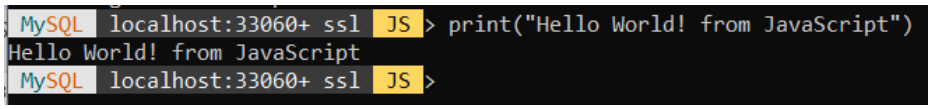
```
\js
```



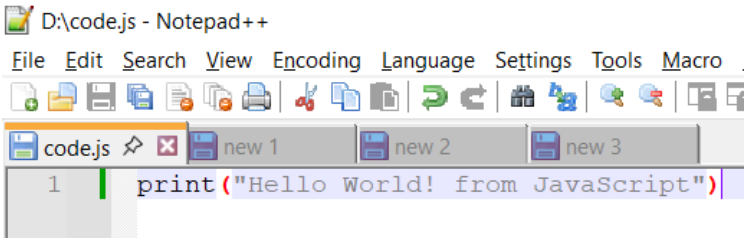
Now the prompt changes to `MySQL localhost:33060+ ssl JS >` to run JavaScript code.

Run the following code to display a message:

```
print("Hello World! from JavaScript")
```

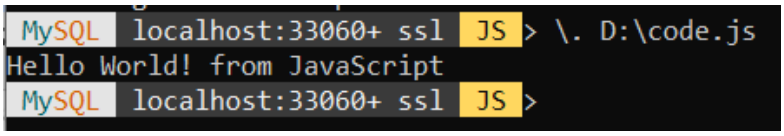


Instead of executing JavaScript code line by line, MySQL Shell can execute a JavaScript file directly.
First save the above code in `code.js` file in **D** drive.



Run the following command in MySQL Shell to execute the `D:\code.js` file:

```
\. D:\code.js
```



Note that `mysqlsh` also allows to execute JavaScript file without starting shell.
For example, use any of the below commands to run `D:\code.js` file using `mysqlsh`:

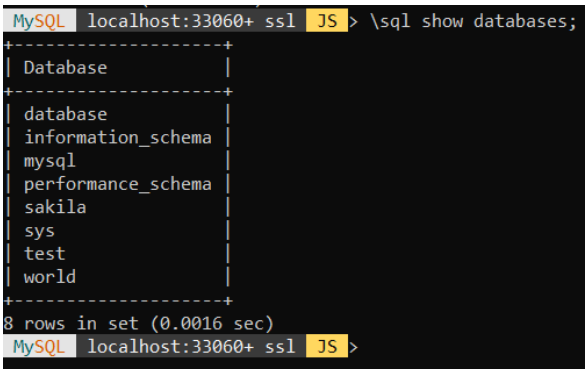
```
mysqlsh --js < D:\code.js
```

or

```
mysqlsh --js --file D:\code.js
```

While remaining in JavaScript mode in MySQL Shell, single SQL query can be executed by prefixing with `\sql` before the SQL command as below:

```
\sql show databases;
```



8.8. MySQL Shell Utilities:

MySQL Shell provides built-in utilities that are useful to perform various administrative and maintenance tasks such as data import/export, diagnostics collection and server upgrades. These utilities are accessible through global object called `util` which is available in JavaScript and Python modes only.

8.8.1. Dump Utilities:

Dump utilities include `util.dumpInstance()` which exports data from a MySQL instance, `util.dumpSchemas()` which exports data from specific schemas, and `util.dumpTables()` which exports data from selected tables or views into a set of local

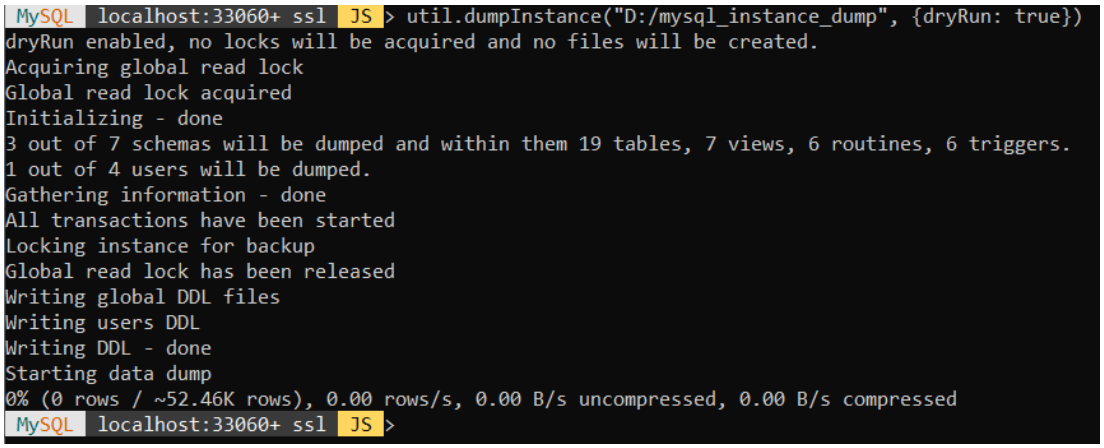
files or an Oracle Cloud Infrastructure (OCI) Object Storage bucket. For complete details, refer to the official MySQL documentation.

Follow the below syntax to use these dump utilities:

```
util.dumpInstance(outputUrl[, options])
util.dumpSchemas(schemas, outputUrl[, options])
util.dumpTables(schema, tables, outputUrl[, options])
```

For example, run the following command to perform a dry run to check for any compatibility issues before performing a dump to `D:\mysql_instance_dump` directory using `dumpInstance` utility:

```
util.dumpInstance("D:/mysql_instance_dump", {dryRun: true})
```

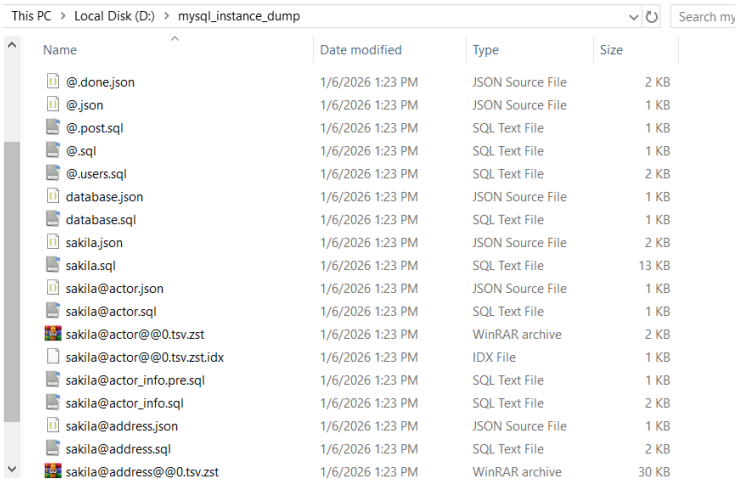


Since there are no issues reported, it is ready for actual data dump. Run the following command to data dump to a local `D:\mysql_instance_dump` directory:

```
util.dumpInstance("D:/mysql_instance_dump")
```

```
MySQL  localhost:33060+ ssl  JS > util.dumpInstance("D:/mysql_instance_dump")
Acquiring global read lock
Global read lock acquired
Initializing - done
3 out of 7 schemas will be dumped and within them 19 tables, 7 views, 6 routines, 6 triggers.
1 out of 4 users will be dumped.
Gathering information - done
All transactions have been started
Locking instance for backup
Global read lock has been released
Writing global DDL files
Writing users DDL
Running data dump using 4 threads.
NOTE: Progress information uses estimated values and may not be accurate.
Writing schema metadata - done
Writing DDL - done
Writing table metadata - done
Starting data dump
100% (52.57K rows / ~52.46K rows), 0.00 rows/s, 0.00 B/s uncompressed, 0.00 B/s compressed
Dump duration: 00:00:00s
Total duration: 00:00:00s
Schemas dumped: 3
Tables dumped: 19
Uncompressed data size: 3.23 MB
Compressed data size: 802.72 KB
Compression ratio: 4.0
Rows written: 52570
Bytes written: 802.72 KB
Average uncompressed throughput: 3.23 MB/s
Average compressed throughput: 802.72 KB/s
MySQL  localhost:33060+ ssl  JS >
```

Once data dump is completed, it creates `mysql_instance_dump` folder in D drive with all necessary data that can imported to another MySQL instance.

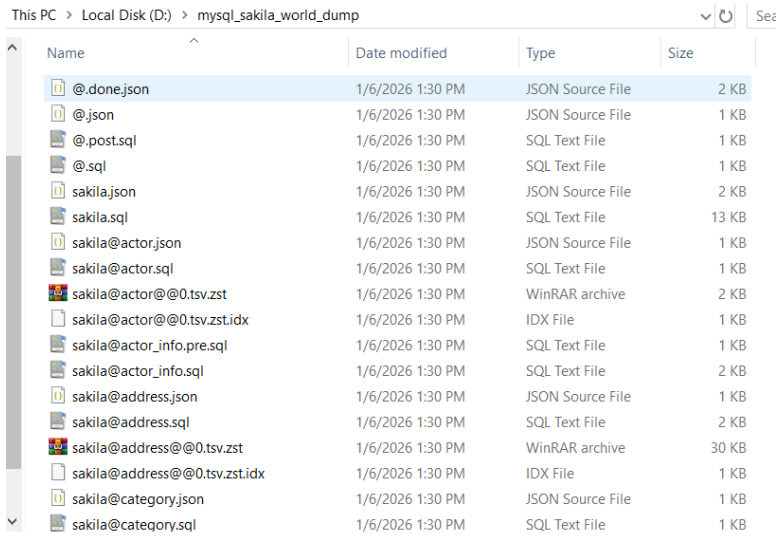


Run the following command to import `sakila` and `world` databases to `D:\mysql_sakila_world_dump` directory using `dumpSchemas` utility:

```
util.dumpSchemas(["sakila","world"],"D:/mysql_sakila_world_dump")
```

```
MySQL  localhost:33060+ ssl  JS > util.dumpSchemas(["sakila","world"],"D:/mysql_sakila_world_dump")
Acquiring global read lock
Global read lock acquired
Initializing - done
2 schemas will be dumped and within them 19 tables, 7 views, 6 routines, 6 triggers.
Gathering information - done
All transactions have been started
Locking instance for backup
Global read lock has been released
Writing global DDL files
Running data dump using 4 threads.
NOTE: Progress information uses estimated values and may not be accurate.
Writing schema metadata - done
Writing DDL - done
Writing table metadata - done
Starting data dump
100% (52.57K rows / ~52.46K rows), 0.00 rows/s, 0.00 B/s uncompressed, 0.00 B/s compressed
Dump duration: 00:00:00s
Total duration: 00:00:00s
Schemas dumped: 2
Tables dumped: 19
Uncompressed data size: 3.23 MB
Compressed data size: 802.72 KB
Compression ratio: 4.0
Rows written: 52570
Bytes written: 802.72 KB
Average uncompressed throughput: 3.23 MB/s
Average compressed throughput: 802.72 KB/s
MySQL  localhost:33060+ ssl  JS >
```

Once data dump is completed, it creates `mysql_sakila_world_dump` folder in D drive with all necessary data that can imported to another MySQL instance.

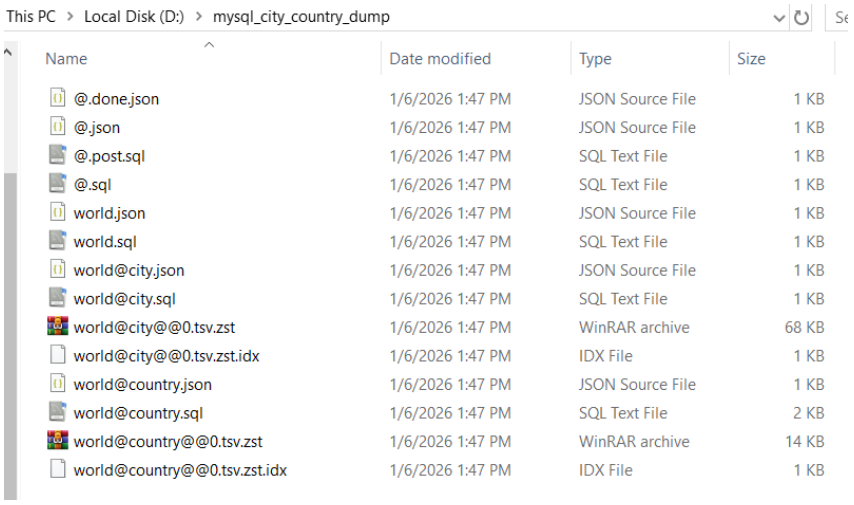


Run the following command to import `city` and `country` tables in the `world` database to `D:\mysql_city_country_dump` directory using `dumpTables` utility:

```
util.dumpTables("world", ["city","country"],
"D:/mysql_city_country_dump")
```

```
MySQL  localhost:33060+ ssl  JS > util.dumpTables("world", ["city","country"], "D:/mysql_city_country_dump")
Acquiring global read lock
Global read lock acquired
Initializing - done
2 tables and 0 views will be dumped.
Gathering information - done
All transactions have been started
Locking instance for backup
Global read lock has been released
Writing global DDL files
Running data dump using 4 threads.
NOTE: Progress information uses estimated values and may not be accurate.
Writing schema metadata - done
Writing DDL - done
Writing table metadata - done
Starting data dump
101% (4.32K rows / ~4.27K rows), 0.00 rows/s, 0.00 B/s uncompressed, 0.00 B/s compressed
Dump duration: 00:00:00s
Total duration: 00:00:00s
Schemas dumped: 1
Tables dumped: 2
Uncompressed data size: 176.37 KB
Compressed data size: 82.92 KB
Compression ratio: 2.1
Rows written: 4318
Bytes written: 82.92 KB
Average uncompressed throughput: 176.37 KB/s
Average compressed throughput: 82.92 KB/s
MySQL  localhost:33060+ ssl  JS >
```

Once data dump is completed, it creates `mysql_city_country_dump` folder in D drive with all necessary data that can imported to another MySQL instance.



8.8.2. Load Utility:

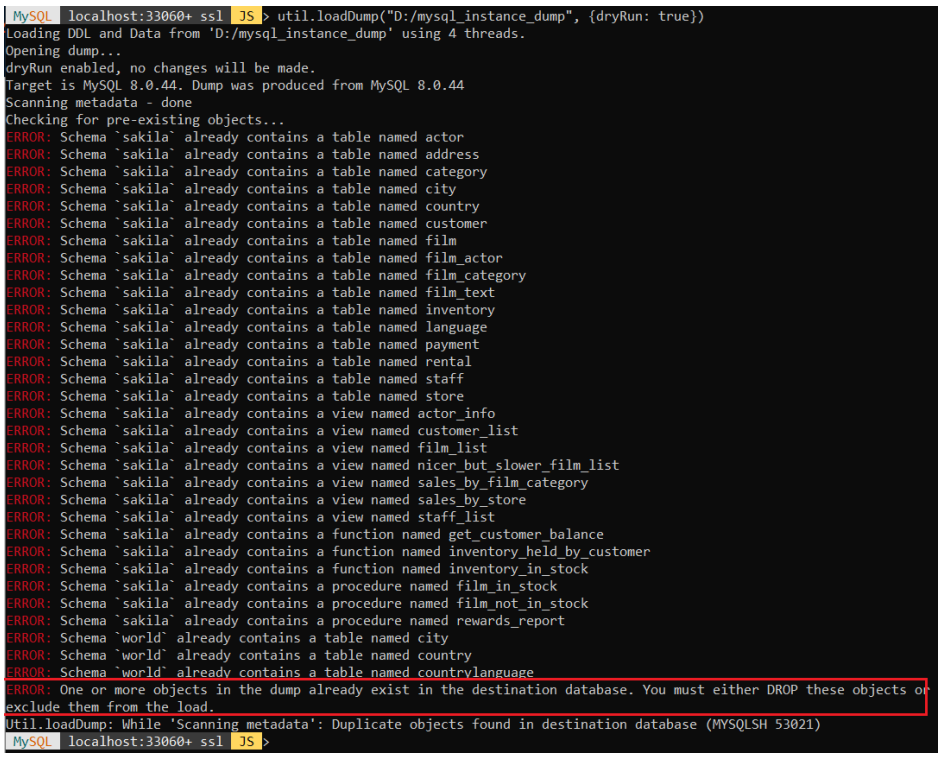
Loading utility include `util.loadDump()` which imports data that was exported using the dump utilities. It supports parallel loading of table chunks, progress tracking, and the ability to resume partially completed imports. For complete details, refer to the official MySQL documentation.

Follow the below syntax to use this load utility:

```
util.loadDump(url[, options])
```

For example, run the following command to perform a dry run to check for any compatibility issues before performing load:

```
util.loadDump("D:/mysql_instance_dump", {dryRun: true})
```



As you can see here, it reported many errors that the schemas already exist in the database and they should be dropped before performing the load.

8.8.3. Table Export Utility:

Table export utility include `util.exportTable()` which exports data from a MySQL relational table into a single data file stored either locally or an Oracle Cloud Infrastructure (OCI) Object Storage bucket in CSV, TSV and text formats. For complete details, refer to the official MySQL documentation.

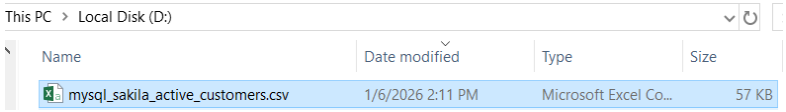
Follow the below syntax to use this utility:

```
util.exportTable(table, outputUrl[, options])
```

For example, run the following command to export active customers from `sakila.customer` table to a local `D:\mysql_sakila_active_customers.csv` file:

```
util.exportTable("sakila.customer",
"file:///D:/mysql_sakila_active_customers.csv", {"where":
"active=1"})
```

Once the table export is completed, it creates `mysql_sakila_active_customers.csv` file in D drive.



8.8.4. Parallel Table Import Utility:

Parallel Table Import Utility include `util.importTable()` which is useful for rapid import of large data files like CSV or TSV into a single relational table using parallel connections. It is much faster than a standard single-threaded `LOAD DATA` statement for large imports. It uses `LOAD DATA LOCAL INFILE` statements to import data which requires the `local_infile` system variable to be enabled before using this utility. For complete details, refer to the official MySQL documentation.

Follow the below syntax to use this utility:

```
util.importTable ({file_name | file_list}, options)
```

Let us try to import `D:\mysql_sakila_active_customers.csv` file into `test.customer` table using `importTable` utility.

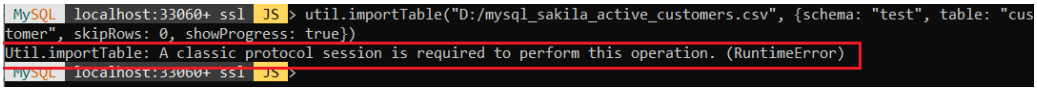
First, create `test.customer` table using the following command:

```
\sql CREATE TABLE test.customer (customer_id int NOT NULL
AUTO_INCREMENT,store_id int NOT NULL,first_name varchar(45) NOT
NULL,last_name varchar(45) NOT NULL,email varchar(50) DEFAULT
```

```
NULL,address_id int NOT NULL,active int NOT NULL DEFAULT
'1',create_date datetime NOT NULL,last_update timestamp NULL DEFAULT
CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,PRIMARY KEY
(`customer_id`));
```

Then run the following command to import `D:\mysql_sakila_active_customers.csv` file into `test.customer` table :

```
util.importTable("D:/mysql_sakila_active_customers.csv", {schema:
"test", table: "customer", skipRows: 0, showProgress: true})
```



The parallel table import utility throws runtime error *A classic protocol session is required to perform this operation* since it requires a classic connection to the target server and does not support X Protocol connections.

8.8.5. JSON Import Utility:

JSON Import Utility such as `util.importJSON()` imports JSON documents from a file to a MySQL relational table or a collection in JSON format. If the table or collection does not exist in the database, this utility creates the default table or collection and imports data. For complete details, refer to the official MySQL documentation.

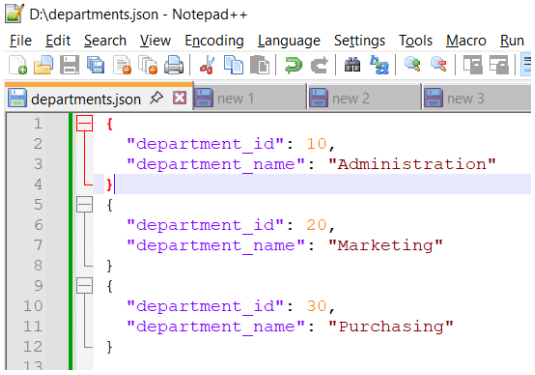
Follow the below syntax to use this utility:

```
util.importJSON (path, options)
```

First, create a simple JSON file `D:\departments.json` with the below contents:

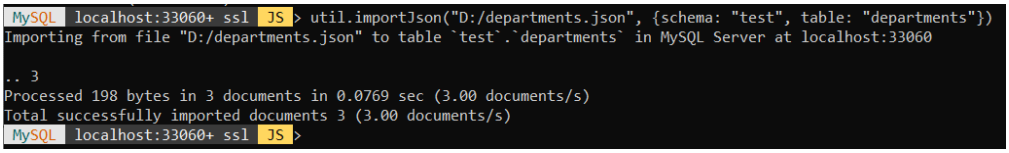
```
{
   "department_id": 10,
   "department_name": "Administration"
 }
 {
   "department_id": 20,
   "department_name": "Marketing"
 }
 {
   "department_id": 30,
   "department_name": "Purchasing"
 }
```

Note: MySQL Shell does not accept a JSON file containing square brackets and commas in between array contents.



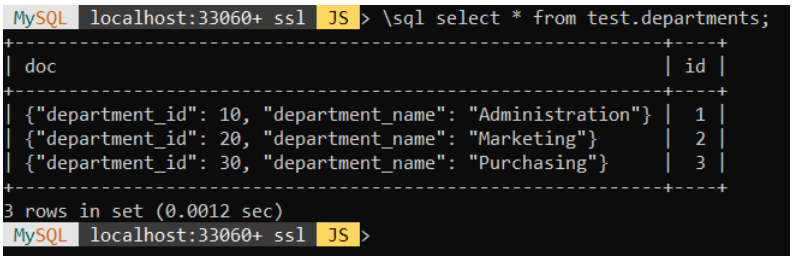
Then run the below command to import the above JSON file to a `test.departments` table in JSON format:

```
util.importJson("D:/departments.json", {schema: "test", table: "departments"})
```



Run the below command to verify data in `test.departments` table:

```
\sql select * from test.departments;
```



8.8.6. Copy Utilities:

Copy utilities include `util.copyInstance()`, `util.copySchemas()`, and `util.copyTables()` which copy DDL and stream data directly between two MySQL instances without the need for intermediate local storage. It combines both the dump and load operations into one step. These utilities uses `LOAD DATA LOCAL INFILE` statements to

import data which requires the `local_infile` system variable to be enabled. For complete details, refer to the official MySQL documentation.

Follow the below syntax to use these copy utilities:

```
util.copyInstance(connectionData[, options])
util.copySchemas(schemaList, connectionData[, options])
util.copyTables(schemaName, tablesList, connectionData[, options])
```

8.8.7. Upgrade Checker Utility:

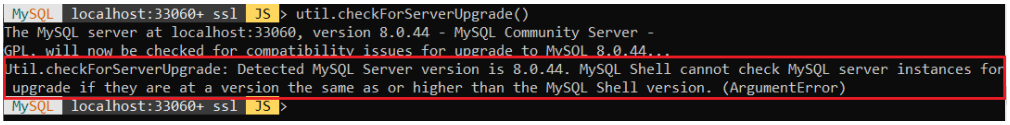
Upgrade checker utility include `util.checkForServerUpgrade()` performs checks on a MySQL server instance to determine its readiness for an upgrade to a newer version. For complete details, refer to the official MySQL documentation.

Follow the below syntax to use this utility:

```
util.checkForServerUpgrade (ConnectionData connectionData, Dictionary
options)
```

For example, run the following command to check for server upgrade:

```
util.checkForServerUpgrade()
```



Here, it displays an error *MySQL Shell cannot check MySQL server instances for upgrade if they are at a version the same as or higher than the MySQL Shell version* since MySQL Shell and MySQL server are both at same version.

8.8.8. Diagnostic Utilities:

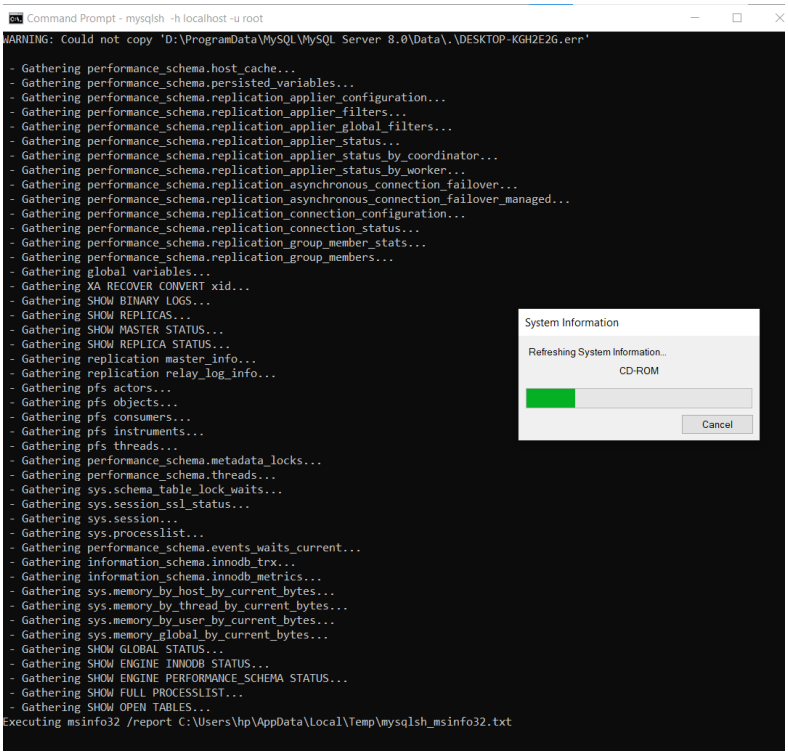
Diagnostic utilities include `util.debug.collectDiagnostics()`, `util.debug.collectHihgLoadDiagnostics()`, `util.debug.collectSlowQueryDiagnostics()` which gather diagnostic information and performance metrics from the connected MySQL server and generate diagnostic reports on overall health, performance and SQL queries in YAML and TSV formats that help in troubleshooting system issues. For complete details, refer to the official MySQL documentation.

Follow the below syntax to use these utilities:

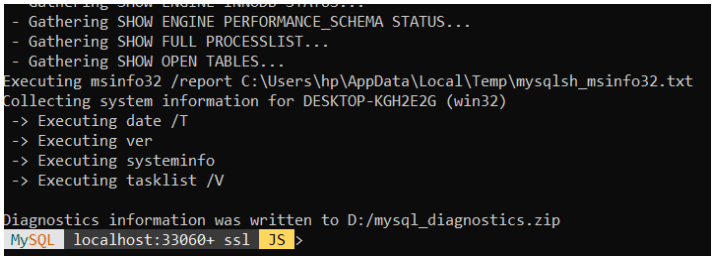
```
util.debug.collectDiagnostics("path/",{options})
util.debug.collectHighLoadDiagnostics(path, {options})
util.debug.collectSlowQueryDiagnostics("path", "query", {options})
```

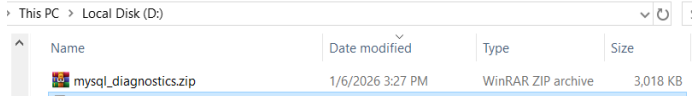
For example, run the following command to collect diagnostics of the current system:

```
util.debug.collectDiagnostics("D:/mysql_diagnostics")
```



It collects system information and writes to `D:\mysql_diagnostics.zip` file.

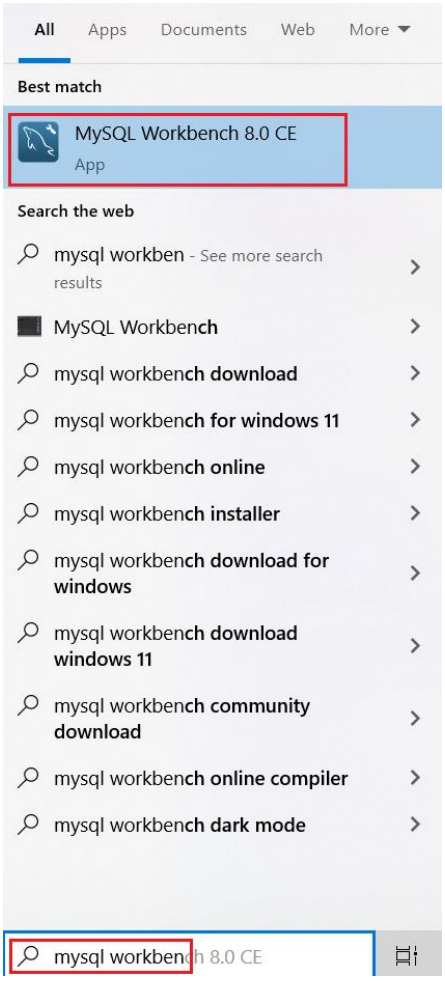




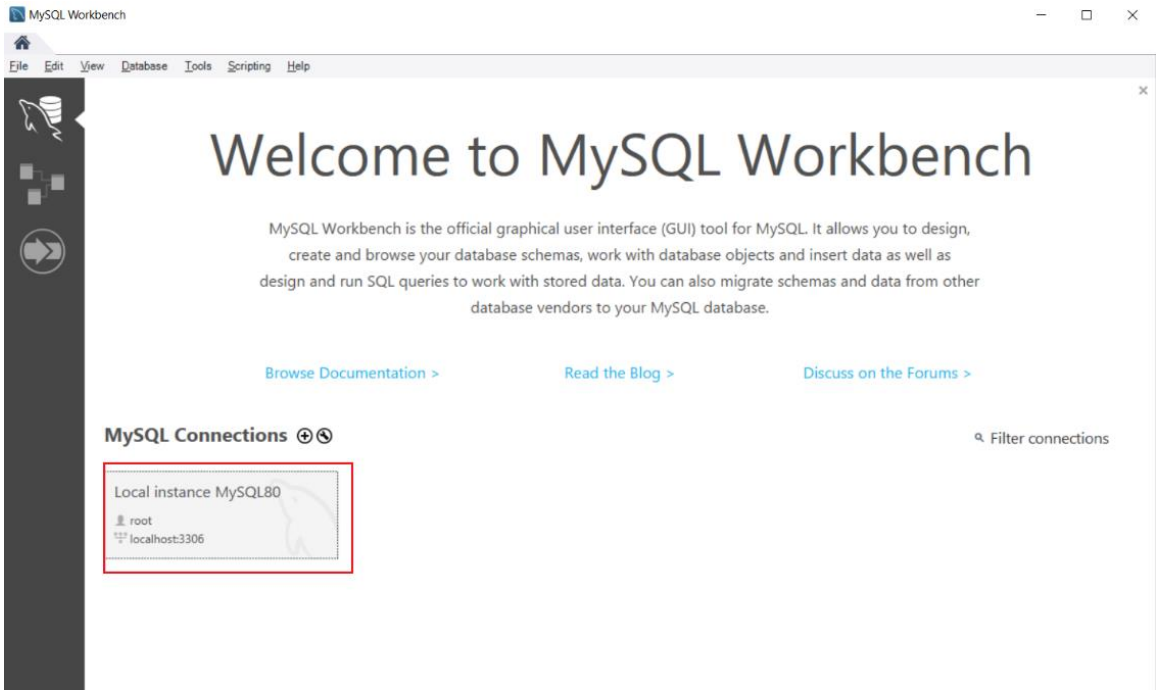
9. MySQL Workbench:

MySQL Workbench is a visual tool that provides a graphical user interface for working with MySQL databases that simplifies creating, executing and optimizing SQL queries.

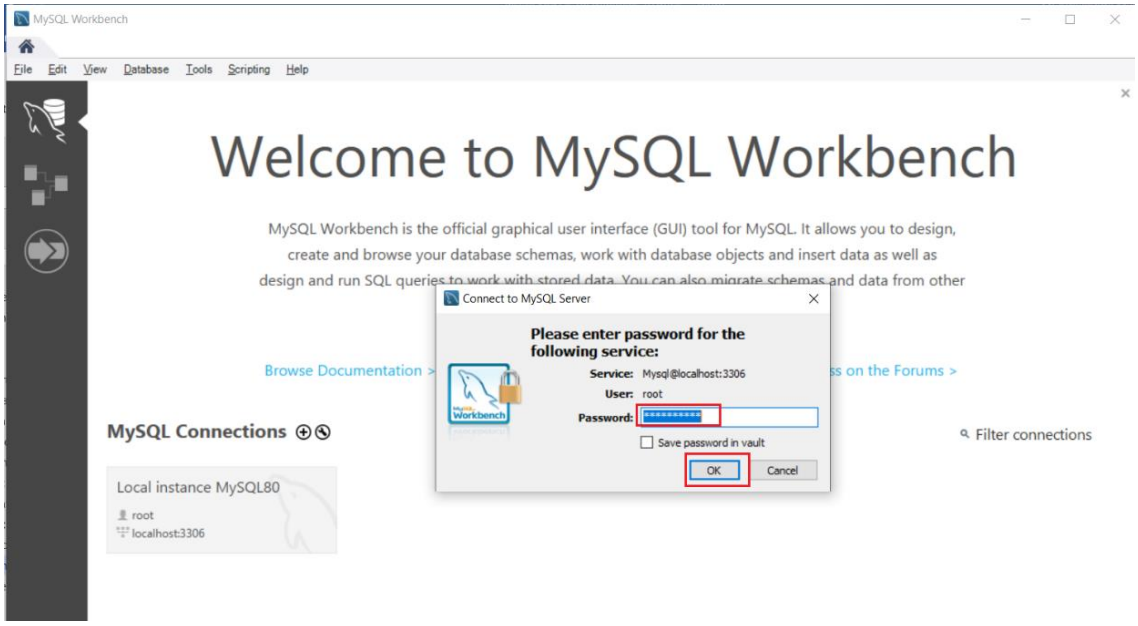
To open MySQL Workbench, search for **mysql workbench** in the task bar and choose the first option:



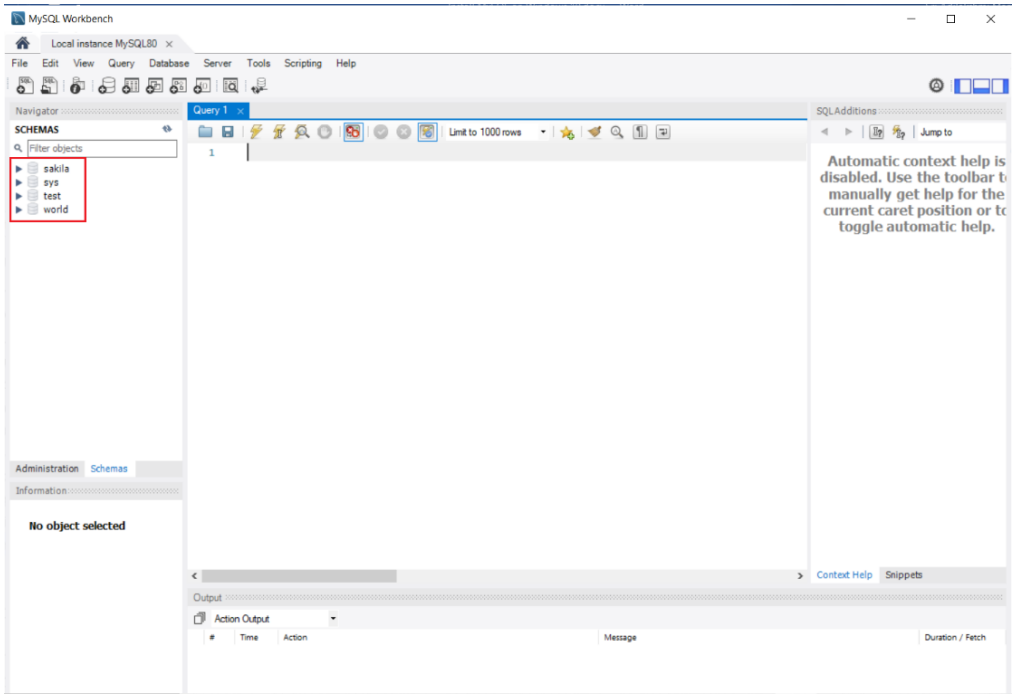
On the home screen, it displays the list of MySQL connections. Select **Local instance MySQL** to connect to the MySQL server installed locally:



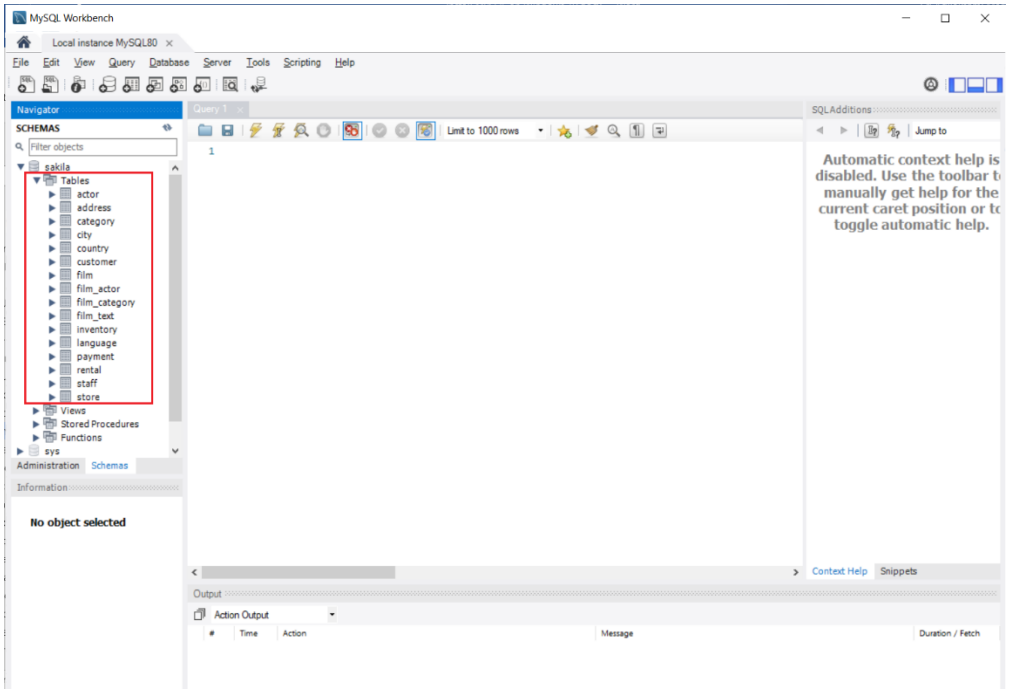
It prompts for root password and enter the password given at the time of installation and press **OK**.



Once it is successfully connected to the local MySQL server, it displays the list of databases available in the server:

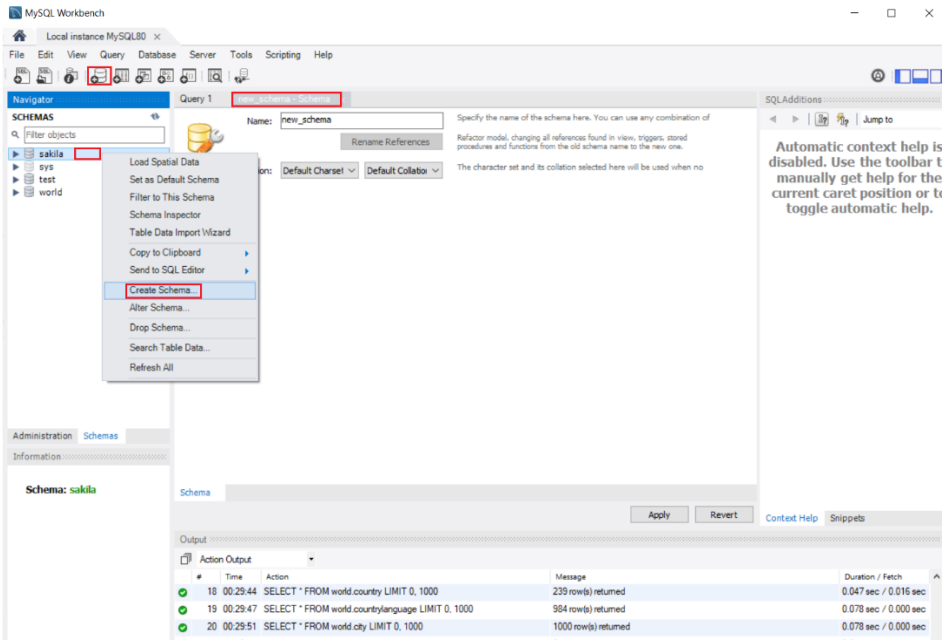


To see the list of tables in a database, expand the database name (for instance, `sakila`) and expand **Tables**.

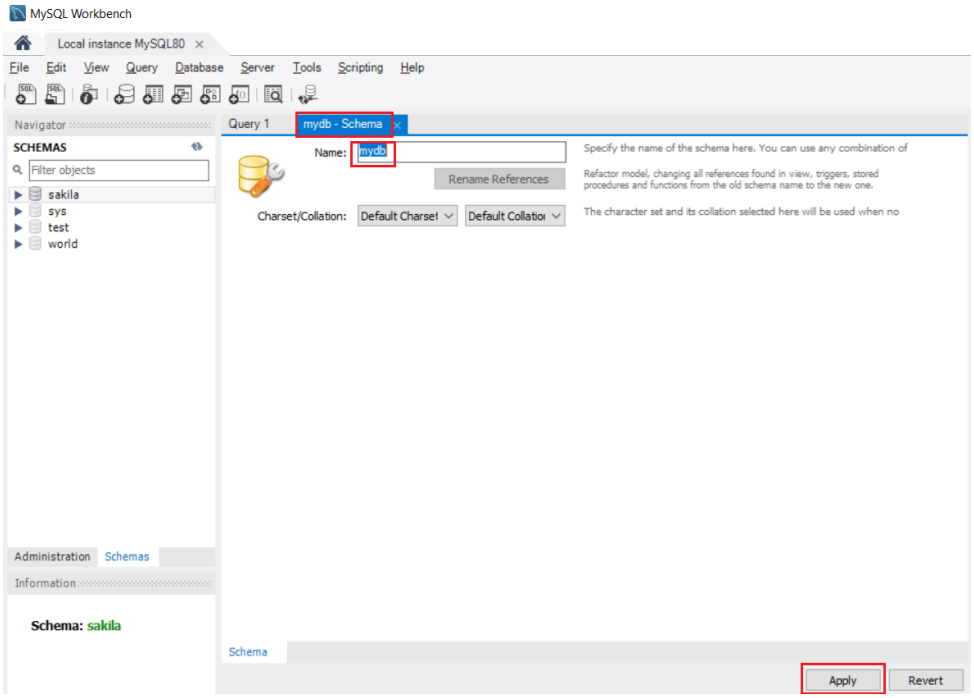


9.1. Create Database:

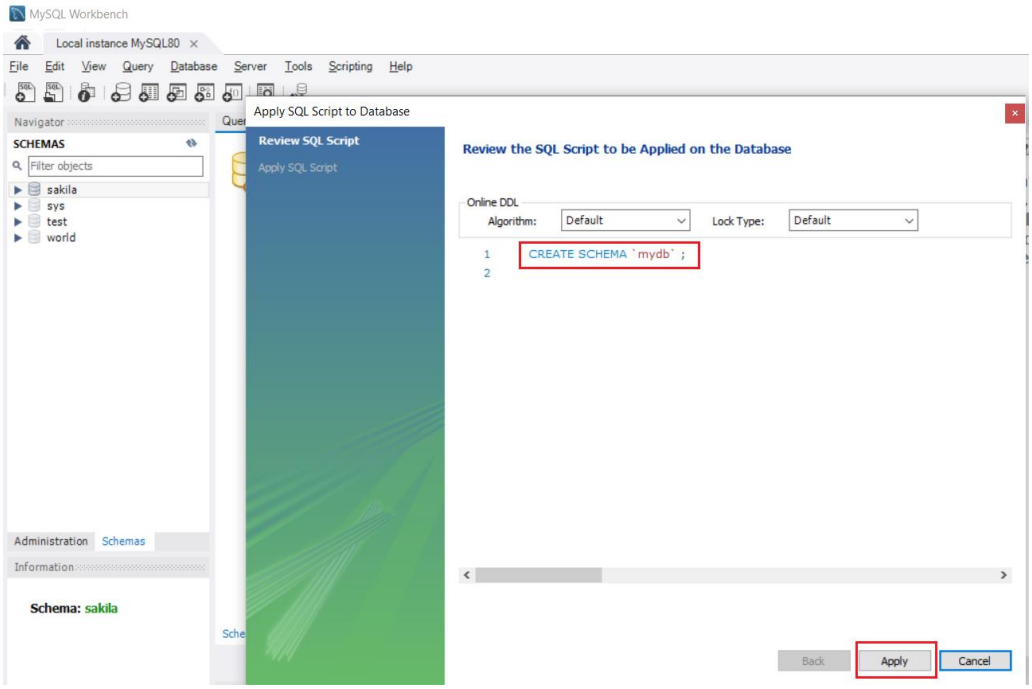
To create a new database, click on **Create new schema** icon on the top menu or right click on existing schema and select **Create Schema** option which opens **new_schema - Shema** tab.



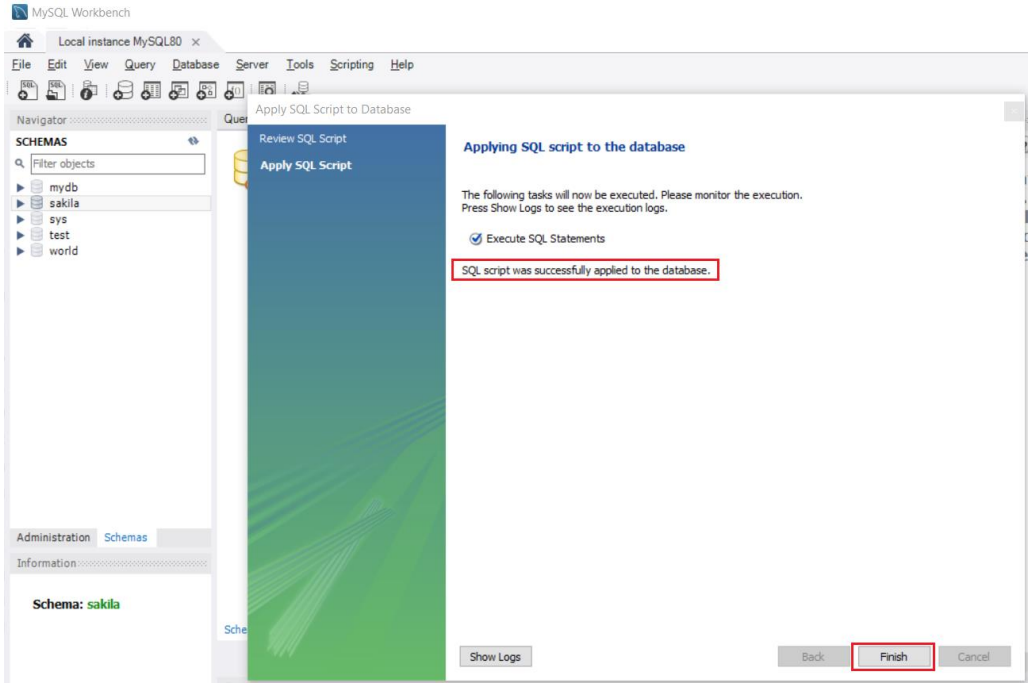
Provide the new database name such as `mydb` (after which you can observe the tab name changes to **mydb - Schema**) and click on **Apply** button.



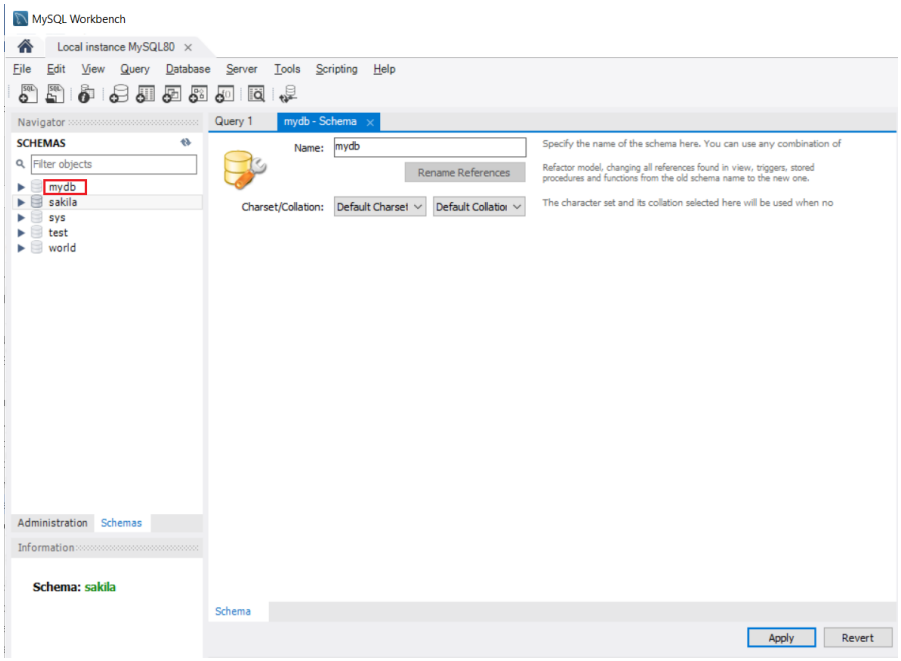
Review the SQL script provided and click on **Apply** button.



It executes the task and displays message that SQL script was successfully applied to the database. Click on **Finish** button to close.

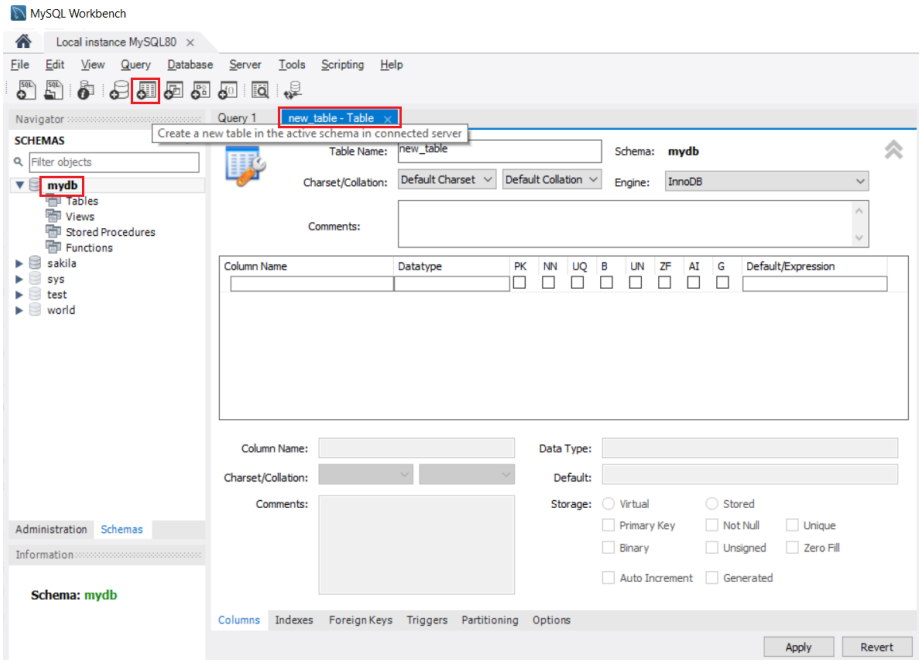


Under **Schemas** list, the new database `mydb` is visible. Close **mydb - Schema** tab.

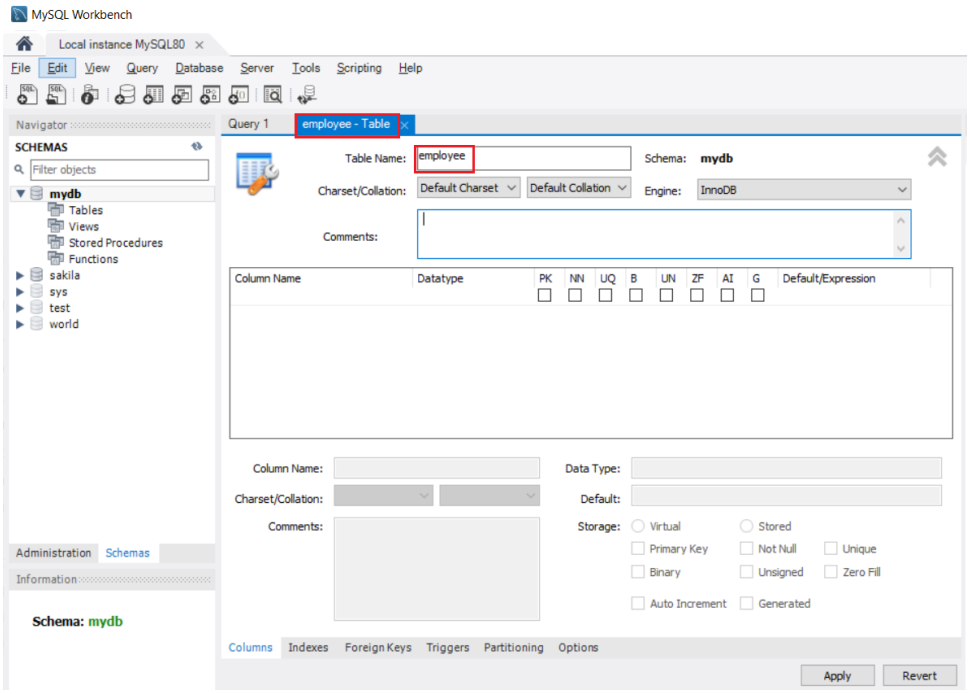


9.2. Create Table:

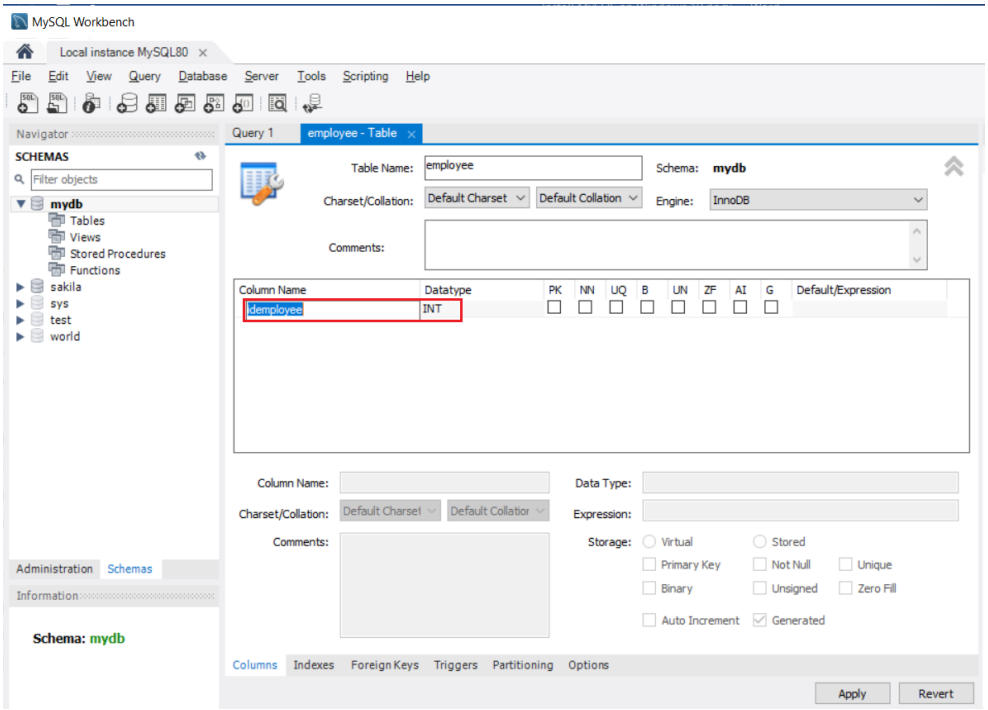
To create a new table, double click on `mydb` which selects the table and click on **Create new table** icon on the top menu which opens **new_table - Table** tab.



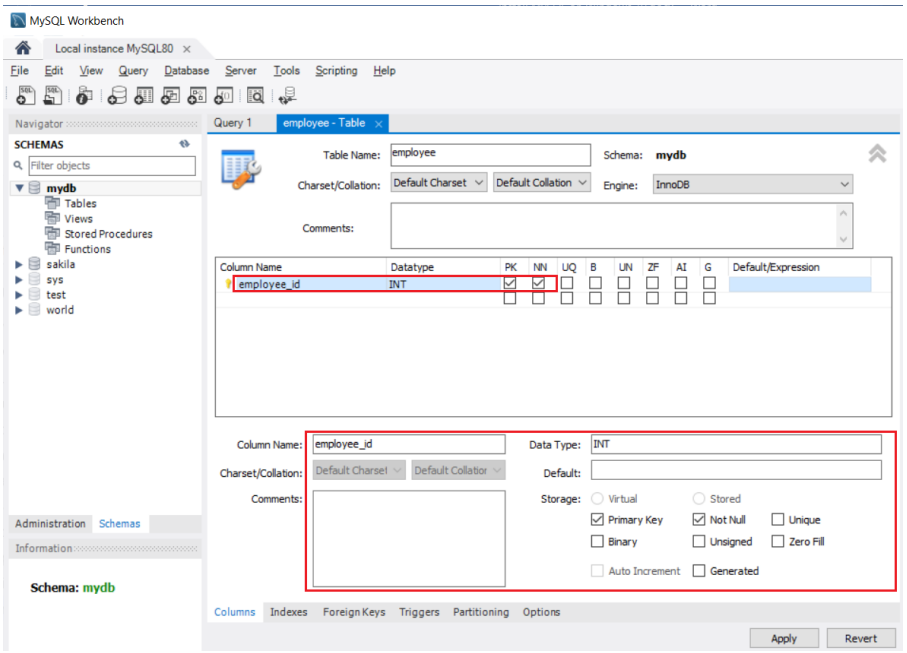
Provide the new table name such as `employee` (after which you can observe the tab name changes to **employee - Table**).



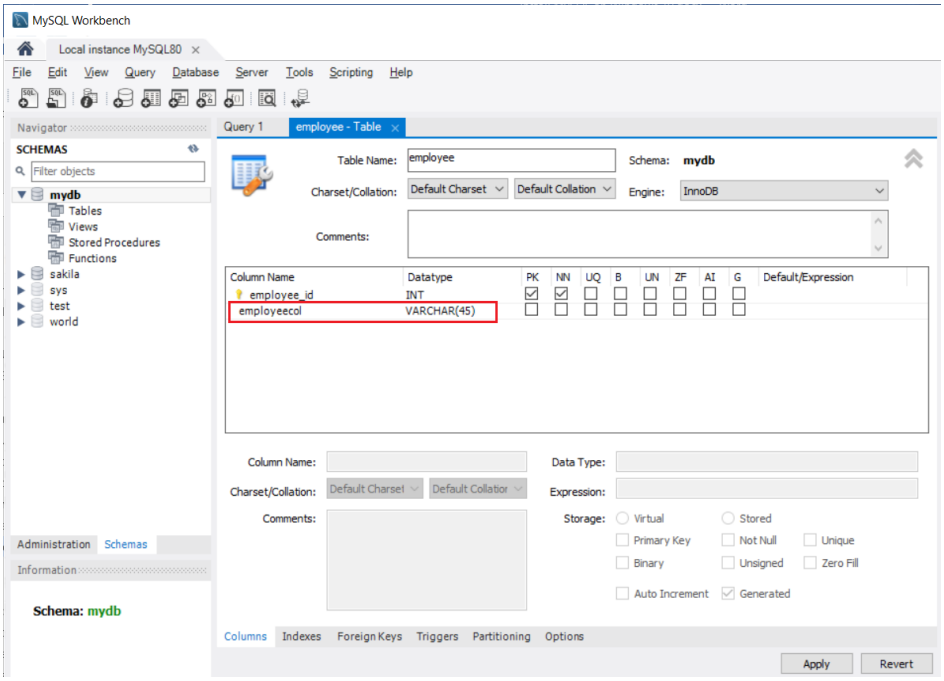
Click under **Column Name** which by default displays as `idemployee` and **Datatype** as `INT`**.**



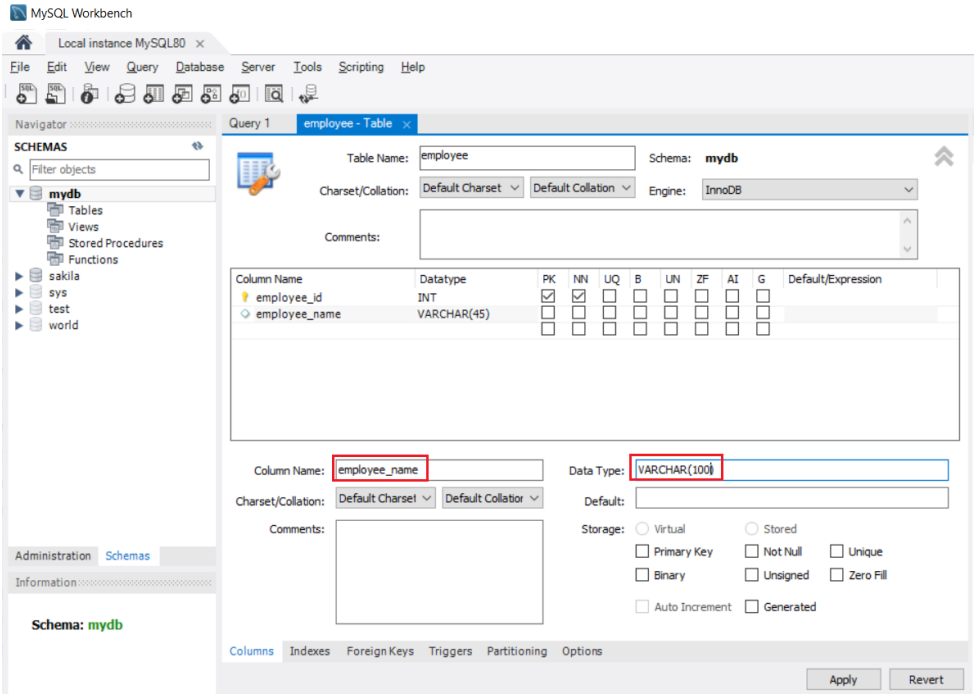
Update the **Column Name** to `employee_id` and keep **Datatype** as `INT` and it automatically selects **PK** (Primary Key) and **NN** (Not Null).



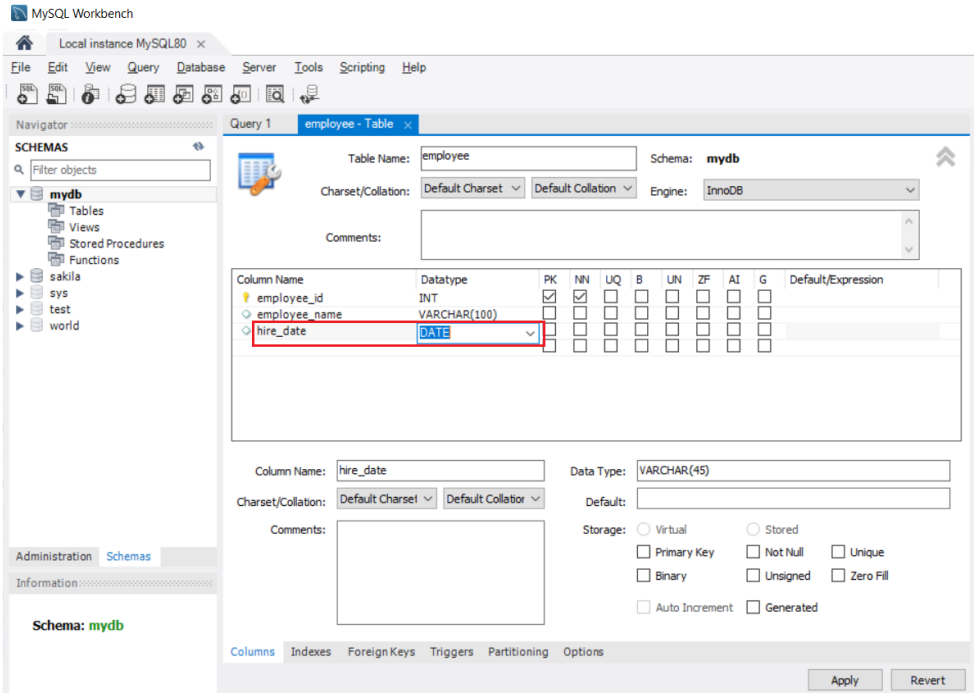
Then select other column where by default it displays **Column Name** as `employeecol` and **Datatype** as `VARCHAR(45)`**.**



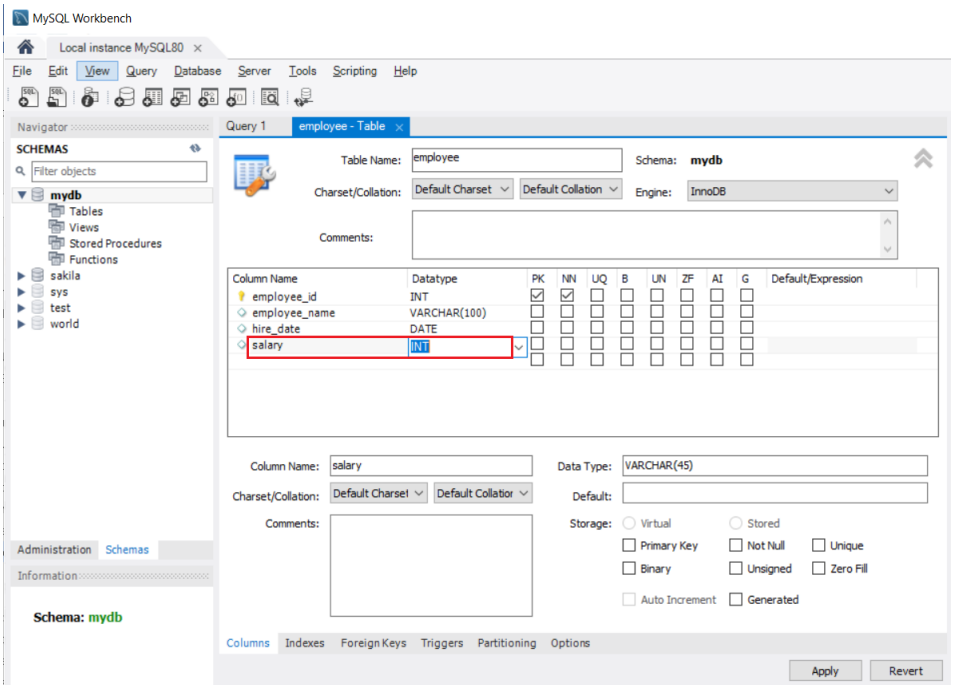
Select `employeecol` and update the **Column Name** to `employee_name` and **Datatype** to `VARCHAR(100)`.



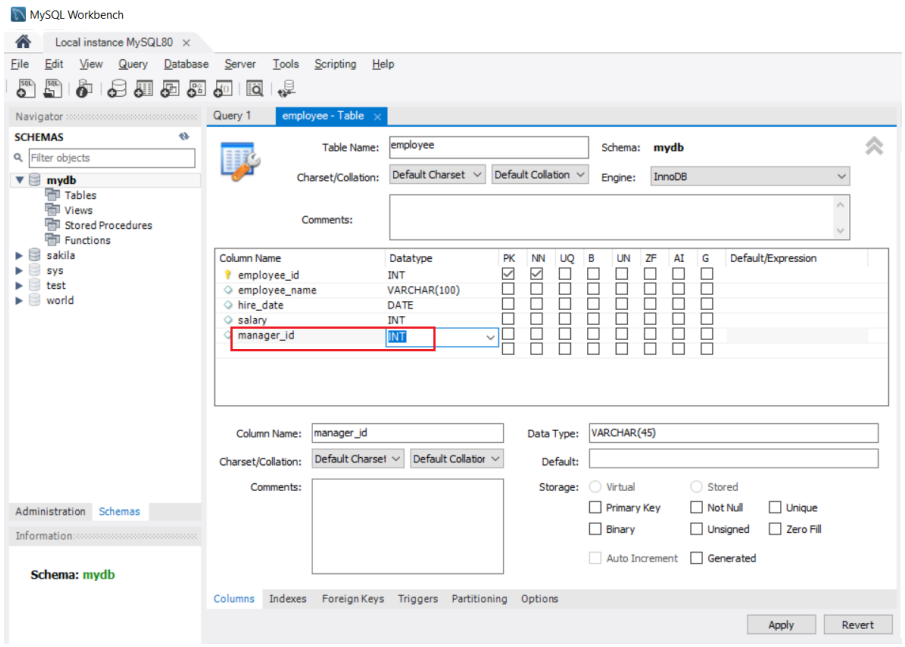
Then select another column and update the **Column Name** to `hire_date` and choose **Datatype** as `DATE` from the dropdown.



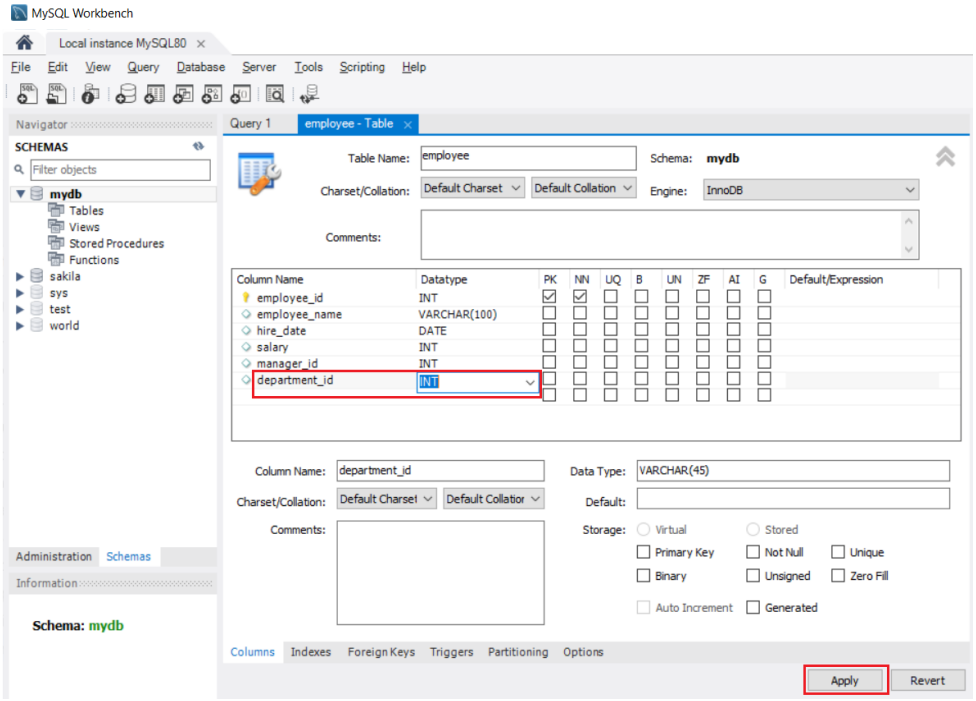
Then select another column and update the **Column Name** to `salary` and choose **Datatype** as `INT` from the dropdown.



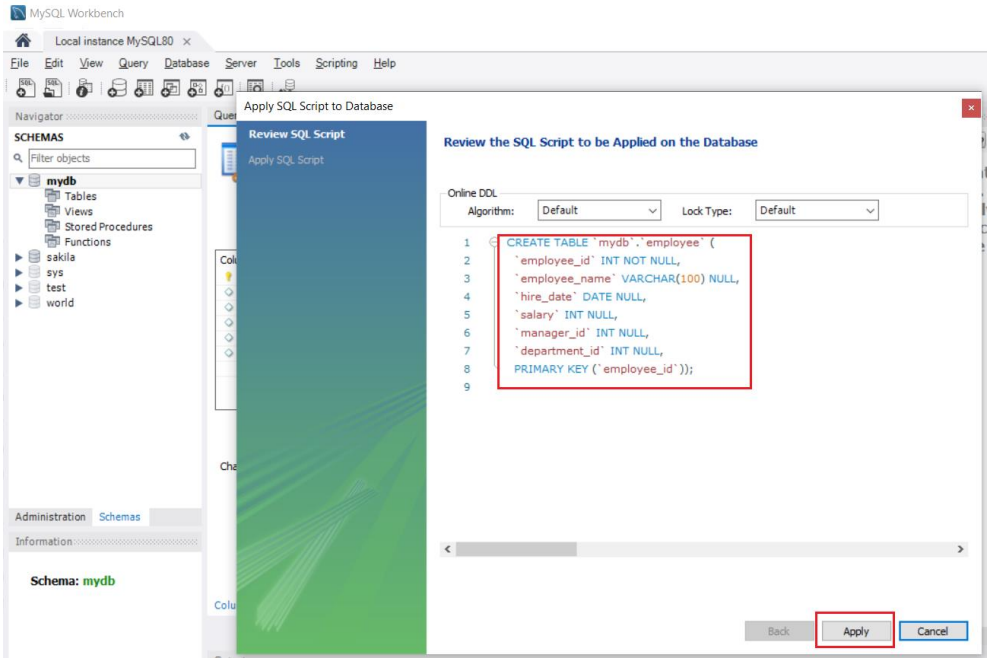
Then select another column and update the **Column Name** to `manager_id` and choose **Datatype** as `INT` from the dropdown.



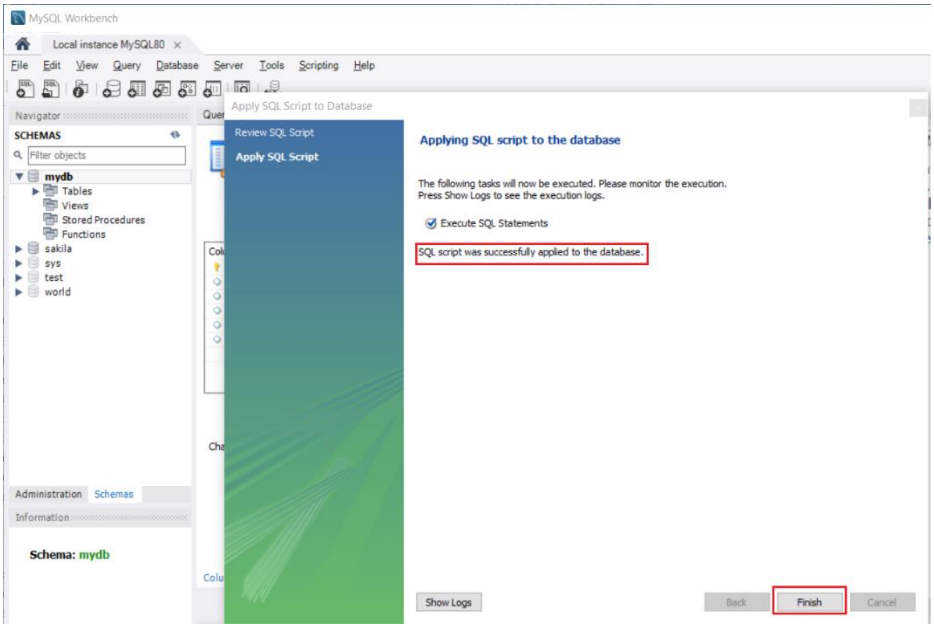
Then select another column and update the **Column Name** to `department_id` and choose **Datatype** as `INT` from the dropdown and click on **Apply** button.



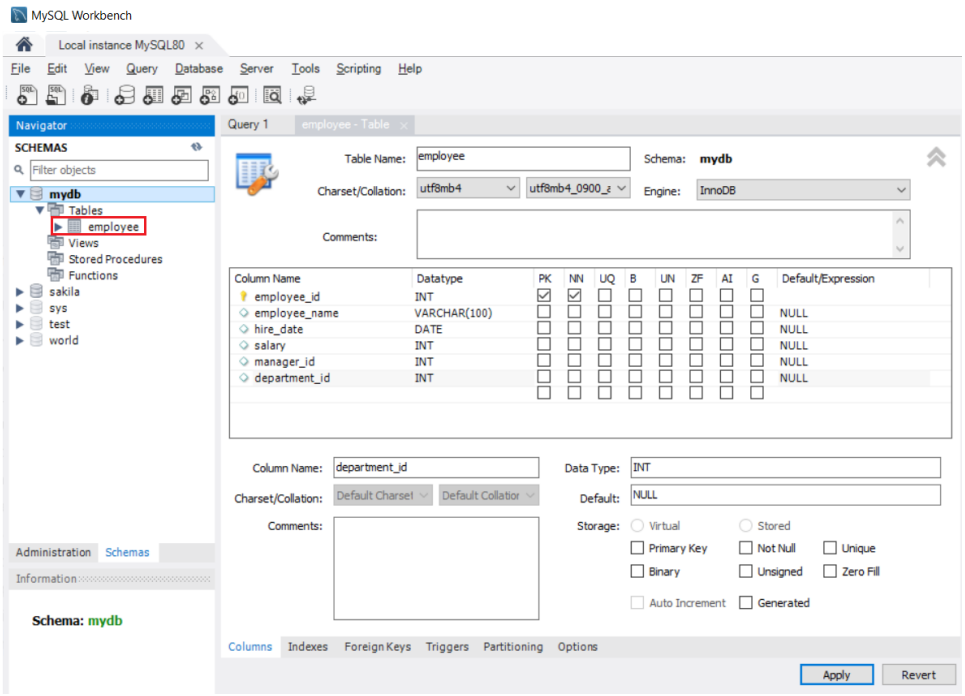
Review the SQL script provided and click on **Apply** button.



It executes the task and displays message that SQL script was successfully applied to the database. Click on **Finish** button to close.

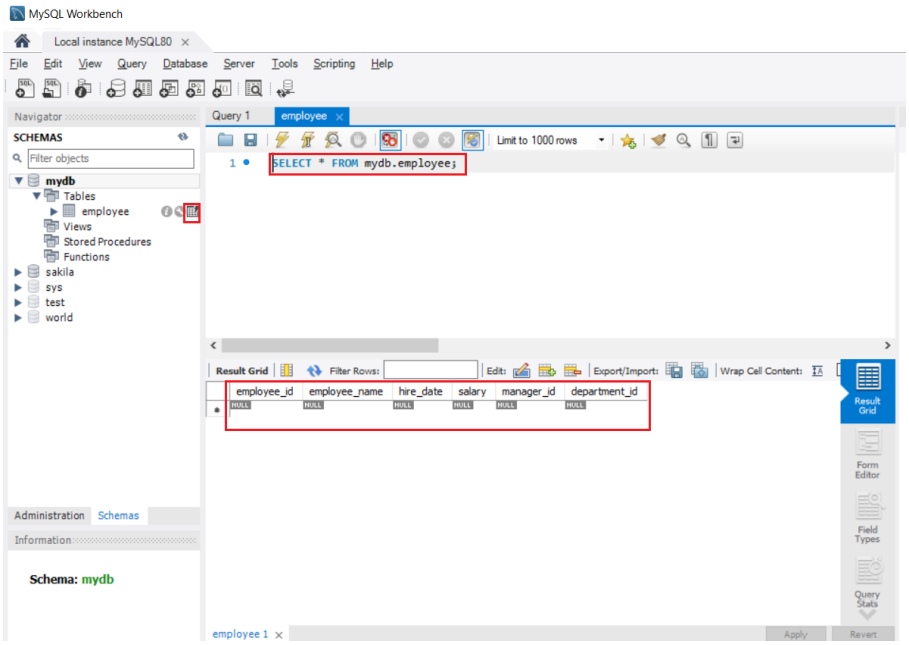


Now, expand **Tables** under `mydb` on the right side where you see `employee` table. Close **employee - Table** tab.

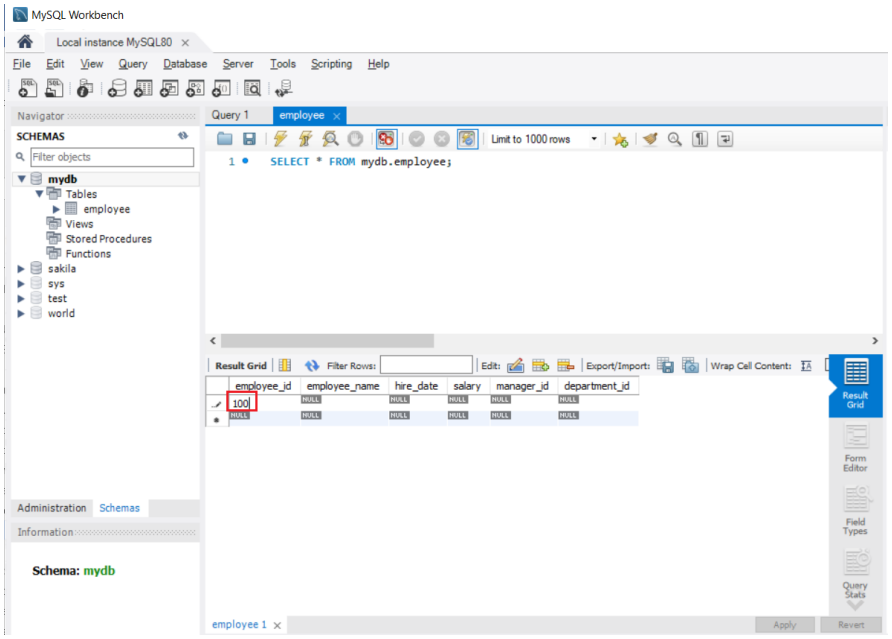


9.3. Insert Data:

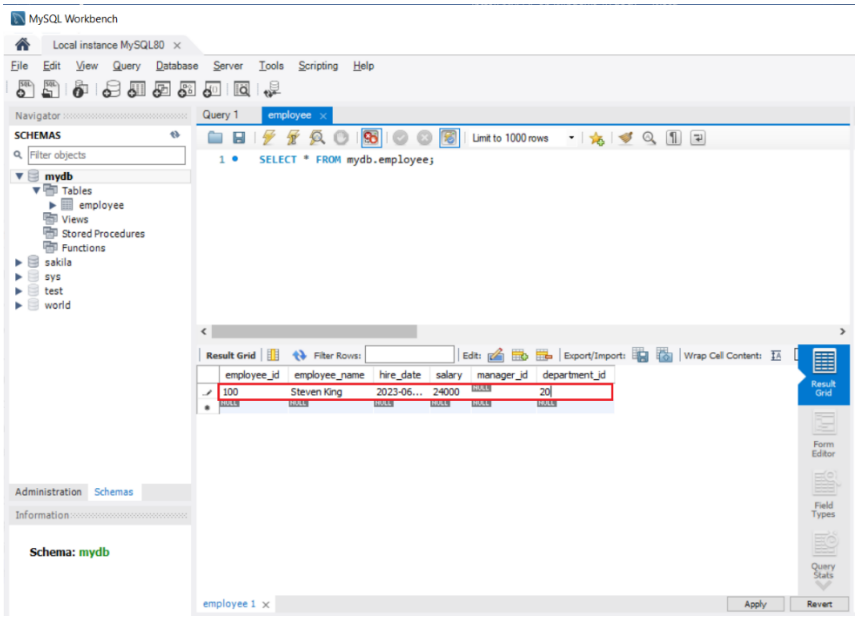
Now, select `employee` table and click on table icon which executes the select query and displays null data as there is no data available in the table.



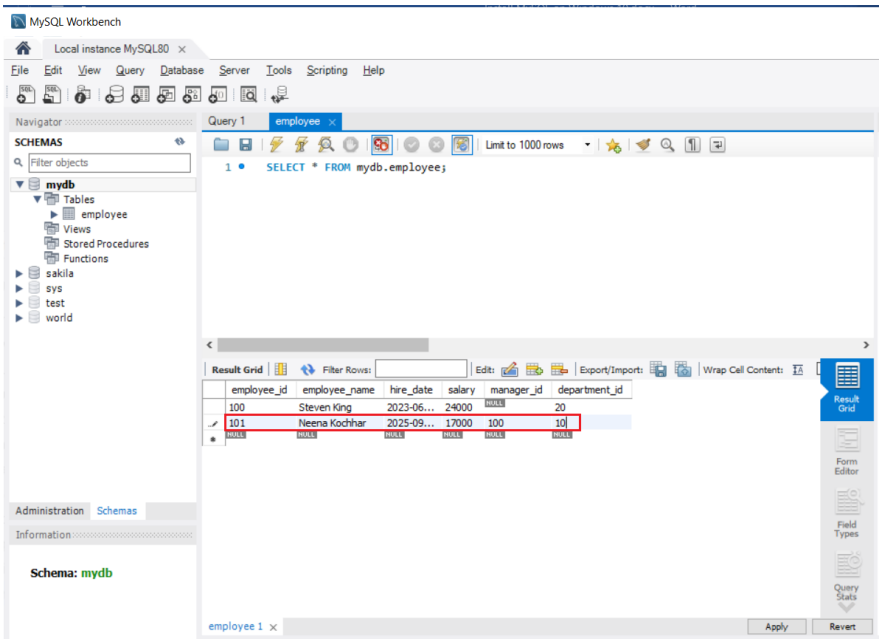
Click on `NULL` under `employee_id` column which displays blank data and update the value to `100`.



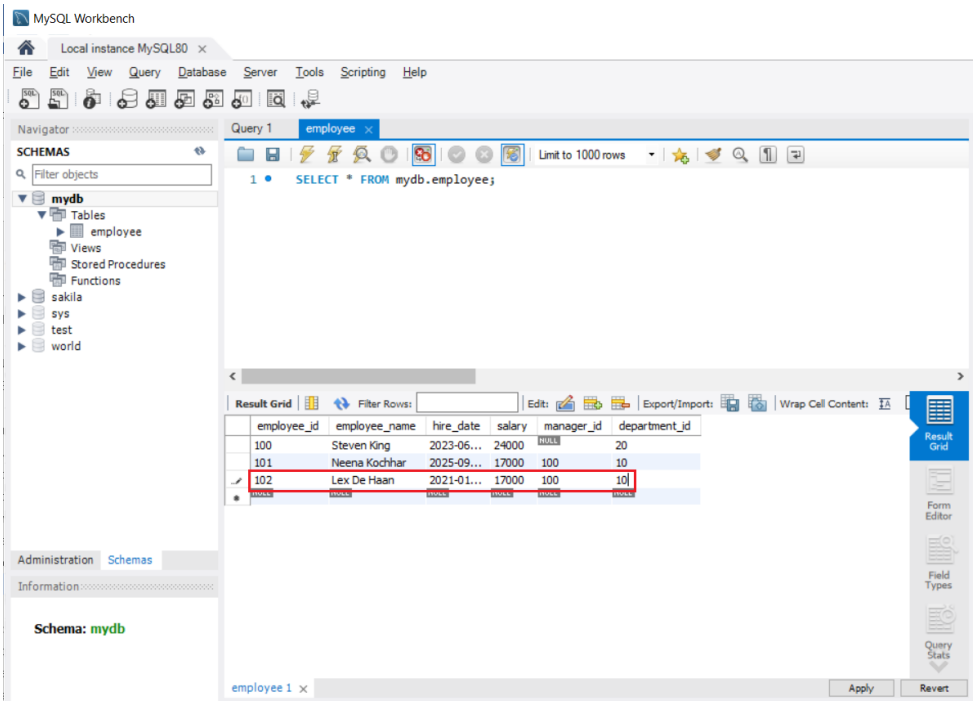
Similarly, select `NULL` under `employee_name` and update value to `Steven King`. Then select `NULL` under `hire_date` and update value to `2023-06-17`. Then select `NULL` under `salary` and update value to `24000`. Then select `NULL` under `department_id` and update value to `20`.



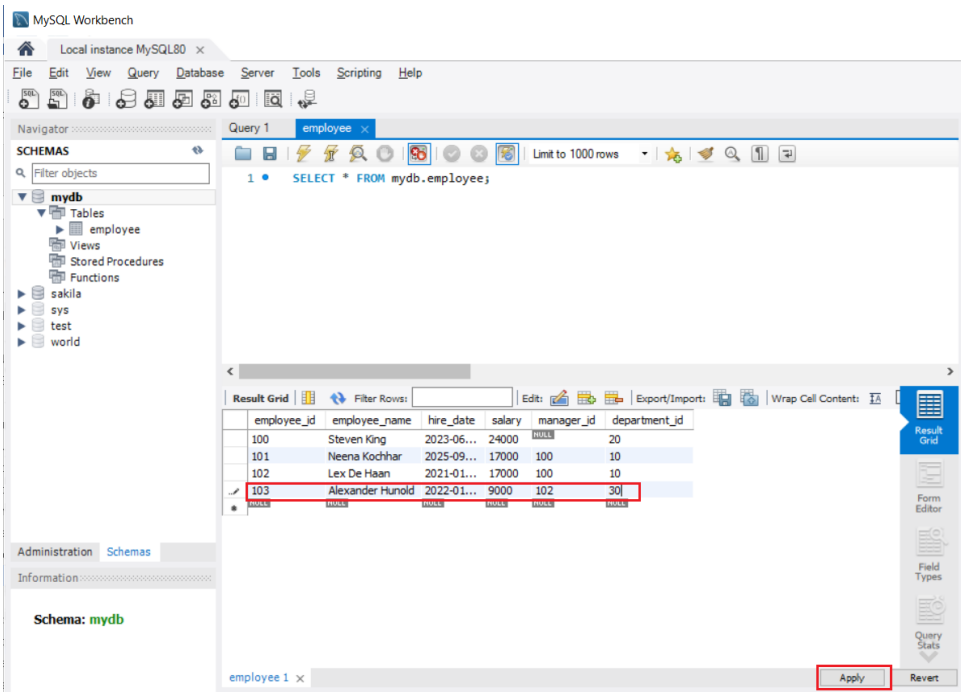
Similarly, select NULL value in the second row and enter the following data: `employee_id`: `101`, `employee_name`: `Neena Kochhar`, `hire_date`: `2025-09-21`, `salary`: `17000`, `manager_id`: `100`, `department_id`: `10`



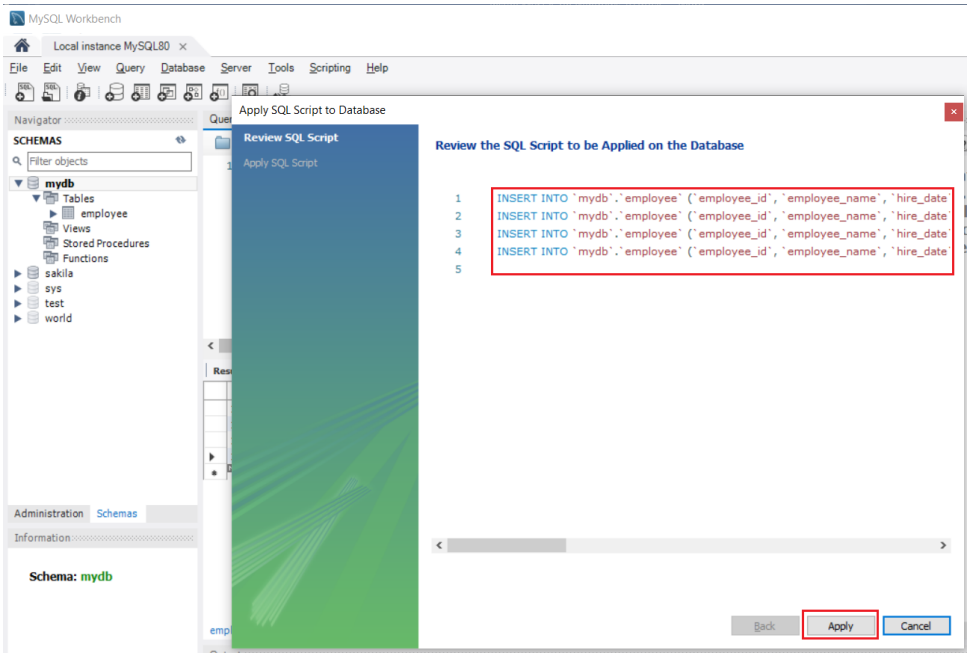
Similarly, select NULL value in the third row and enter the following data:
`employee_id`: `102`, `employee_name`: `Lex De Haan`, `hire_date`: `2021-01-13`, `salary`: `17000`, `manager_id`: `100`, `department_id`: `10`



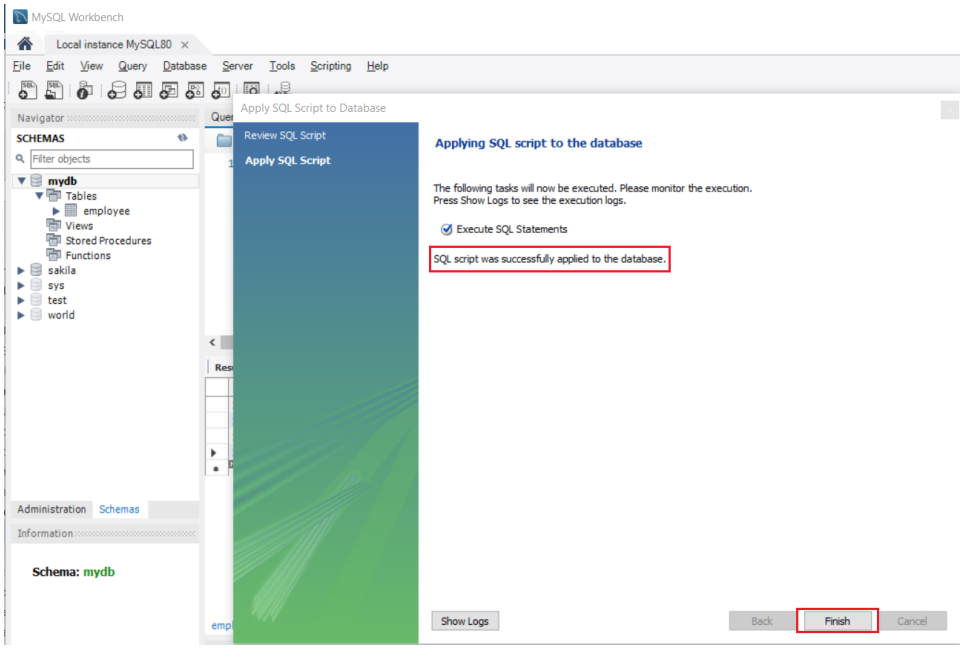
Similarly, select NULL value in the fourth row and enter the following data. Then click on **Apply** button.
`employee_id`: `103`, `employee_name`: `Alexander Hunold`, `hire_date`: `2022-01-03`, `salary`: `9000`, `manager_id`: `102`, `department_id`: `30`



Review the SQL script provided and click on **Apply** button.

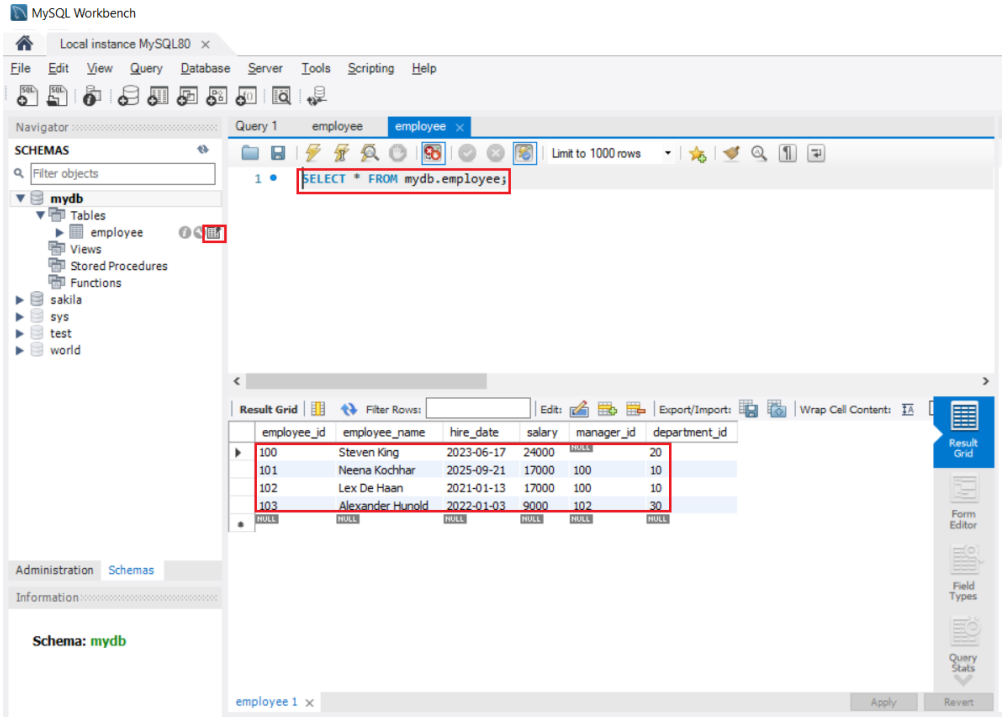


It executes the task and displays message that SQL script was successfully applied to the database. Click on **Finish** button to close.



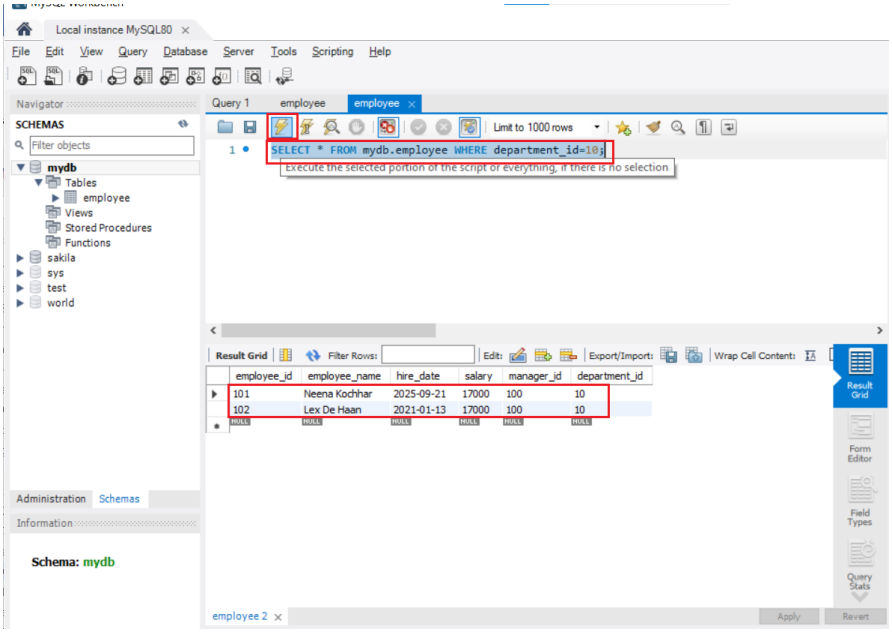
9.4. Select Data:

To select data from a table, click on table icon next to `employee` table which executes the select query and displays all records available in the table.



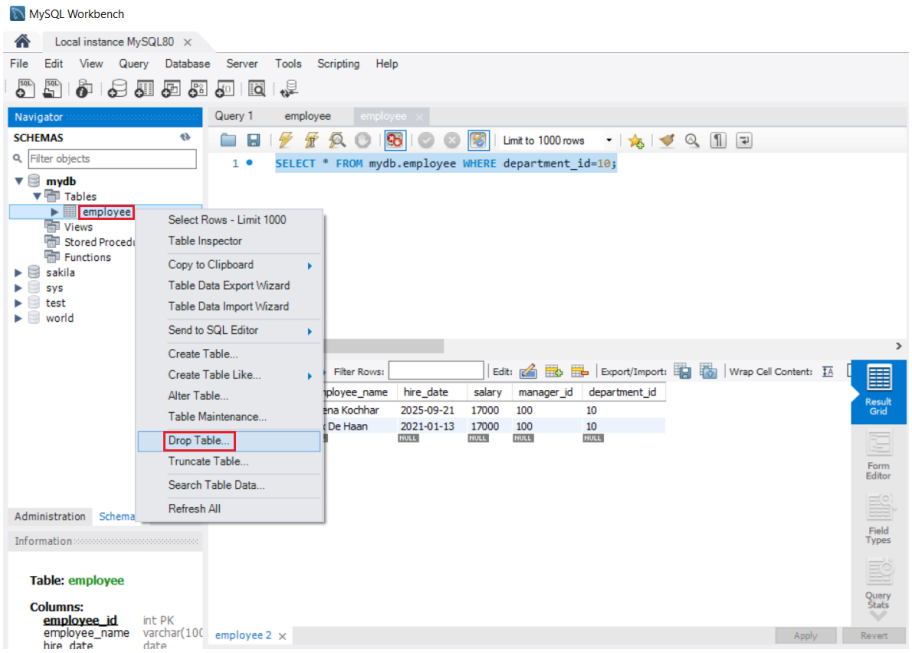
We can modify the select query to filter records and displays data.
Write the following query to display department `10` employees, select the query and click on **Execute** button to execute query and display results:

```
SELECT * FROM mydb.employee WHERE department_id=10;
```

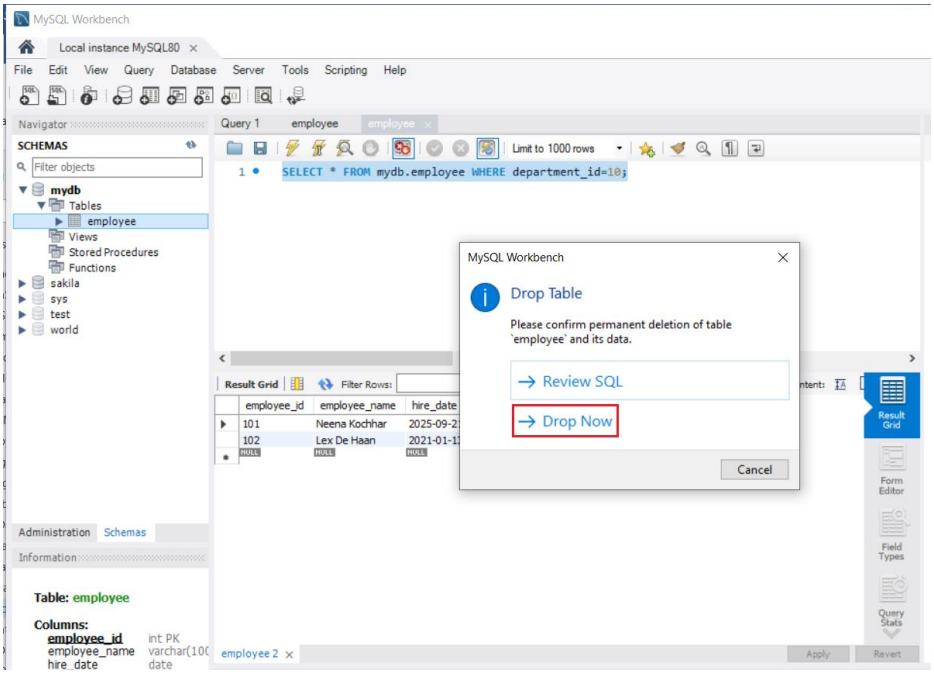


9.5. Drop Table:

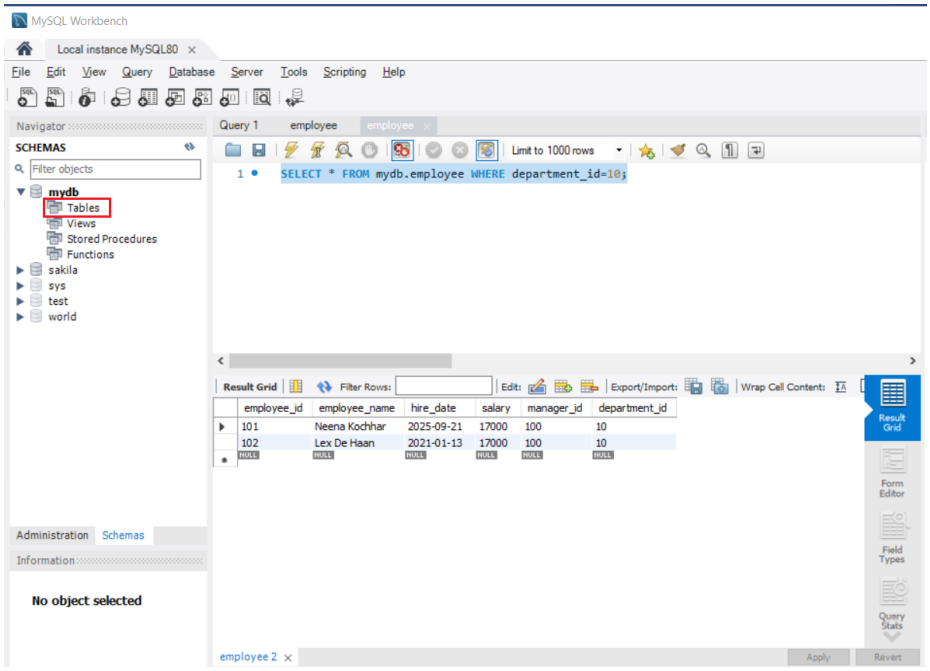
To drop a specific table, right click on table name and select **Drop Table** option.



It opens a popup message where you can choose **Review SQL** if you want to review before dropping. Otherwise, select **Drop Now** to drop the table and its data permanently.

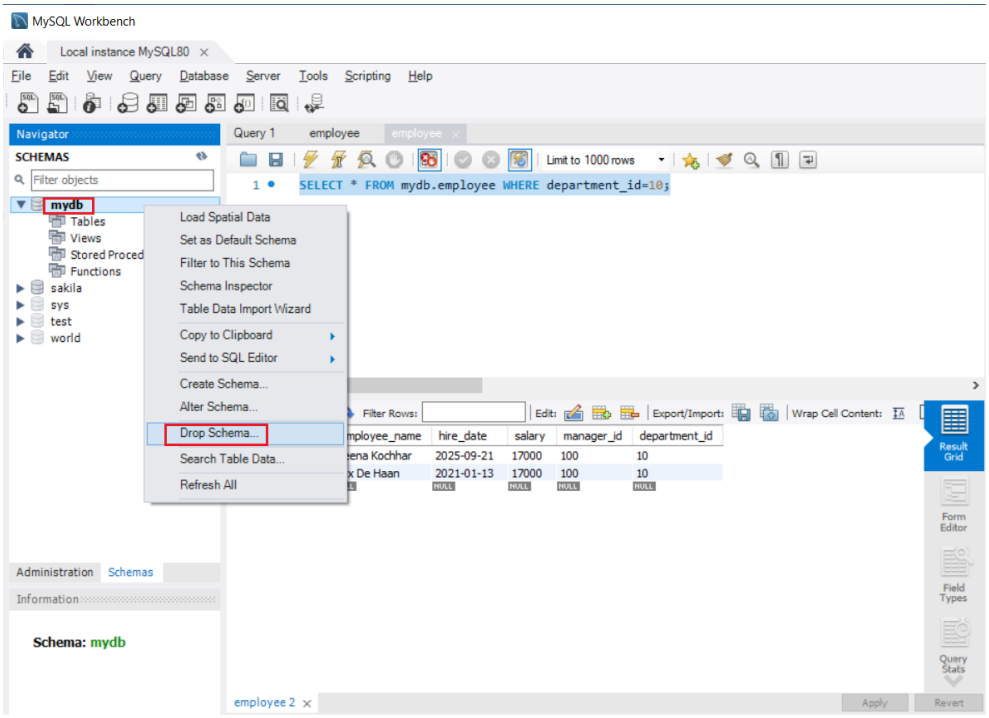


You can observe that `employee` table has disappeared under **Tables** section in the `mydb` database.

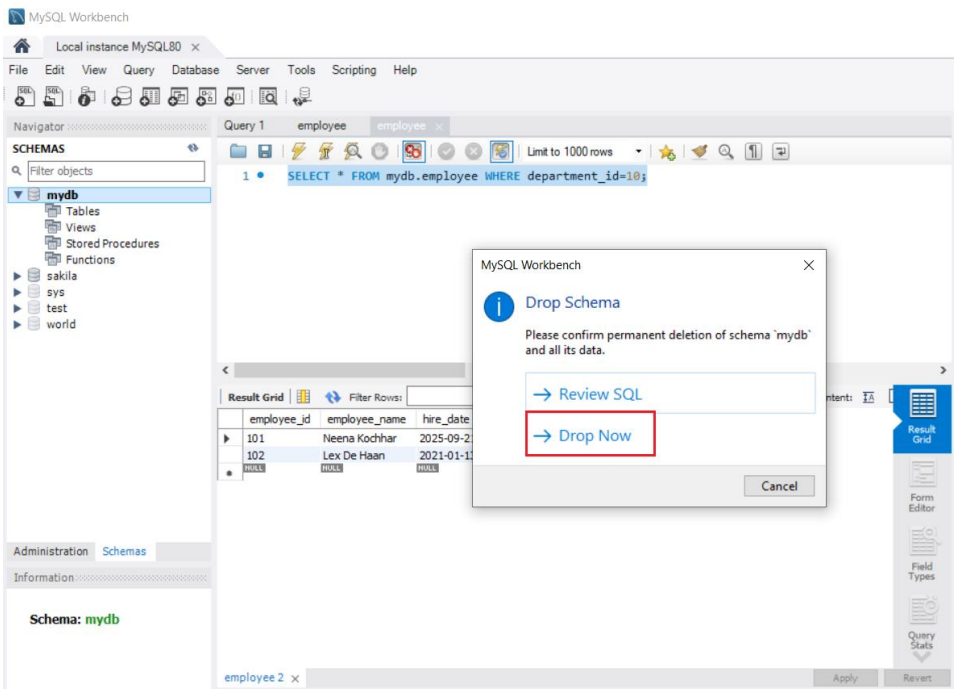


9.6. Drop Database:

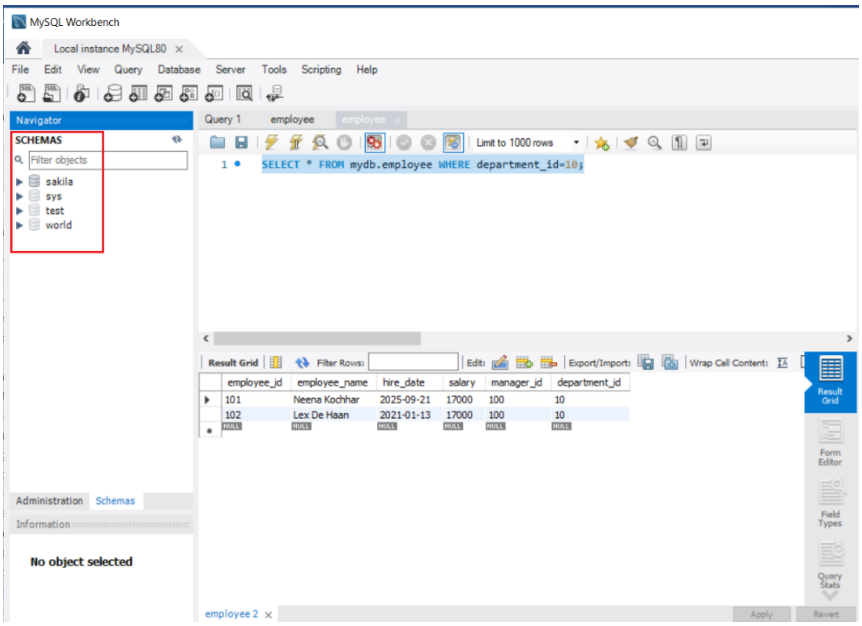
To drop a specific database, right click on database name and select **Drop Schema** option.



It opens a popup message where you can choose **Review SQL** if you want to review before dropping. Otherwise, select **Drop Now** to drop the schema and its entire contents permanently.

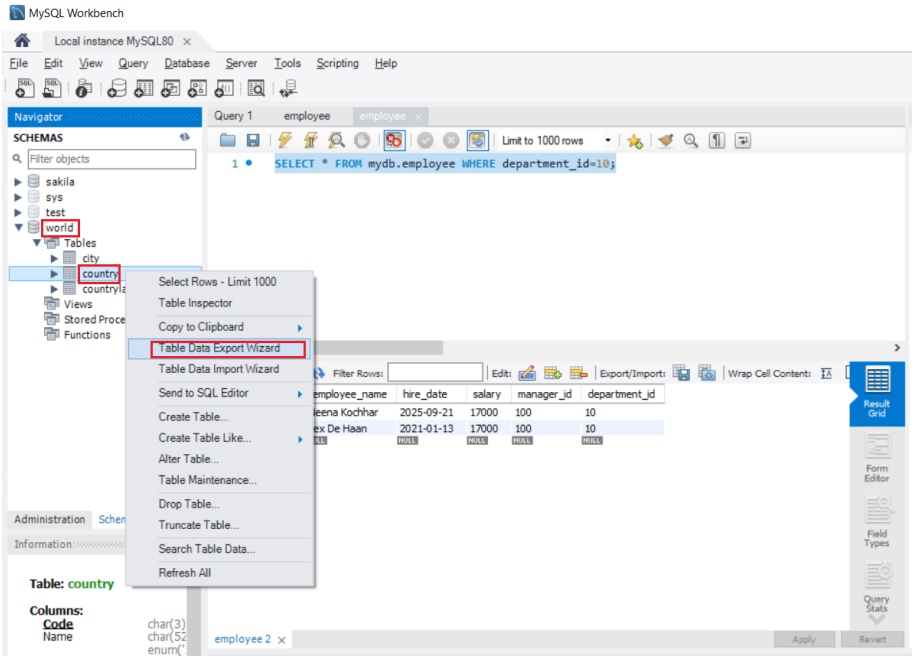


You can observe that `mydb` table has disappeared under **Schemas** section.

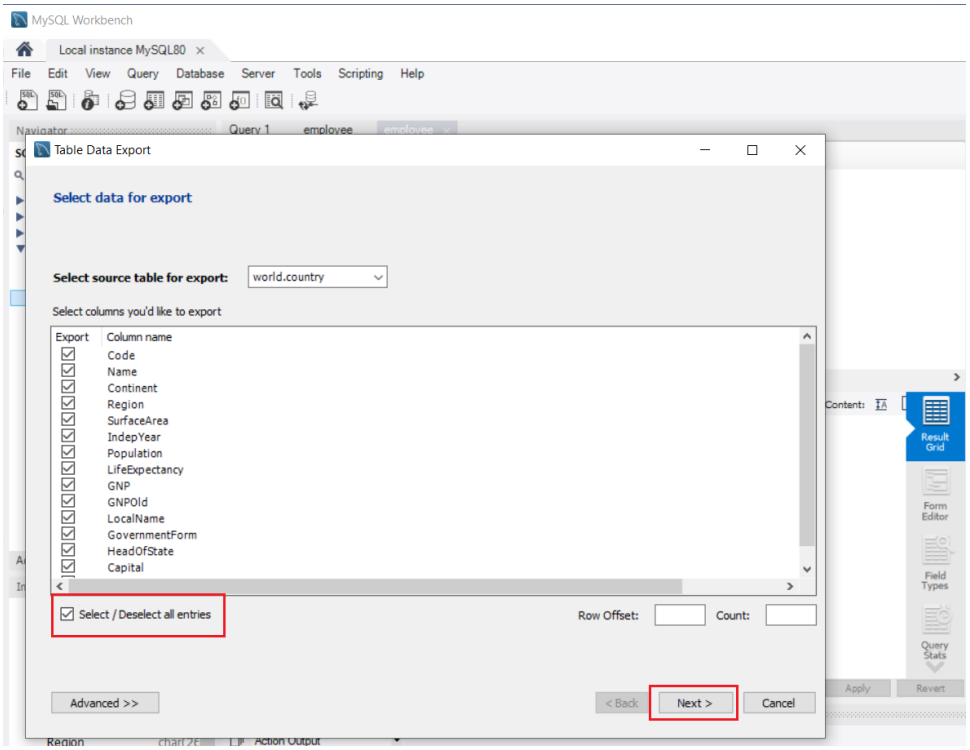


9.7. Export Table:

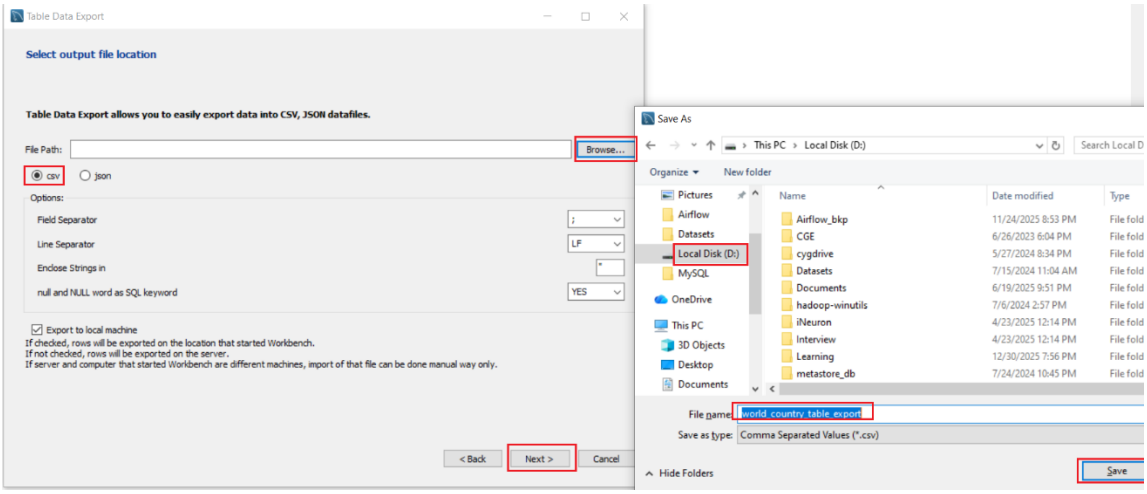
To export a specific table, such as `country` table under `world` schema, right click on the table name and select **Table Data Export Wizard** option.



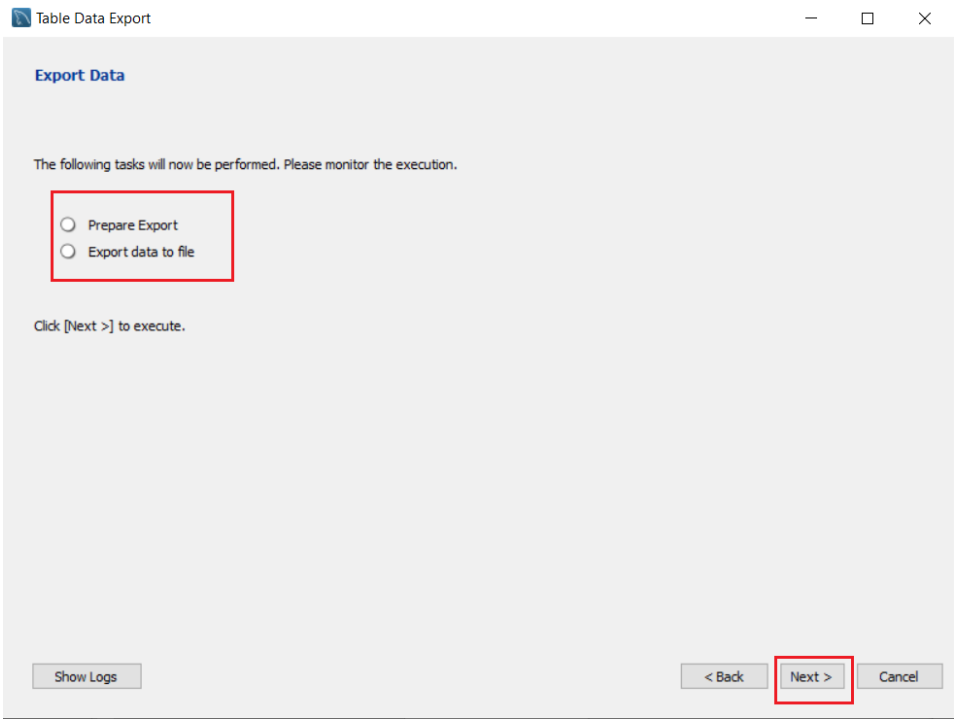
Keep all columns selected (or select required columns for export) and click on **Next** button.



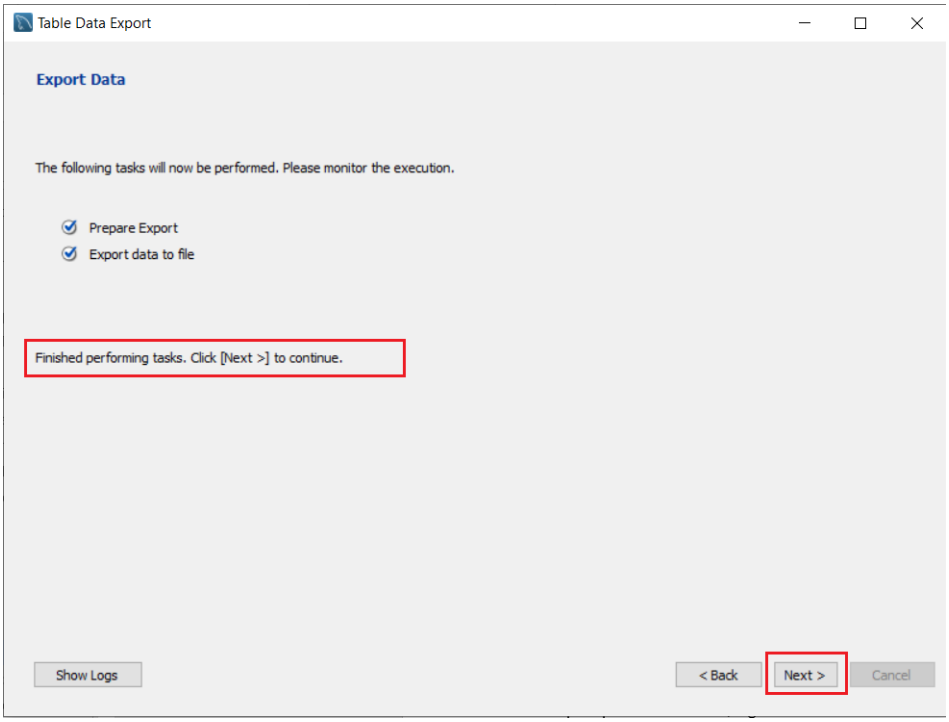
Select the file format (CSV or JSON) and click on **Browse** button to select the desired directory and provide the file name and click on **Save**. Then click on **Next** to continue.



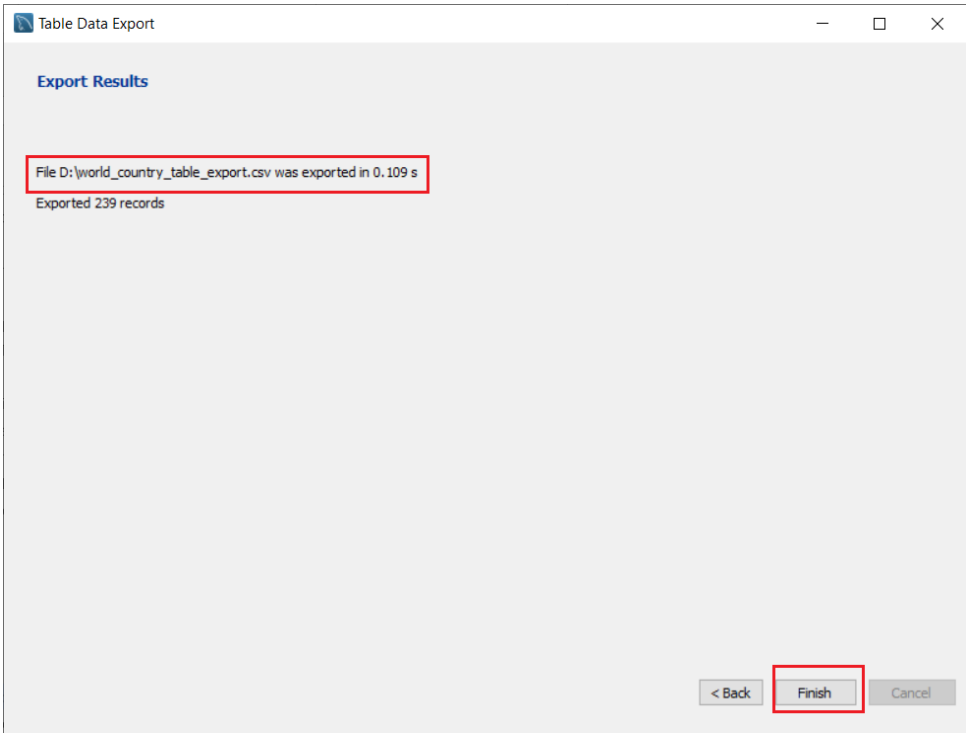
It displays the list of tasks to be performed. Click on **Next** to continue.



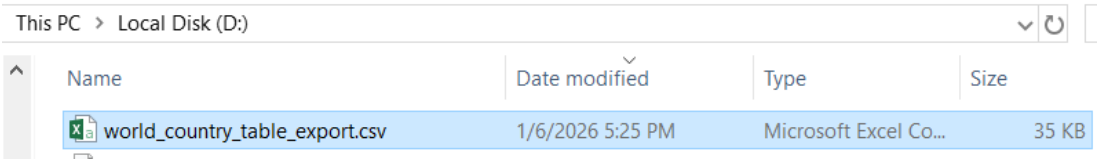
In few seconds, data export would be finished. Click **Next** to continue.



It displays the file location where the data got exported successfully. Click on **Finish** to exit the wizard.

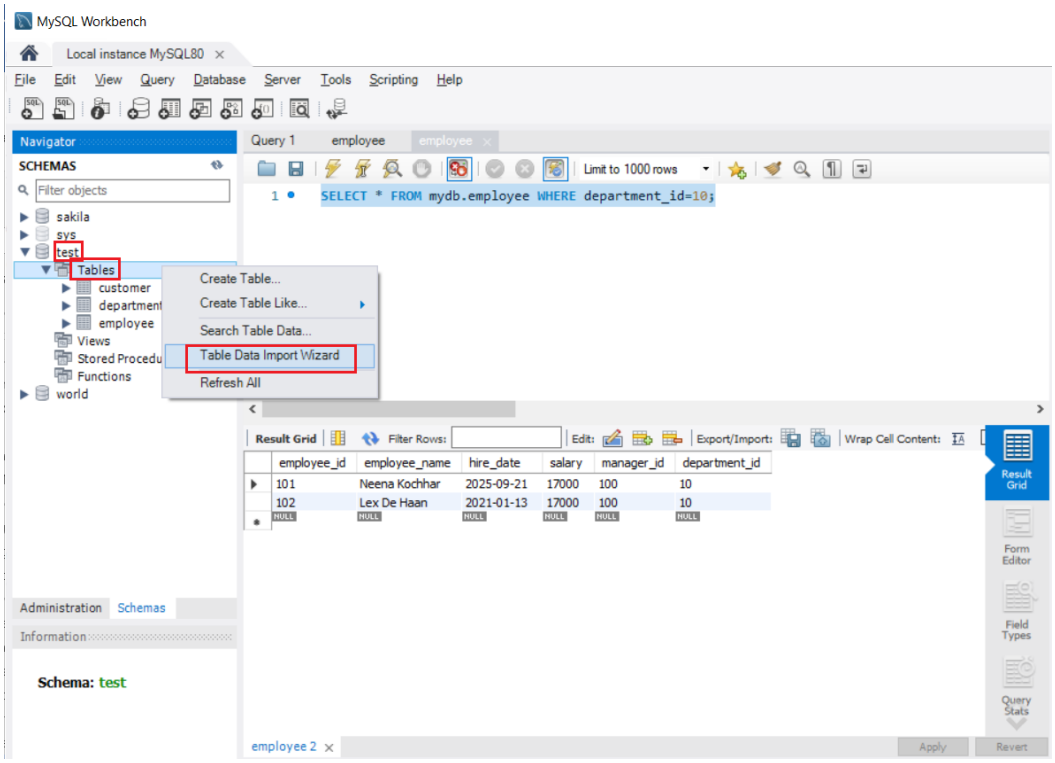


Once finished, navigate to the directory and review the CSV file created.

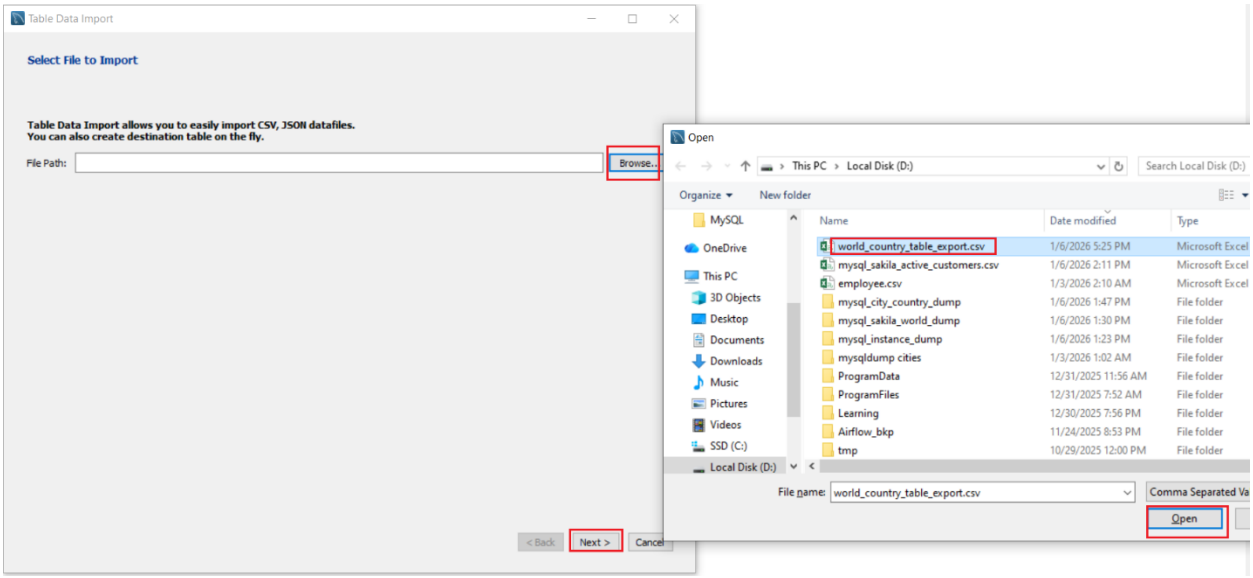


9.8. Import Table:

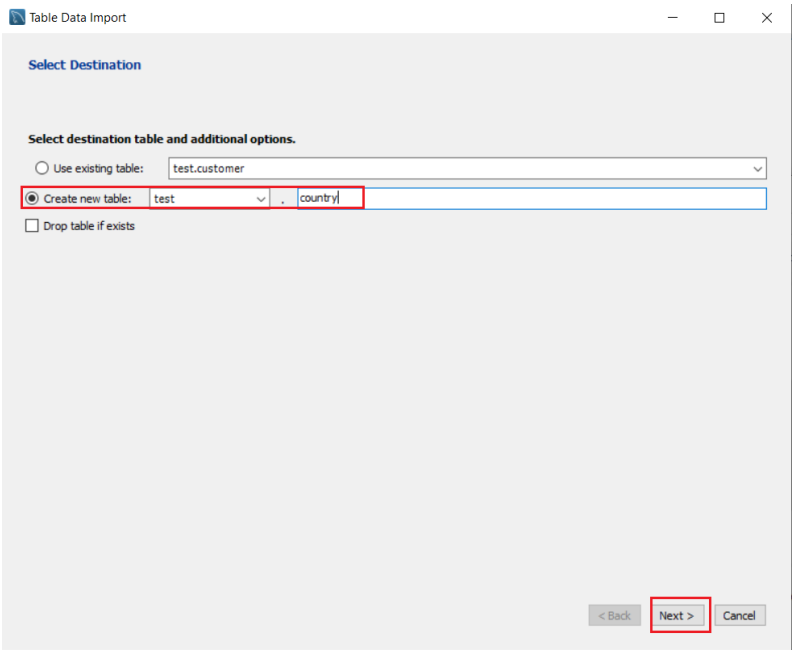
To import data right click on the **Tables** folder under specific schema such as `test` and select **Table Data Export Wizard** option.



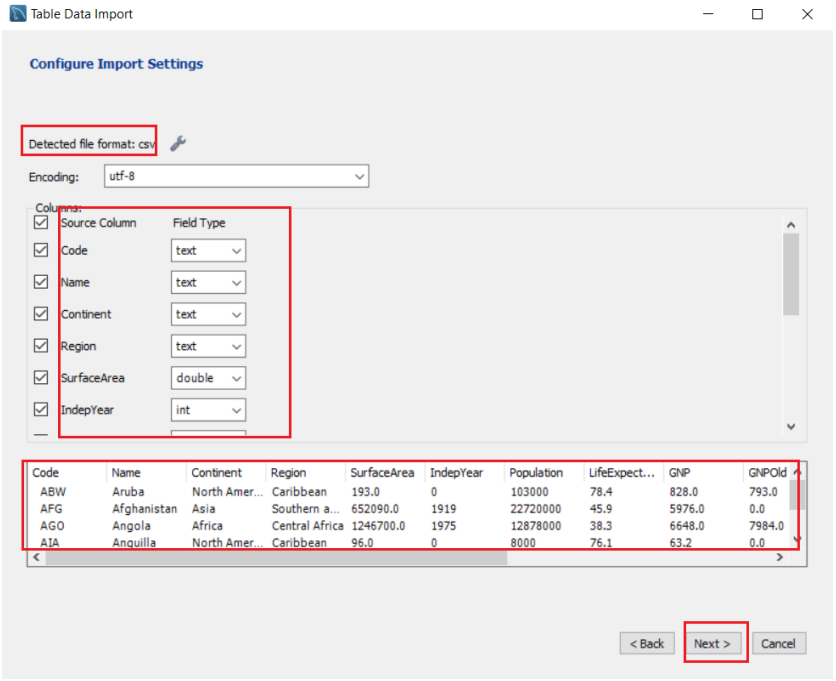
Click on **Browse** button and navigate to the directory and select the file to be imported and click **Open**. Then click on **Next** to continue.



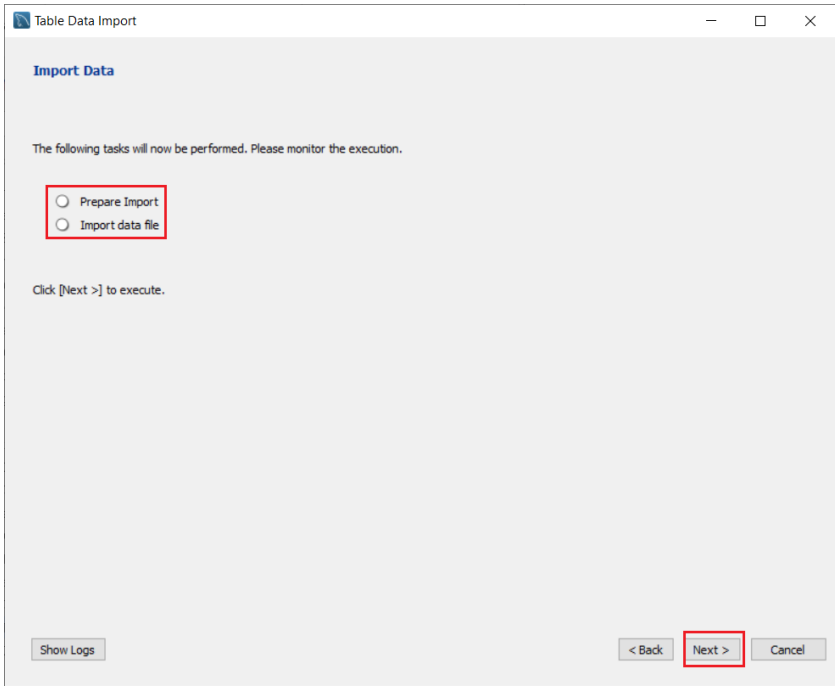
By default, it tries to create new table with the file name. Provide the respective table name and click on **Next** to continue.



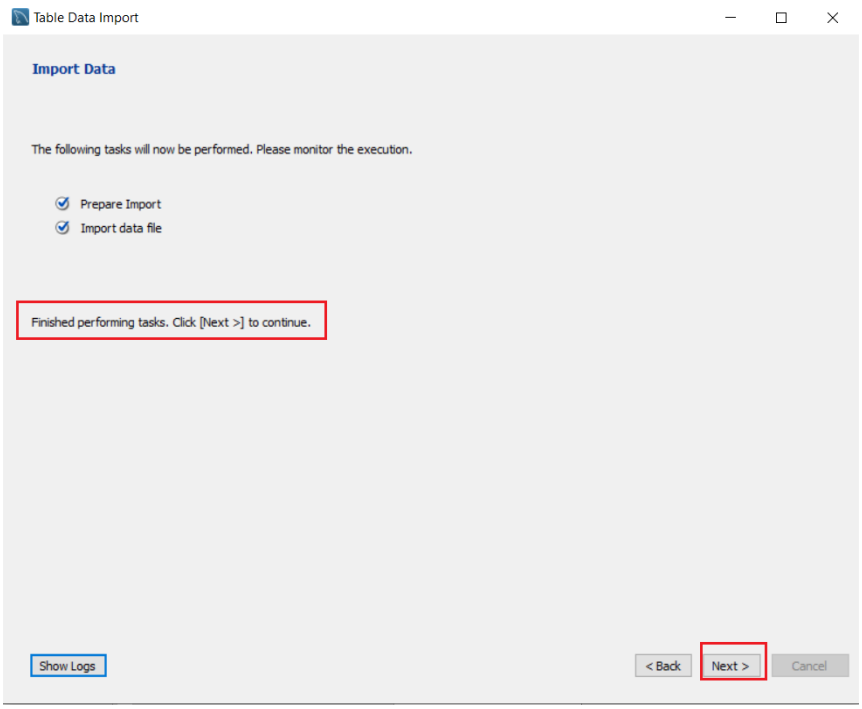
It automatically detects the file format and displays the list of columns with auto-detected datatypes and sample data. Review data types based on sample data and click on **Next** to continue.



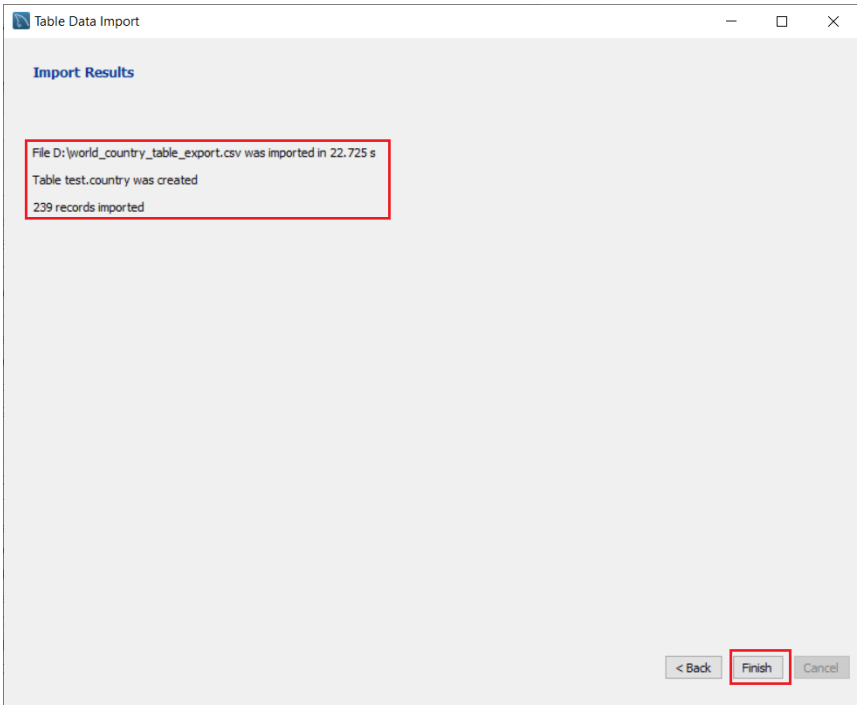
It displays the list of tasks to be performed. Click on **Next** to continue.



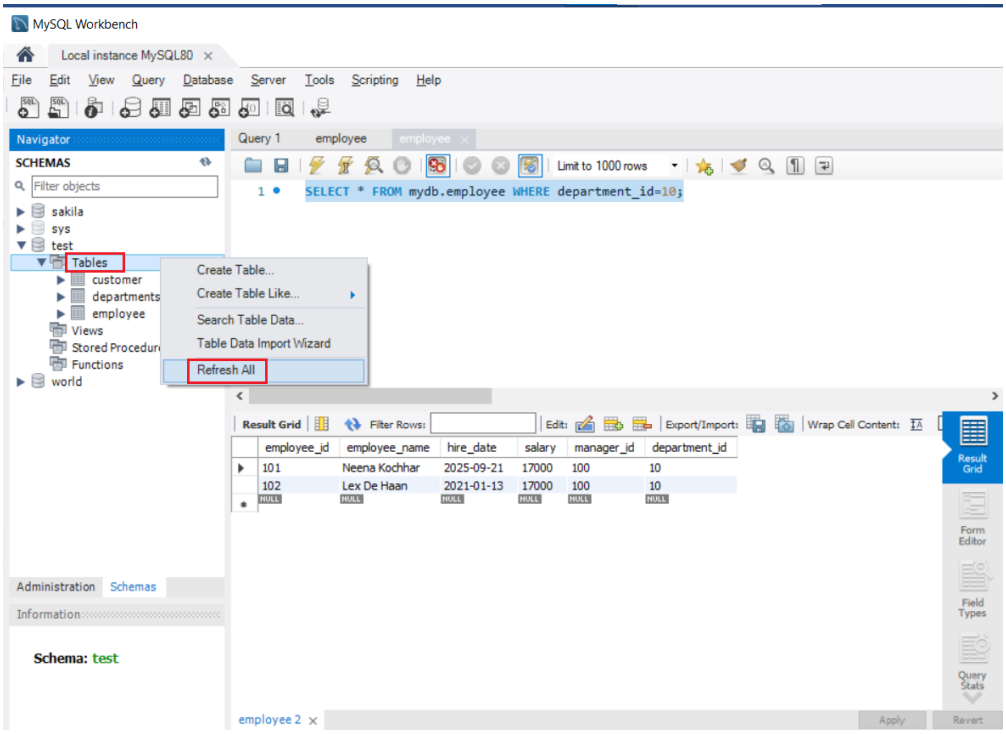
In few seconds, data import would be finished. Click **Next** to continue.



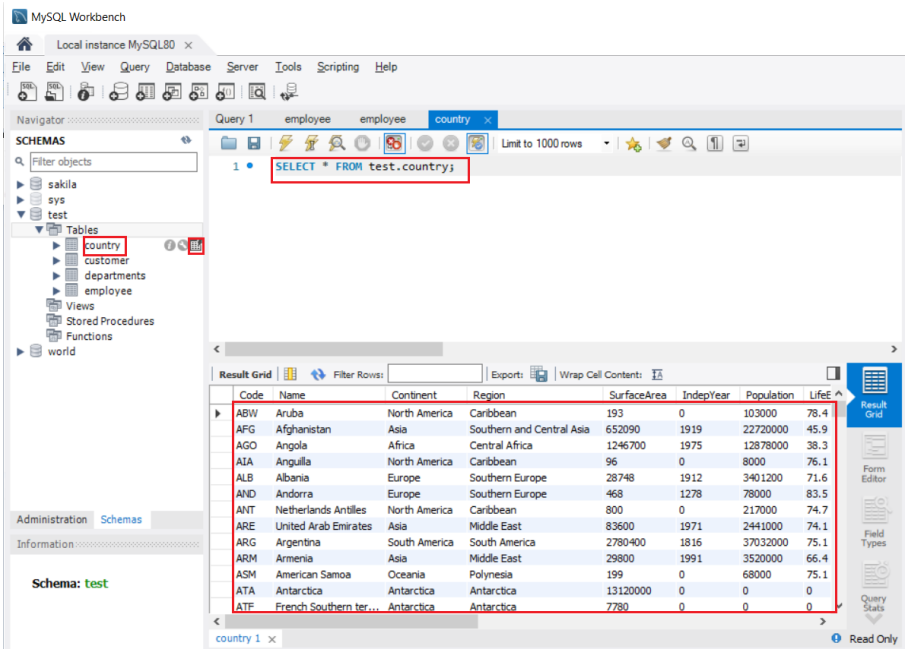
It displays the number of records imported to a table from the file. Click on **Finish** to exit the wizard.



Once finished, right click on **Tables** folder under the schema where table got imported and select **Refresh All** option.

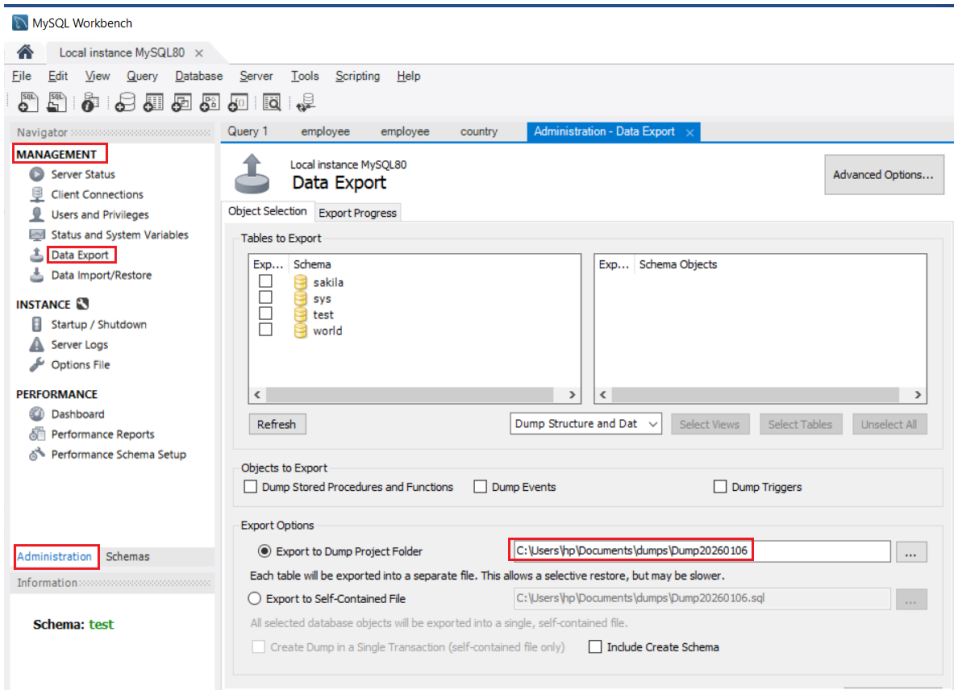


You can see the `country` table created. Click on table icon next to `country` table which executes the select query and displays all records available in the table.

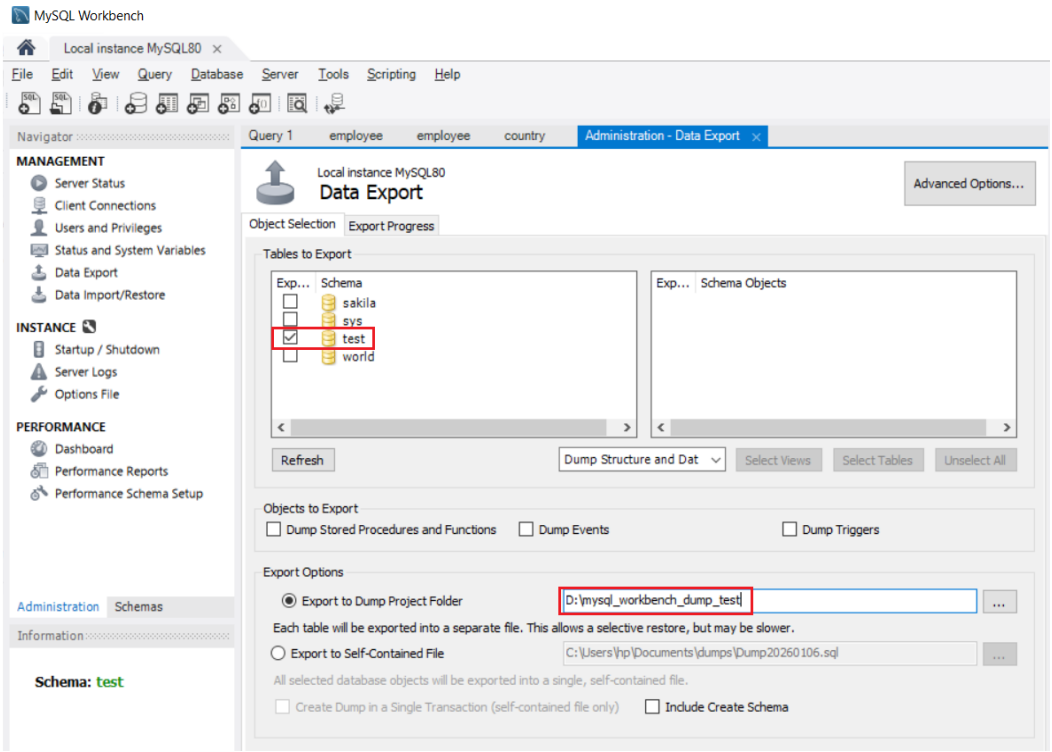


9.9. Export Database:

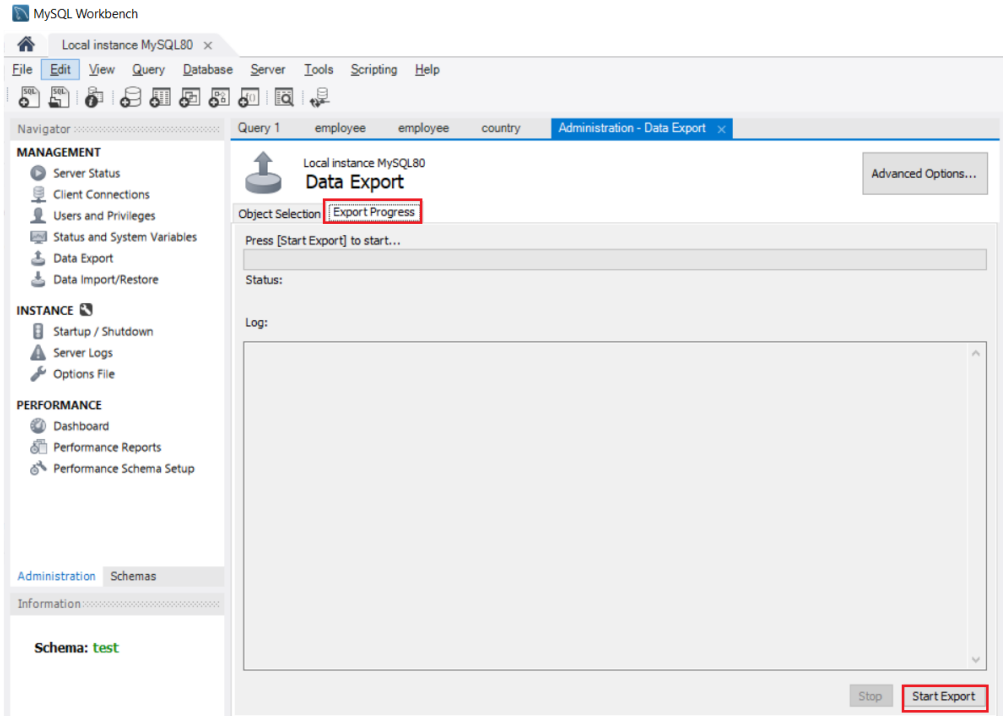
To export a specific database, click on **Administration** tab on the right side and select **Data Export** under **Management** which opens up **Administration-Data Export** tab with list of schemas and the default dump directory.



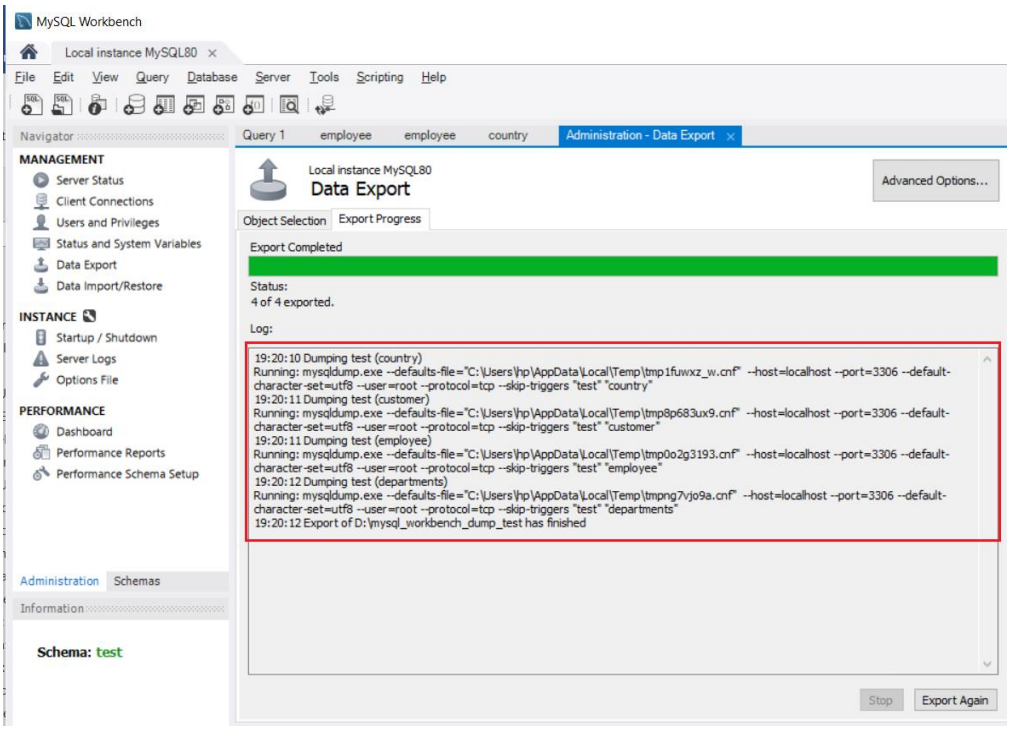
Select the database/schema name such as `test` and provide the appropriate location for data dump.



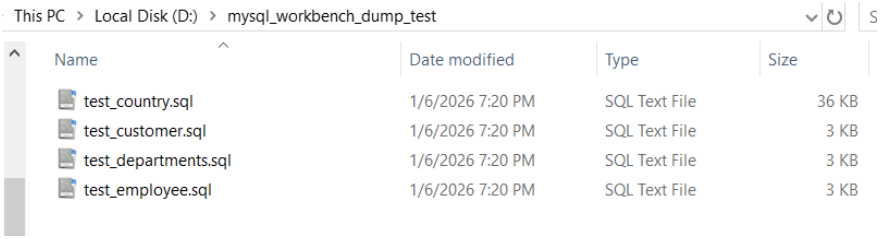
Select **Export Progress** tab and click on **Start Export** to begin with the data export.



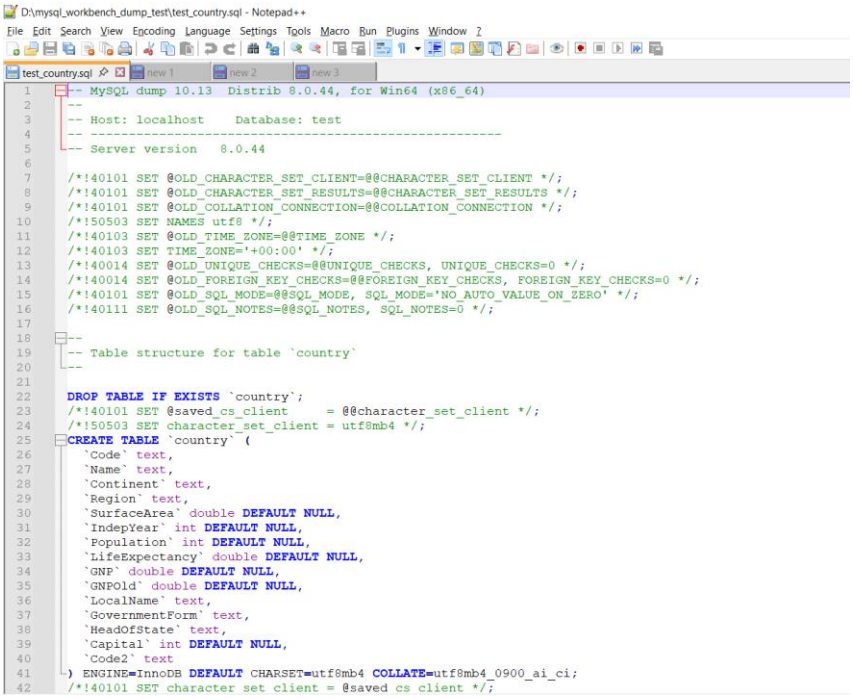
In few seconds, the data export would be finished. Close **Administration-Data Export** tab.



Once finished, go to the location where database dump created and you can see SQL files created

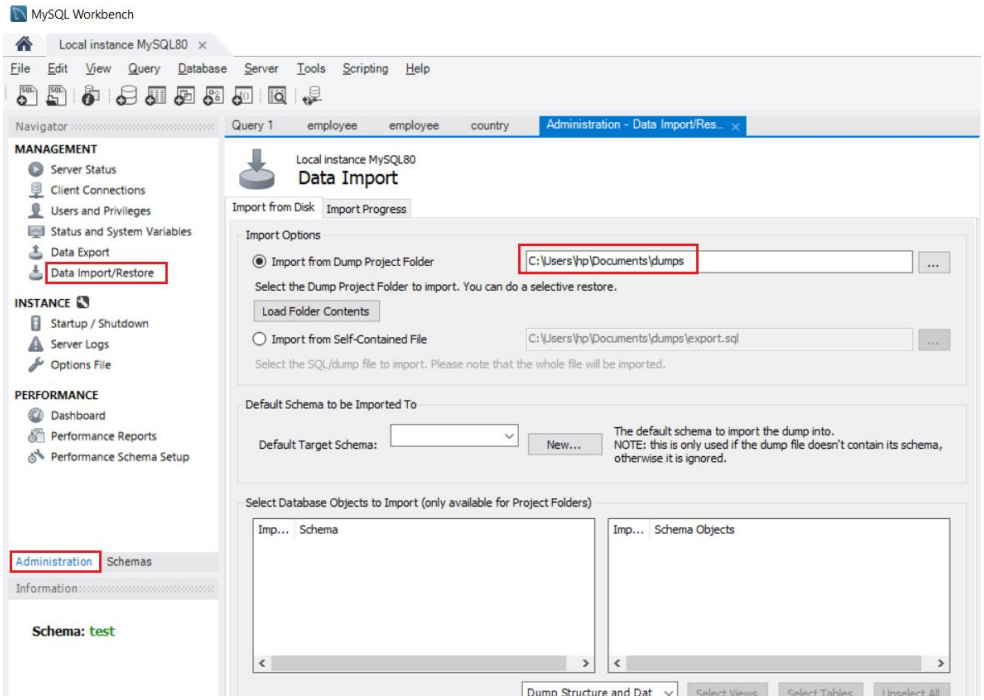


Review the SQL file which contains create and insert statements for the respective table to be created in the target database.

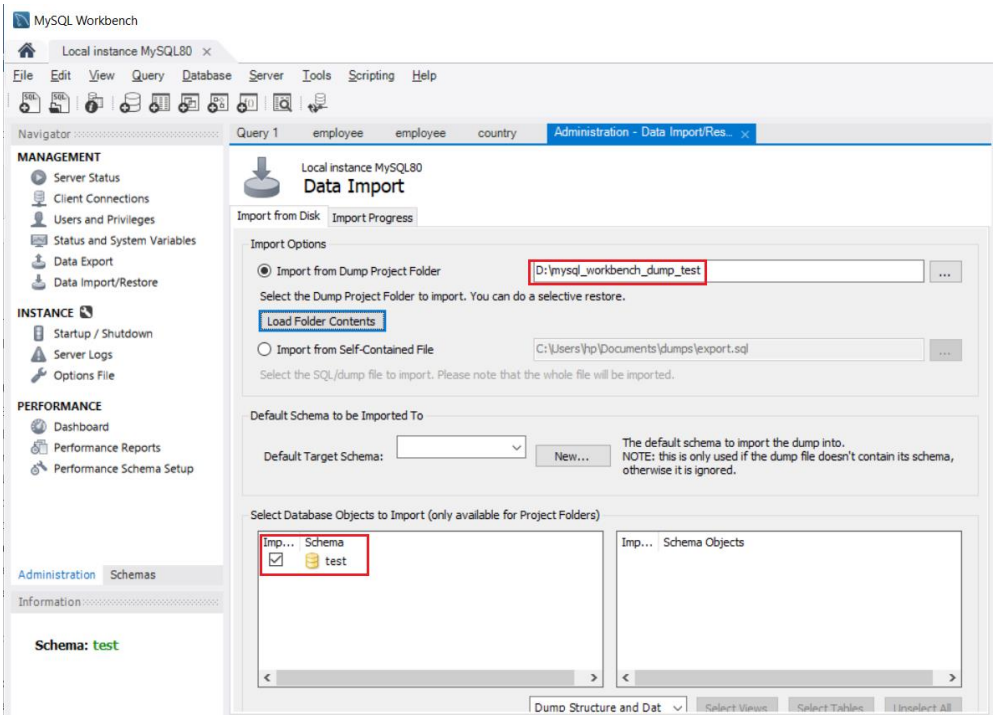


9.10. Import Database:

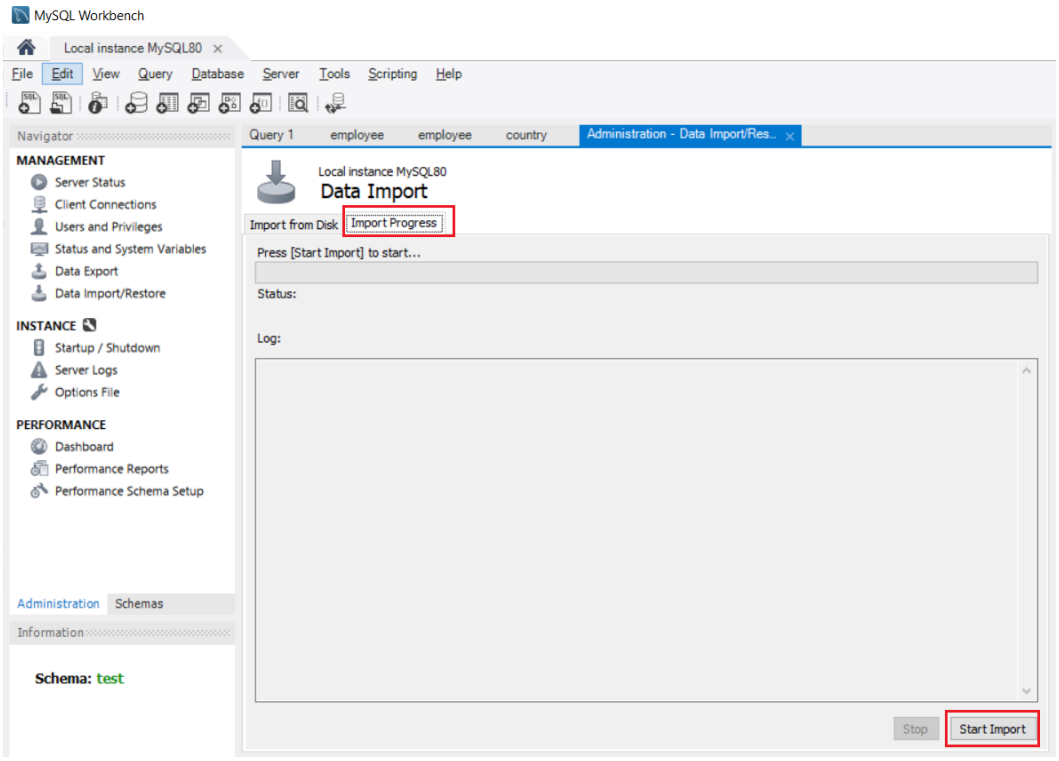
To import a specific database, click on **Administration** tab on the right side and select **Data Import/Restore** under **Management** which opens up **Administration-Data Import** tab with the default dump directory.



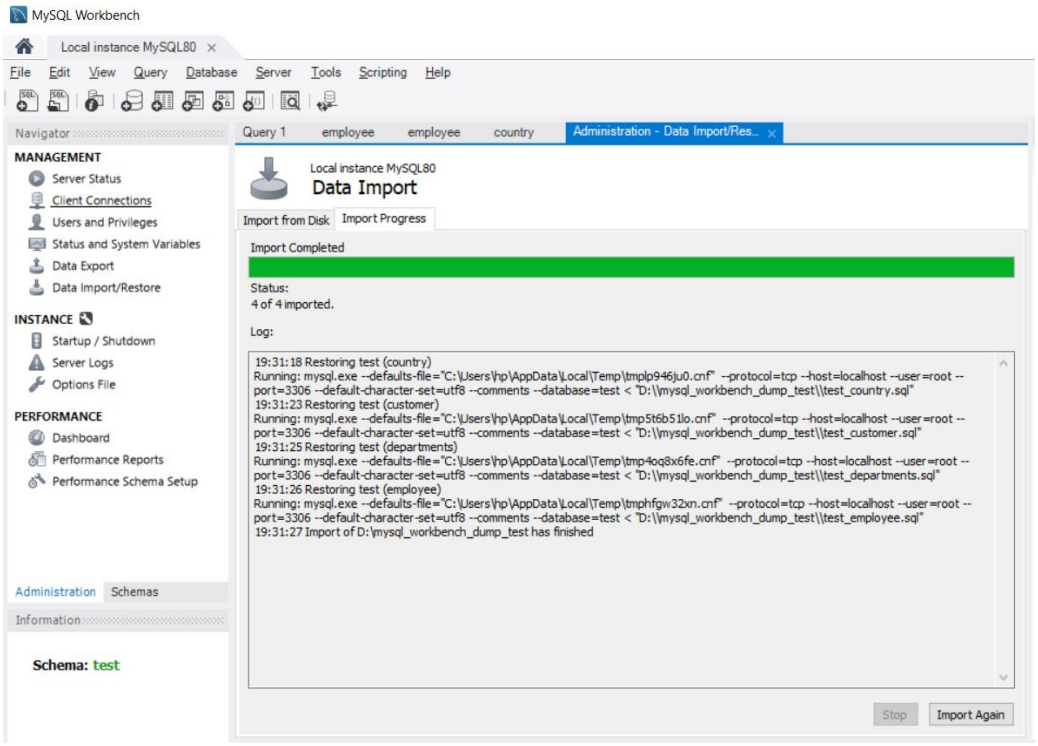
Provide the appropriate location where the database dump exists. Then it displays the list of databases to be imported



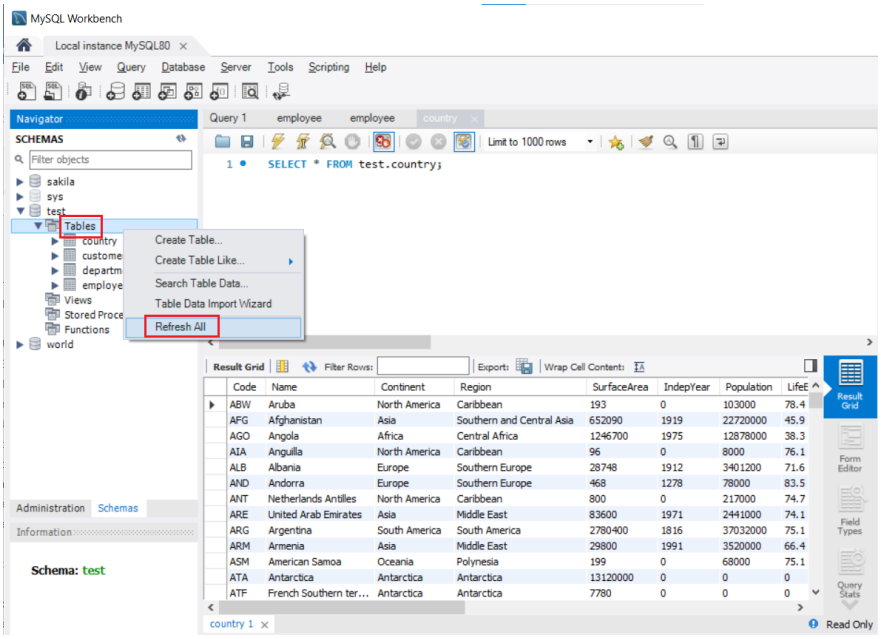
Select **Import Progress** tab and click on **Start Import** to begin with the database import.



In few seconds, the data import would be finished. Close **Administration-Data Import** tab.

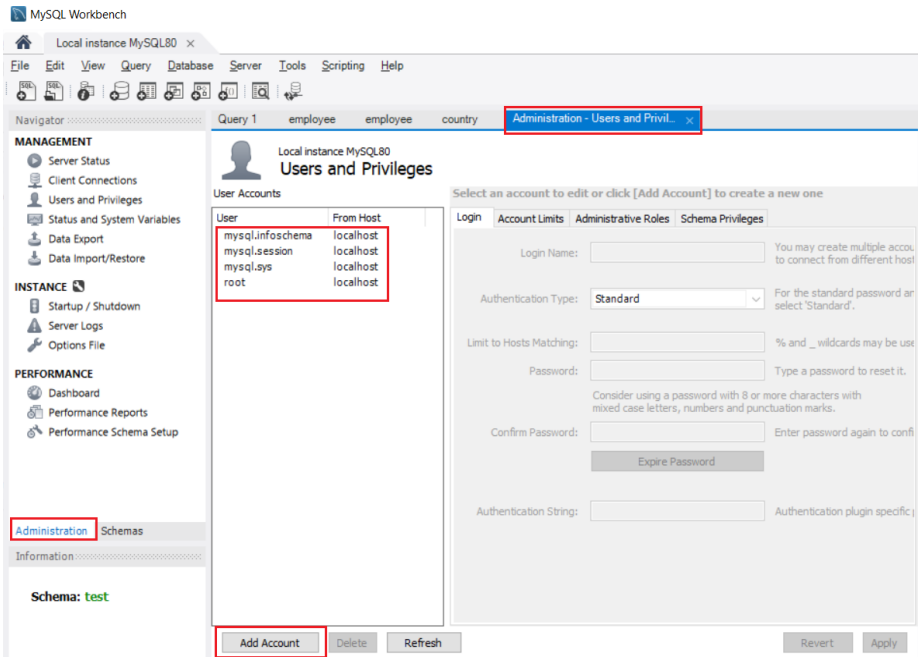


Once finished, right click on **Tables** folder under the schema and select **Refresh All** option to see the list of tables created.

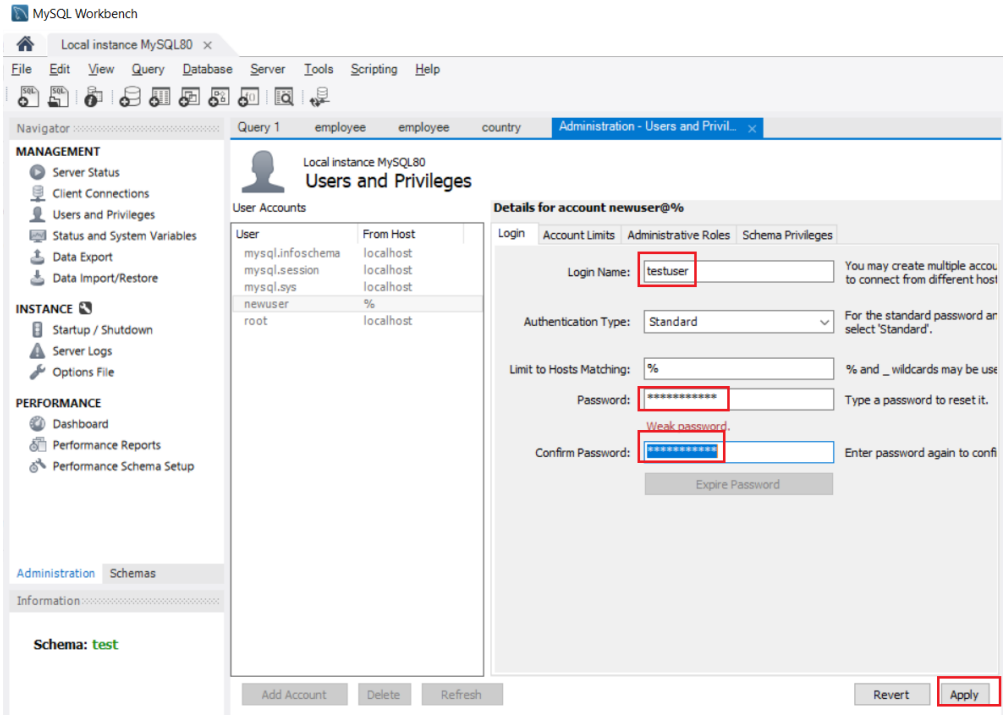


9.11. Create User:

To create a user, click on **Administration** tab on the right side and select **Users and Privileges** under **Management** which opens up **Administration- Users and Privileges** tab with the list of users available. Click on **Add Account** to create user.



Provide the new user Id and password and click on **Apply** button. Close **Administration- Users and Privileges** tab.



Congratulations!! You have successfully installed MySQL and executed SQL commands from various MySQL command line utilities and Workbench GUI.