

EXPLORE || DIGITAL  
SKILLS

SQL for Data Science  
**Creating a Relational Database**



## Installing MySQL Workbench

Connecting to MySQL Server

MySQL Workbench Overview

Table Relationships

Creating a Relational Database



# Installing MySQL workbench - Windows



Navigate to <https://dev.mysql.com/downloads/installer/> and download the MySQL installer.

General Availability (GA) Releases Archives ⓘ

### MySQL Installer 8.0.21

Select Operating System: Microsoft Windows

Accesses internet to download selected packages

Looking for previous GA versions?

Version	File Type	Size	Action
8.0.21	Windows (x86, 32-bit), MSI Installer (mysql-installer-web-community-8.0.21.0.msi)	24.5M	<b>Download</b>
8.0.21	Windows (x86, 32-bit), MSI Installer (mysql-installer-community-8.0.21.0.msi)	427.6M	<b>Download</b>

We suggest that you use the MD5 checksums and GPG signatures to verify the integrity of the packages you download.

All packages included.  
No additional downloads



If you are not using Windows, navigate [here](#) and choose the appropriate version for your operating system.  
Installation instructions for Mac users can be found [here](#) on the MySQL setup guide.



### MySQL Community Downloads

Login Now or Sign Up for a free account.

An Oracle Web Account provides you with the following advantages:

- Fast access to MySQL software downloads
- Download technical White Papers and Presentations
- Post messages in the MySQL Discussion Forums
- Report and track bugs in the MySQL bug system

No thanks, just start my download

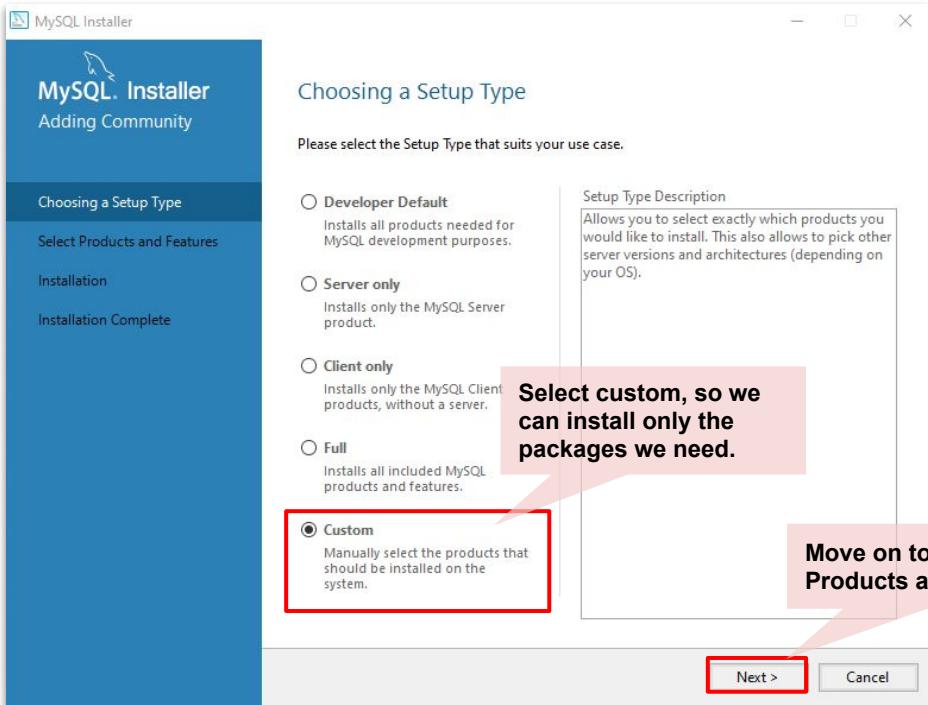
You can skip the account registration step and go straight to the download.

**ORACLE** © 2020, Oracle Corporation and/or its affiliates

[Legal Policies](#) | [Your Privacy Rights](#) | [Terms of Use](#) | [Trademark Policy](#) | [Contributor Agreement](#) | [Cookie Preferences](#)

# Installing MySQL workbench

Run the installer and follow the prompts until you get to the “Choosing a setup Type” screen:



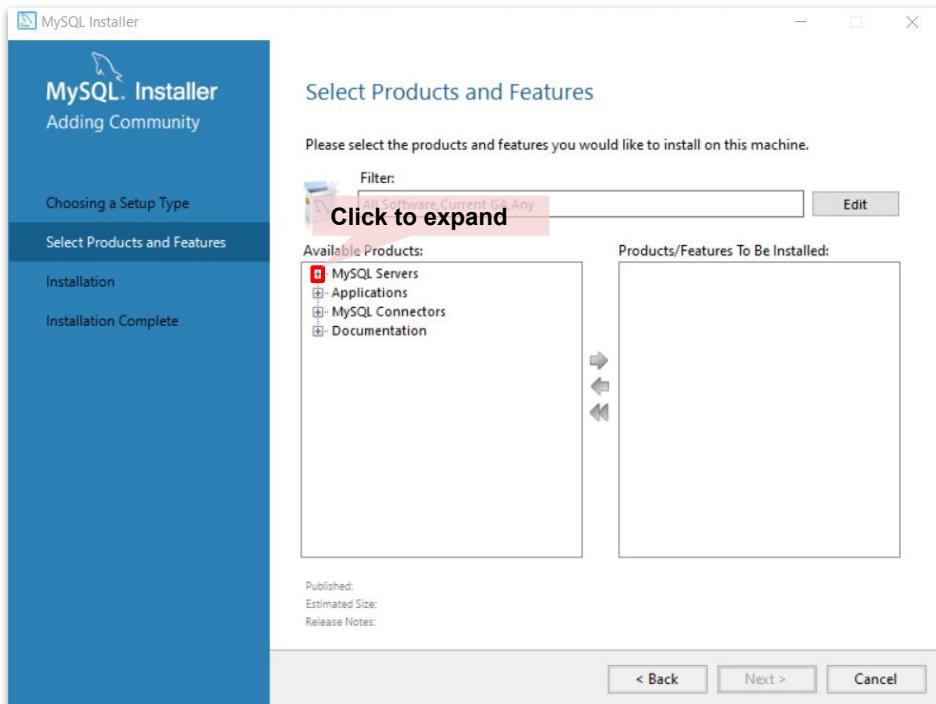
If you are prompted for permissions, click “yes” to allow.

Select custom, so we  
can install only the  
packages we need.

Move on to “Select  
Products and Features”

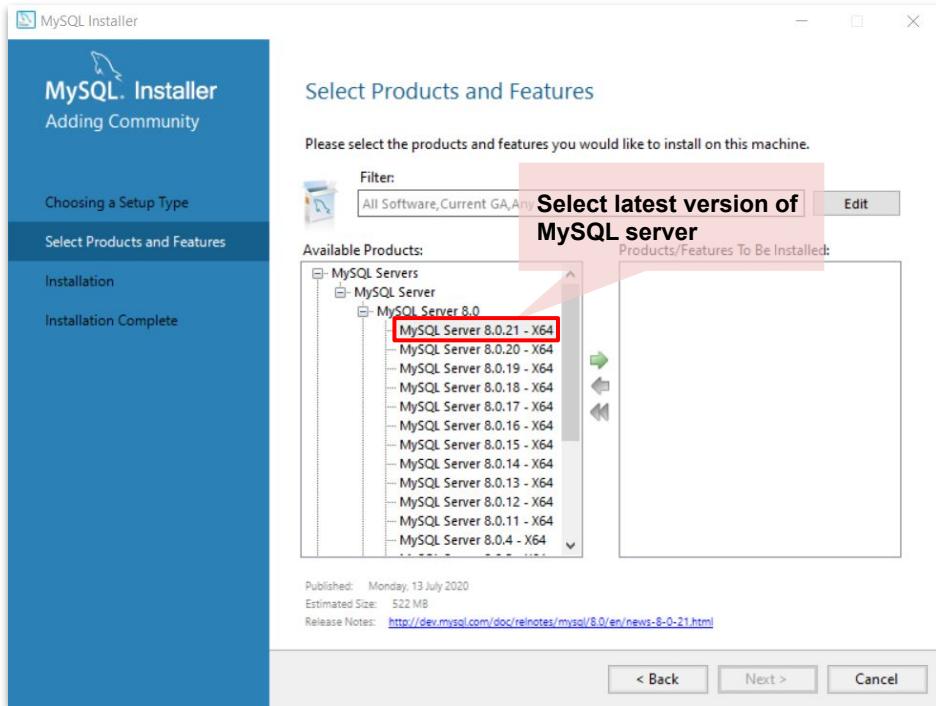
# Installing MySQL workbench

Select a MySQL server version and corresponding MySQL Workbench to install:



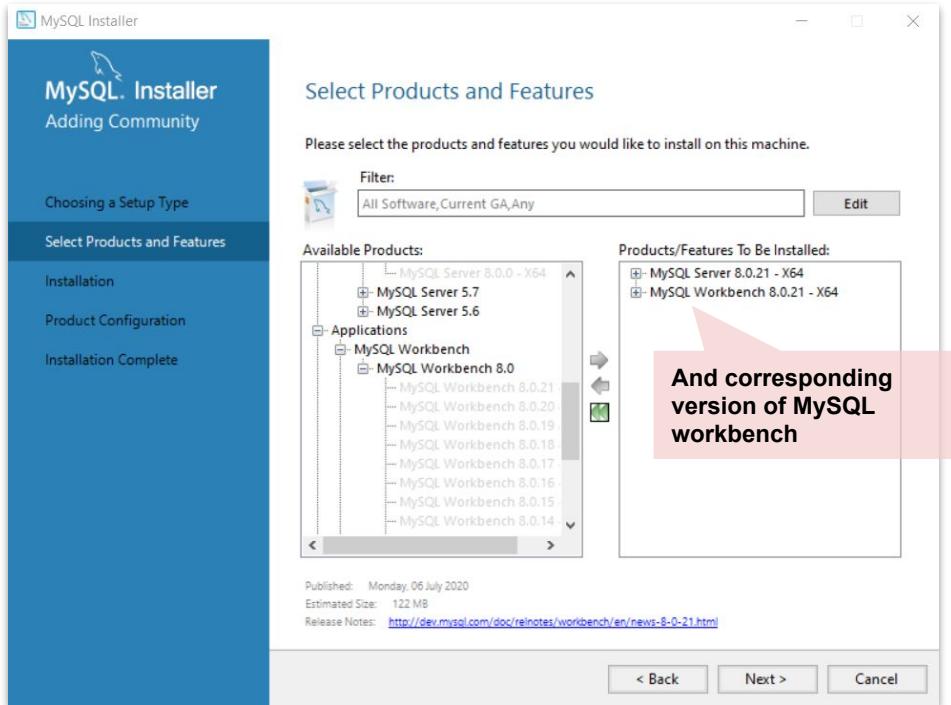
# Installing MySQL workbench

Select a MySQL server version and corresponding MySQL Workbench to install:



# Installing MySQL workbench

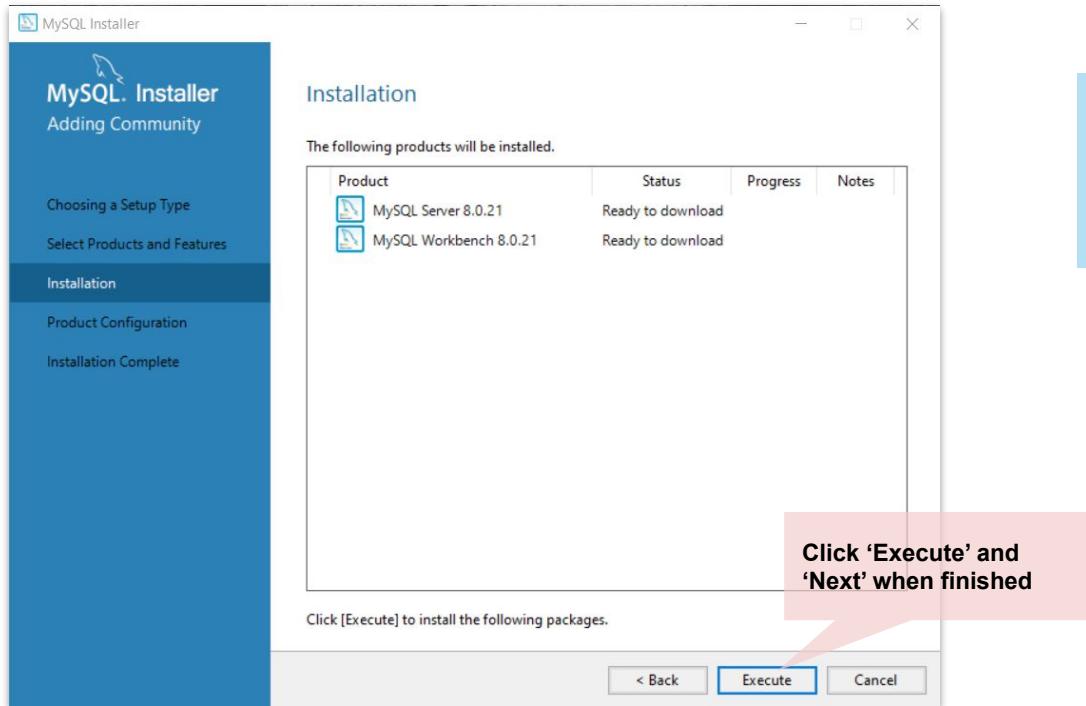
Select a MySQL server version and corresponding MySQL Workbench to install:



Depending on your level of comfort with SQL, it may also be useful to install **samples and examples** under the Documentation tab.

# Installing MySQL workbench

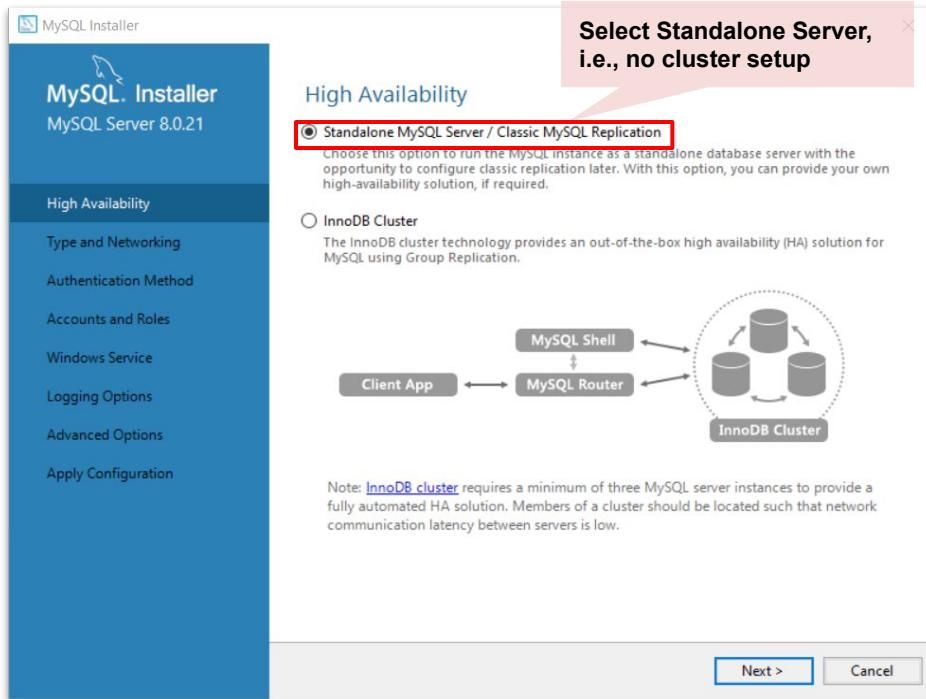
Install MySQL server and Workbench:



Depending on your level of comfort with SQL, it may also be useful to install **samples and examples** under the Documentation tab.

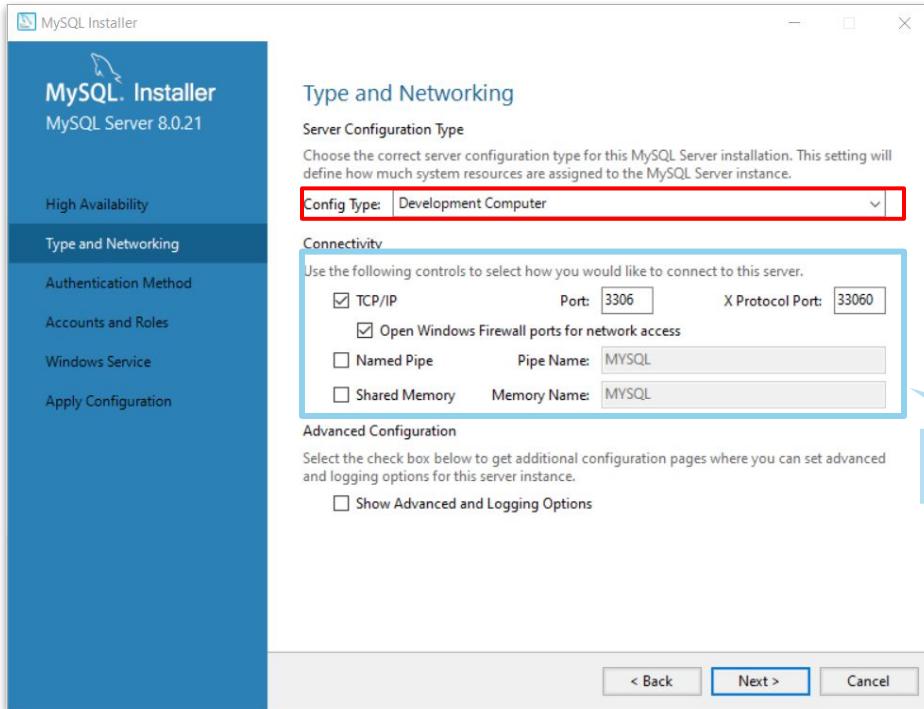
# Installing MySQL workbench

Configure MySQL Server:



# Installing MySQL workbench

Configure MySQL Server:



Choosing **Development Computer** means that we choose MySQL Server for an individual computer. This version of MySQL will also use the least amount of memory (i.e. RAM) as possible when it runs.

Leave everything here as is.

# Installing MySQL workbench

Configure MySQL Server:

The screenshot shows the MySQL Installer interface for MySQL Server 8.0.21. The left sidebar has tabs for High Availability, Type and Networking, Authentication Method (which is selected), Accounts and Roles, Windows Service, and Apply Configuration. The main panel is titled 'Authentication Method' and contains two options: 'Use Strong Password Encryption for Authentication (RECOMMENDED)' (selected) and 'Use Legacy Authentication Method (Retain MySQL 5.x Compatibility)'. A note above the first option says: 'Choose new authentication method since we don't have to support any MySQL version 5.x'. A warning icon is present next to the legacy method note. At the bottom are buttons for '< Back', 'Next >', and 'Cancel'.

Choose new authentication method since we don't have to support any MySQL version 5.x

MySQL Installer  
MySQL Server 8.0.21

High Availability

Type and Networking

**Authentication Method**

Use Strong Password Encryption for Authentication (RECOMMENDED)

MySQL 8 supports a new authentication based on improved stronger SHA256-based password methods. It is recommended that all new MySQL Server installations use this method going forward.

**Attention:** This new authentication plugin on the server side requires new versions of connectors and clients which add support for this new 8.0 default authentication (caching\_sha2\_password authentication).

Currently MySQL 8.0 Connectors and community drivers which use libmysqlclient 8.0 support this new method. If clients and applications cannot be updated to support this new authentication method, the MySQL 8.0 Server can be configured to use the legacy MySQL Authentication Method below.

Use Legacy Authentication Method (Retain MySQL 5.x Compatibility)

Using the old MySQL 5.x legacy authentication method should only be considered in the following cases:

- If applications cannot be updated to use MySQL 8 enabled Connectors and drivers.
- For cases where re-compilation of an existing application is not feasible.
- An updated, language specific connector or driver is not yet available.

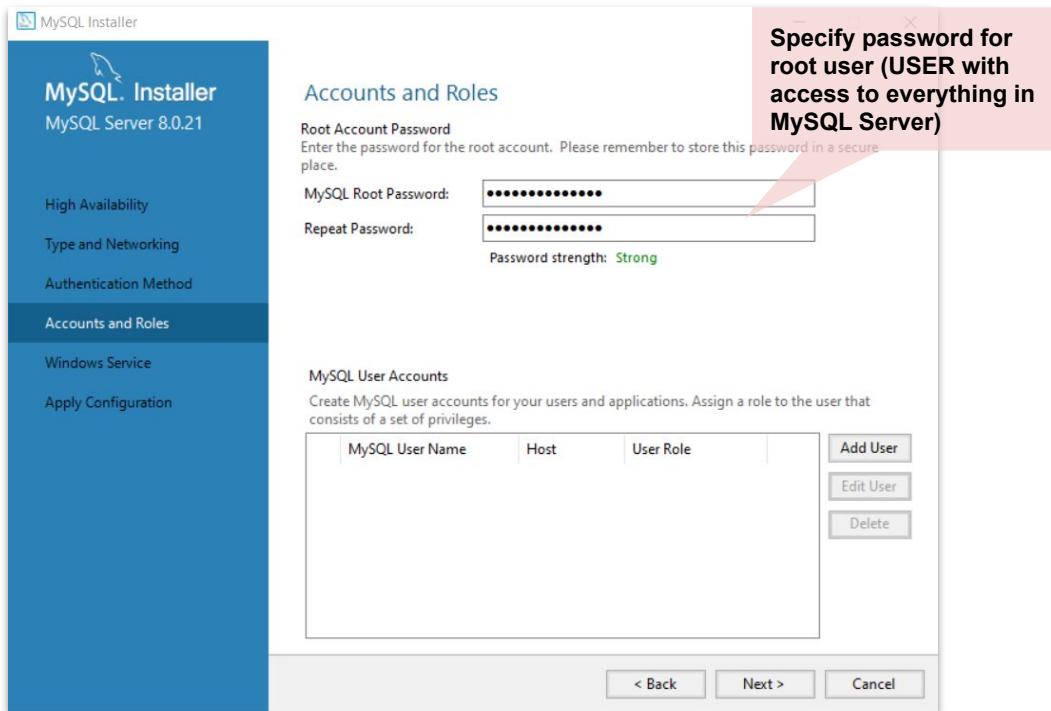
Security Guidance: When possible, we highly recommend taking needed steps towards upgrading your applications, libraries, and database servers to the new stronger authentication. This new method will significantly improve your security.

< Back    Next >    Cancel



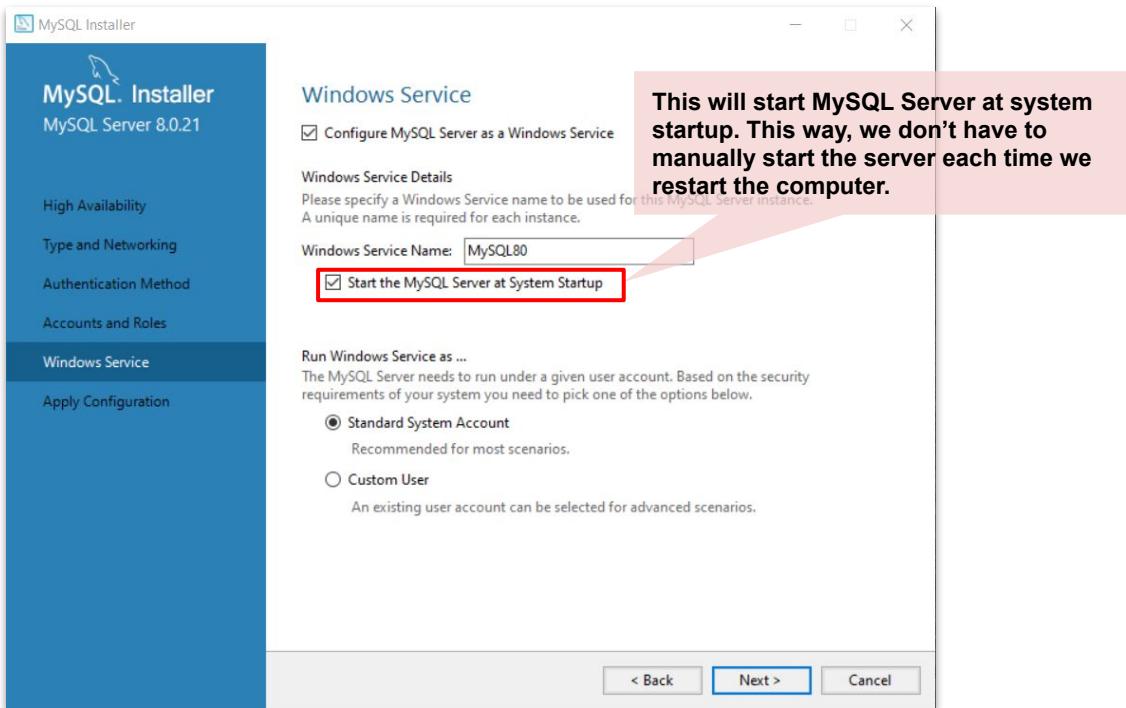
# Installing MySQL workbench

Configure MySQL Server:



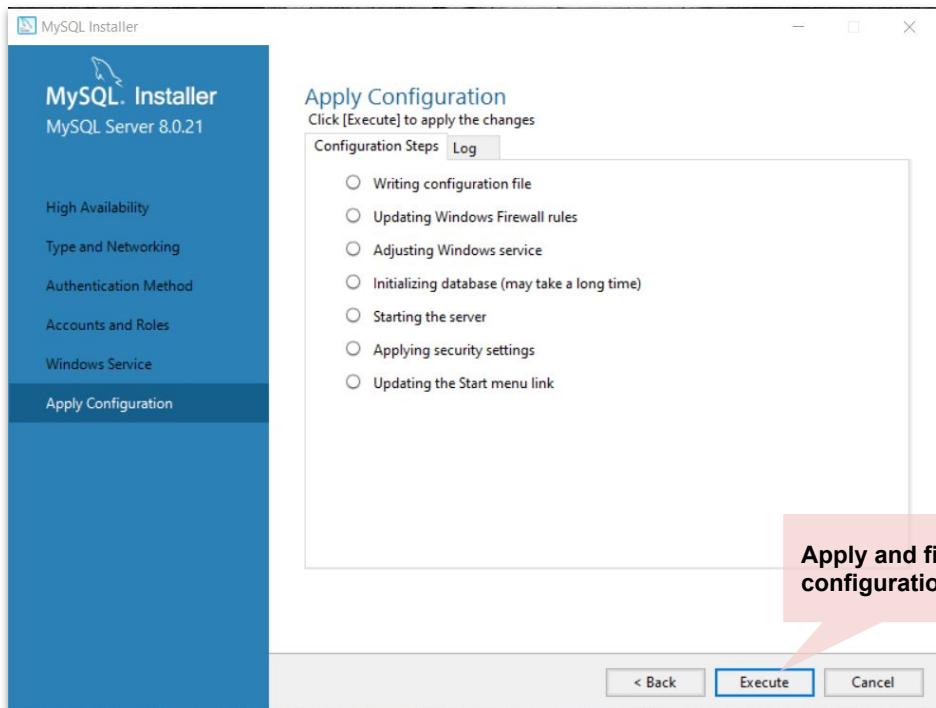
# Installing MySQL workbench

Configure MySQL Server:



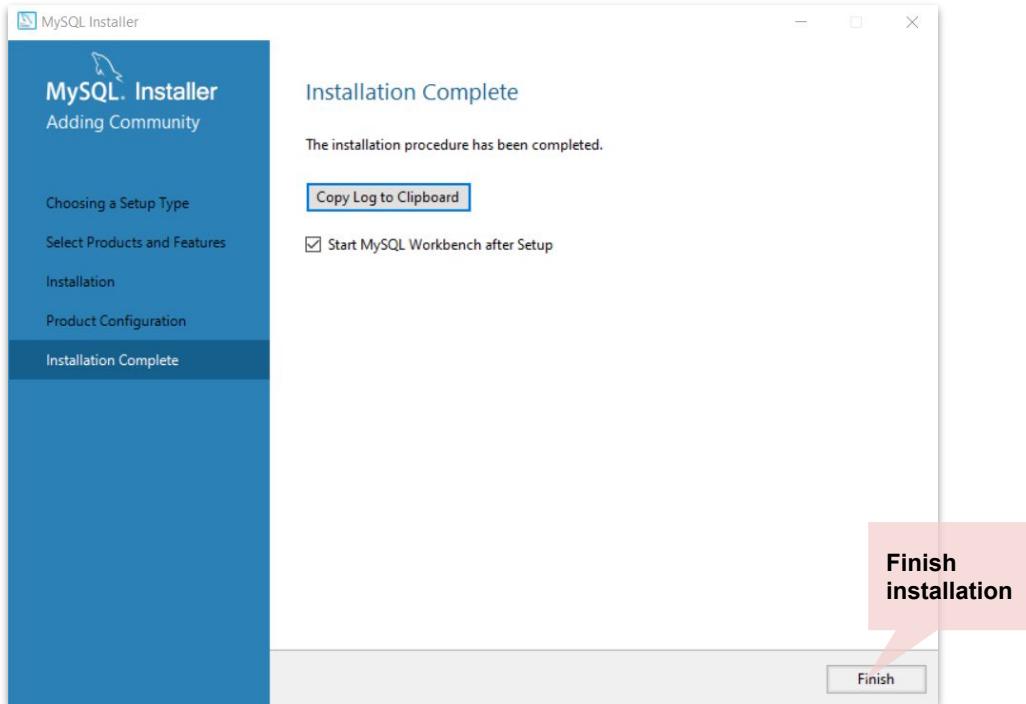
# Installing MySQL workbench

Apply configuration:



# Installing MySQL workbench

Finish installation:





Installing MySQL Workbench

## Connecting to MySQL Server

MySQL Workbench Overview

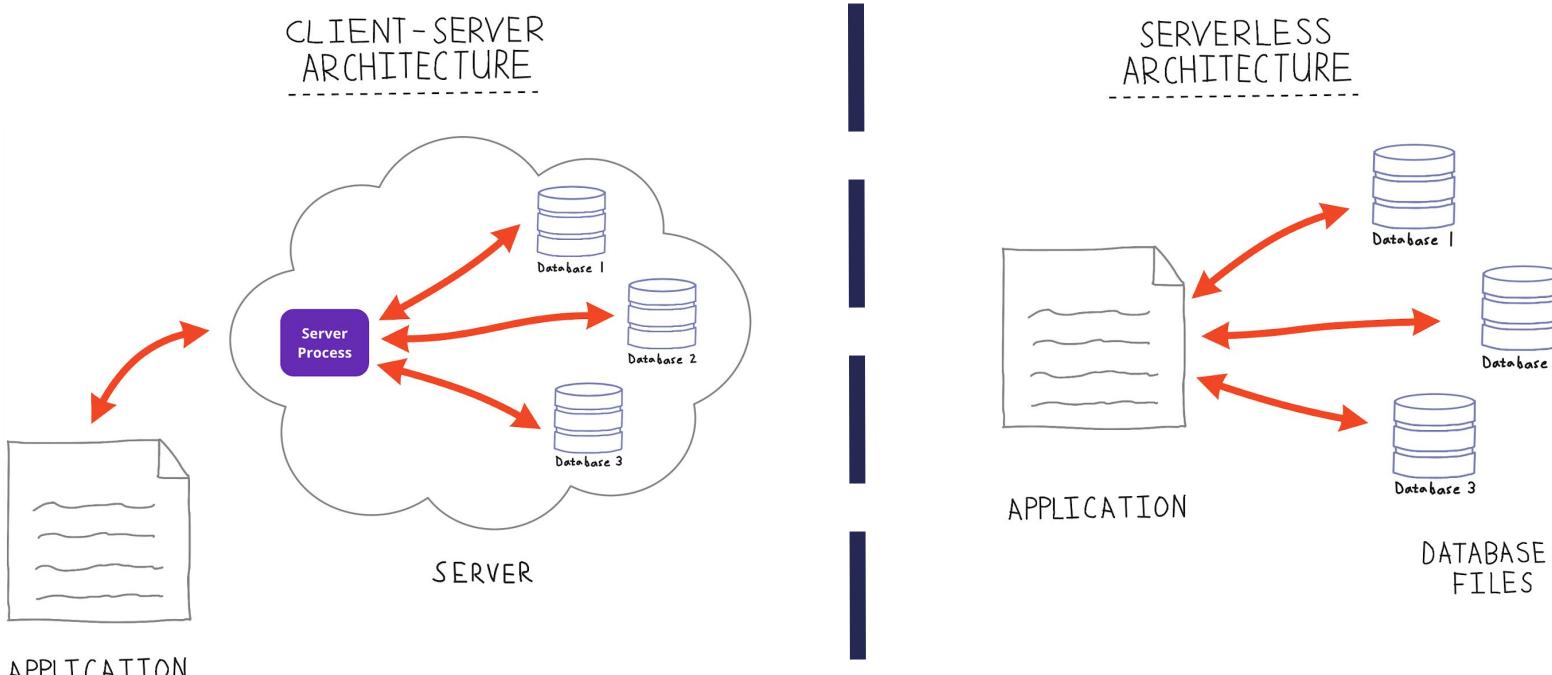
Table Relationships

Creating a Relational Database



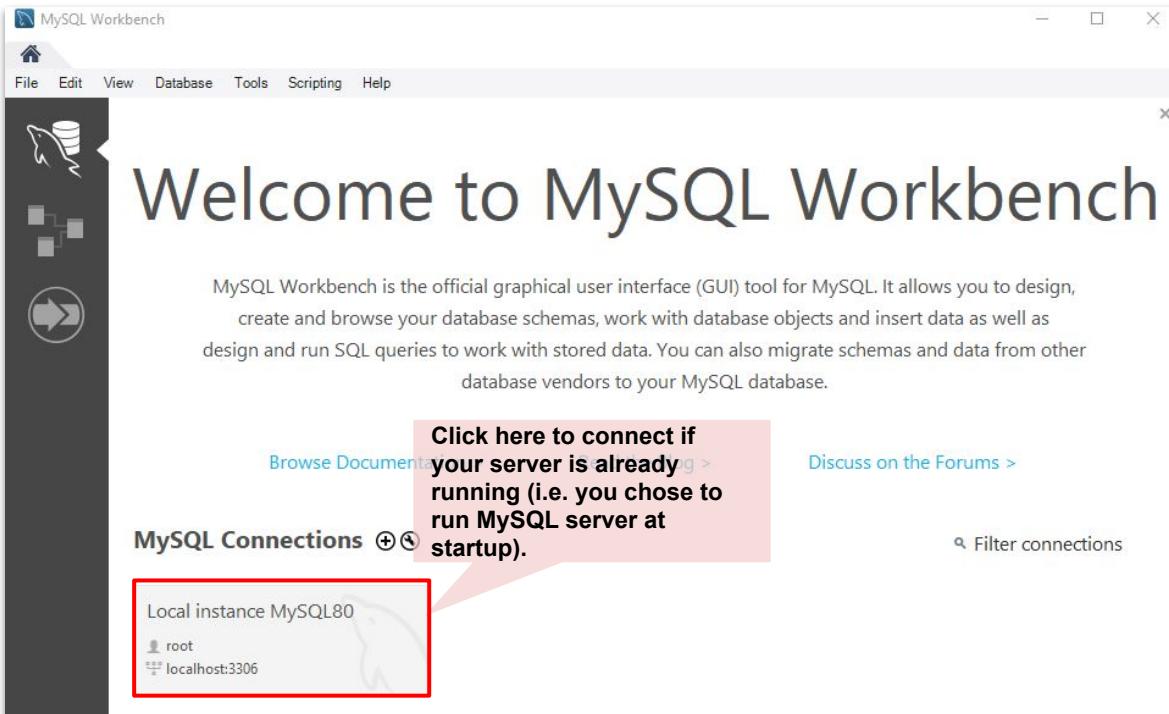
# Connecting to MySQL Server

MySQL follows a **client-server** paradigm, i.e., we interact with databases that are located within a server through a client application.



# Connecting to MySQL Server

To use our client application (i.e. MySQL workbench), we first have to establish a connection to the server (i.e. MySQL server).

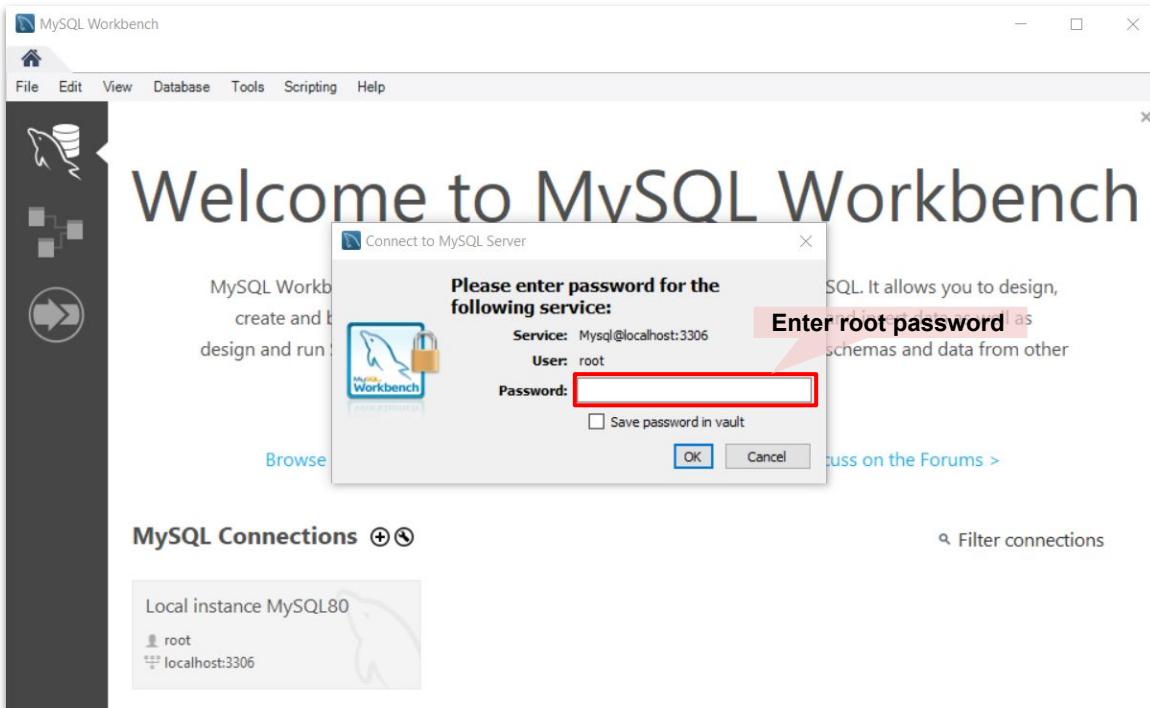


From this point onwards in the train, the MySQL Workbench will have a consistent interface regardless of the home operating system.

We show screenshots from Windows, but you should be able to follow along from any OS.

# Connecting to MySQL Server

To use our client application (i.e. MySQL workbench), we first have to establish a connection to the server (i.e. MySQL server).





Installing MySQL Workbench

Connecting to MySQL Server

## MySQL Workbench Overview

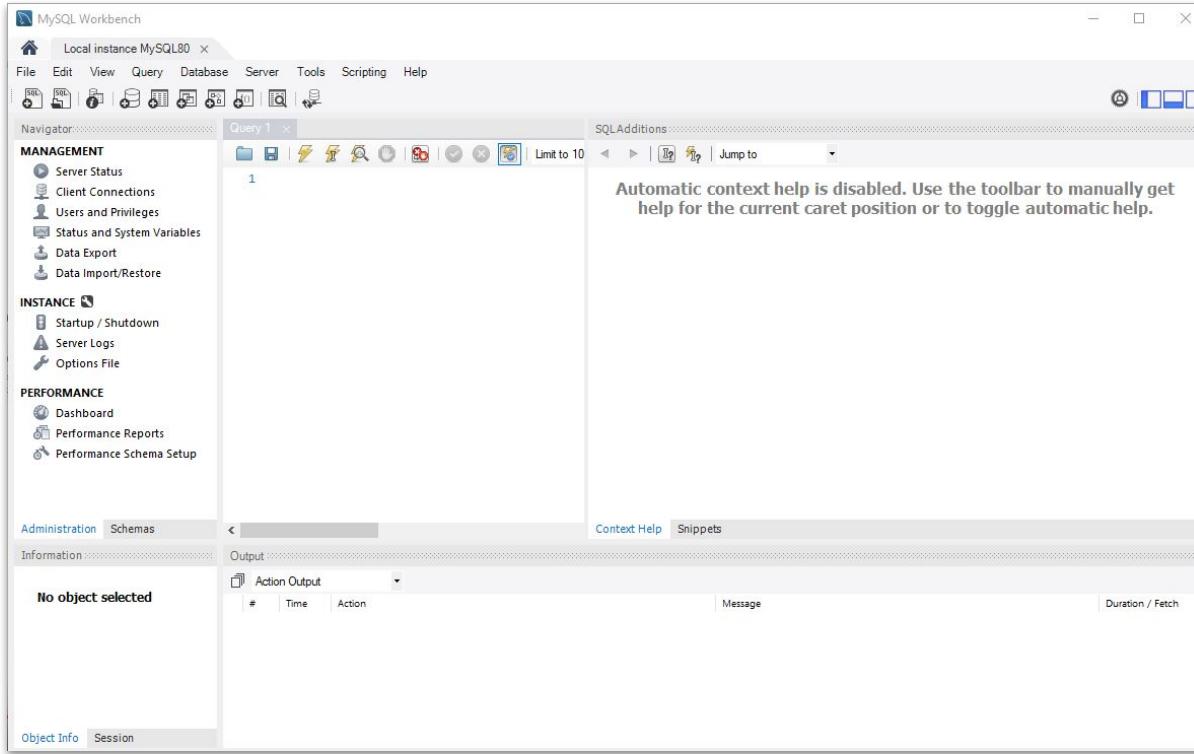
Table Relationships

Creating a Relational Database



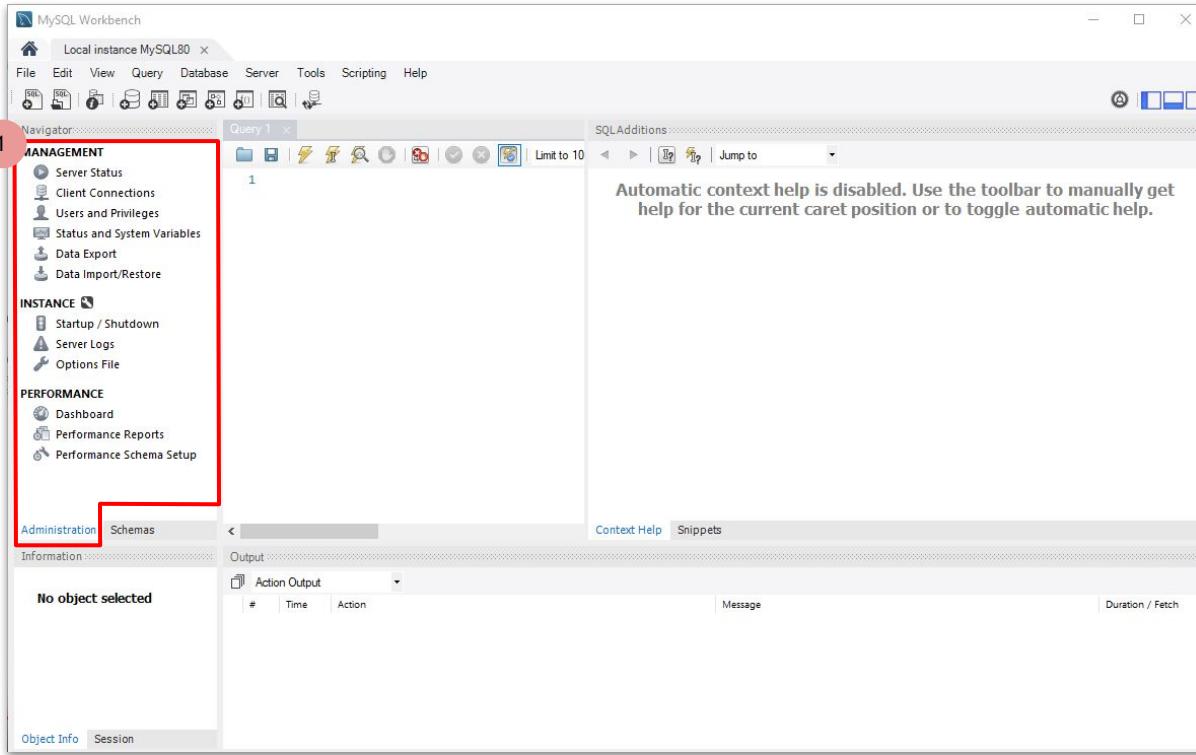
# MySQL Workbench Overview

After successfully connecting, MySQL workbench will open on the following home screen:



# MySQL Workbench Overview

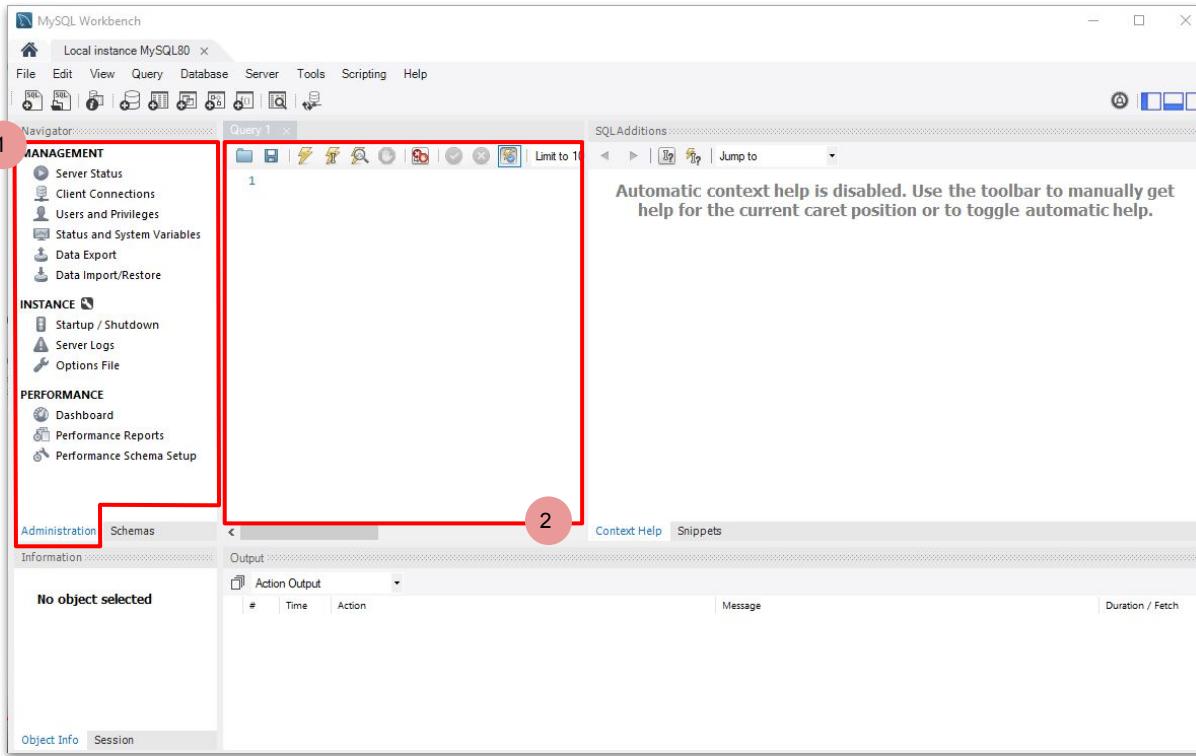
After a successfully connecting, MySQL workbench will open on the following home screen:



1. **Administration tab** - server configuration, user administration, and database health monitoring.

# MySQL Workbench Overview

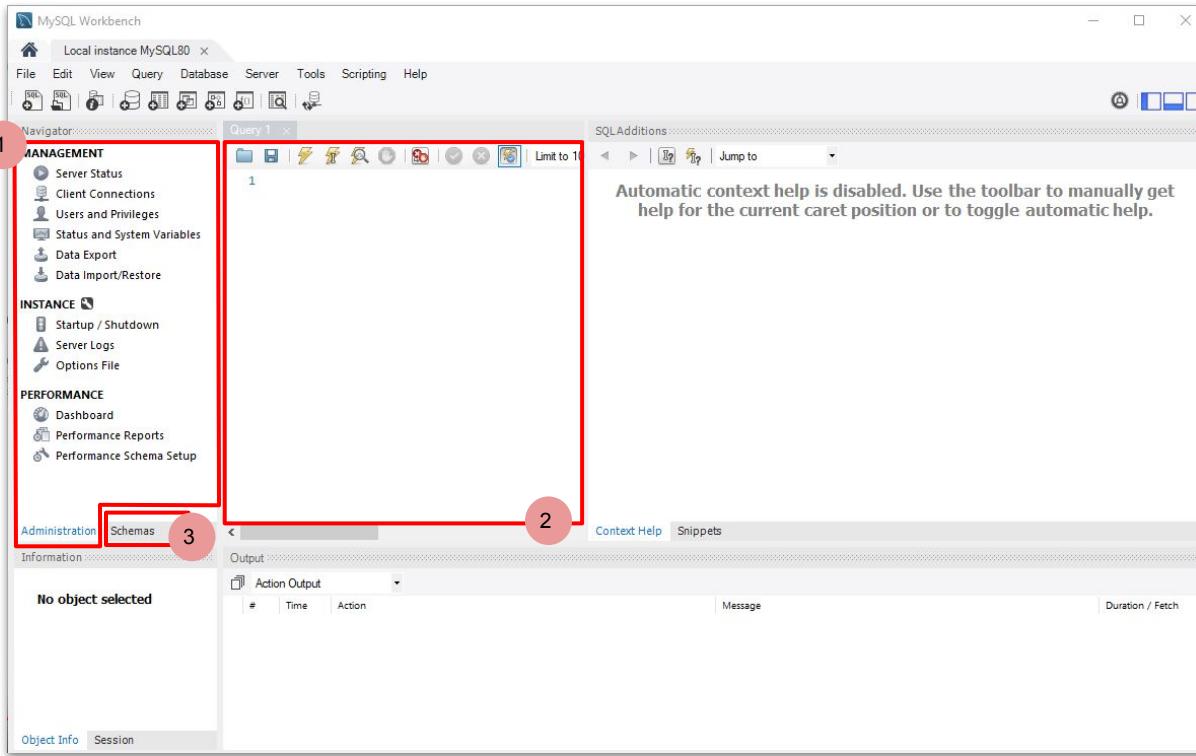
After a successfully connecting, MySQL workbench will open on the following home screen:



1. **Administration tab** - server configuration, user administration, and database health monitoring.
2. **SQL query pane** - Write and execute SQL queries

# MySQL Workbench Overview

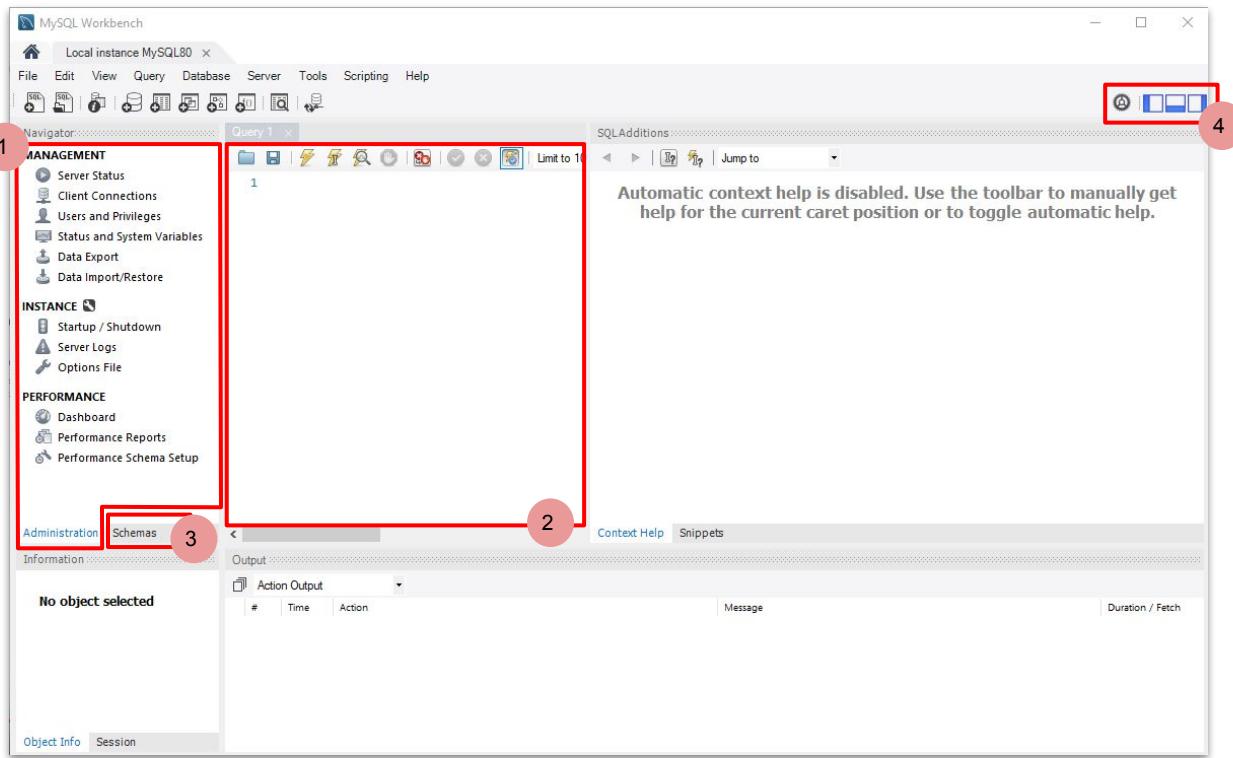
After a successfully connecting, MySQL workbench will open on the following home screen:



1. **Administration tab** - server configuration, user administration, and database health monitoring.
2. **SQL query pane** - Write and execute SQL queries
3. **Schemas tab** - View existing databases

# MySQL Workbench Overview

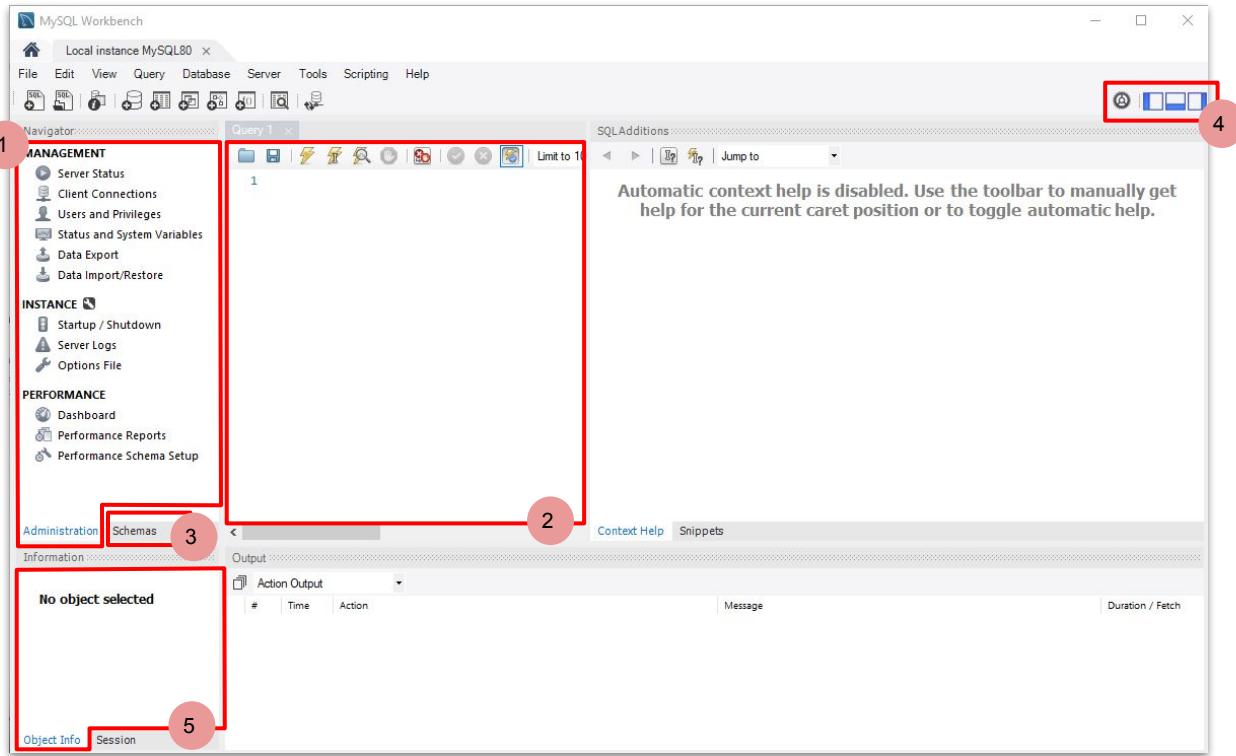
After a successfully connecting, MySQL workbench will open on the following home screen:



1. **Administration tab** - server configuration, user administration, and database health monitoring.
2. **SQL query pane** - Write and execute SQL queries
3. **Schemas tab** - View existing databases
4. **Layout buttons** - Toggle MySQL workbench layout

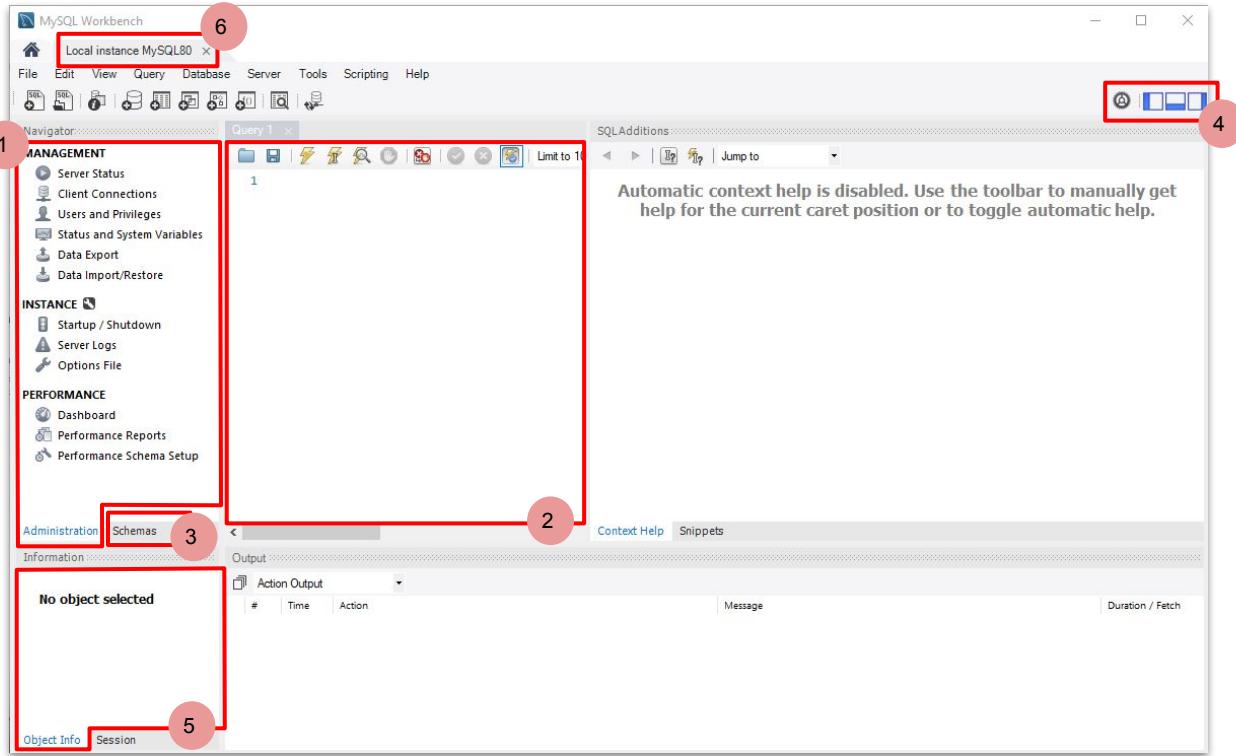
# MySQL Workbench Overview

After a successfully connecting, MySQL workbench will open on the following home screen:



# MySQL Workbench Overview

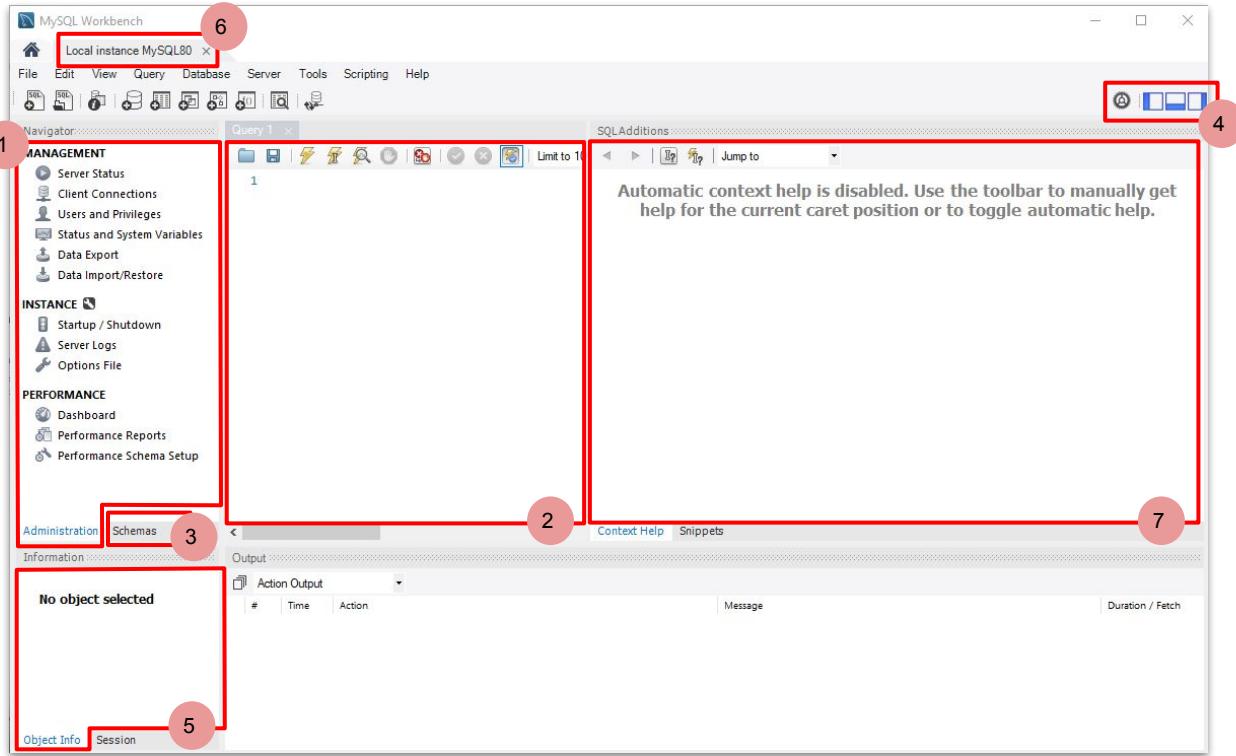
After a successfully connecting, MySQL workbench will open on the following home screen:



1. **Administration tab** - server configuration, user administration, and database health monitoring.
2. **SQL query pane** - Write and execute SQL queries
3. **Schemas tab** - View existing databases
4. **Layout buttons** - Toggle MySQL workbench layout
5. **Active object view** - view active schema (i.e. database).
6. **Active MySQL server connection**

# MySQL Workbench Overview

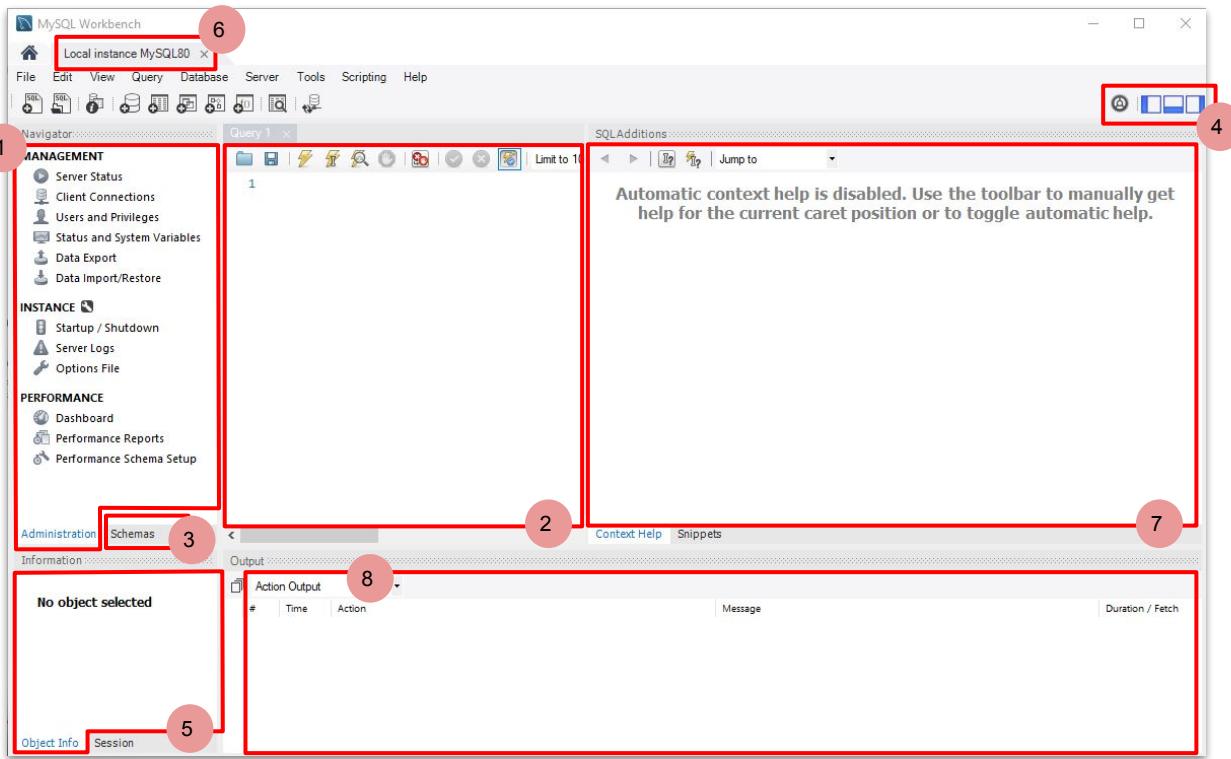
After a successfully connecting, MySQL workbench will open on the following home screen:



1. **Administration tab** - server configuration, user administration, and database health monitoring.
2. **SQL query pane** - Write and execute SQL queries
3. **Schemas tab** - View existing databases
4. **Layout buttons** - Toggle MySQL workbench layout
5. **Active object view** - view active schema (i.e. database).
6. **Active MySQL server connection**
7. **Additions pane** - for containing SQL statements or snippets

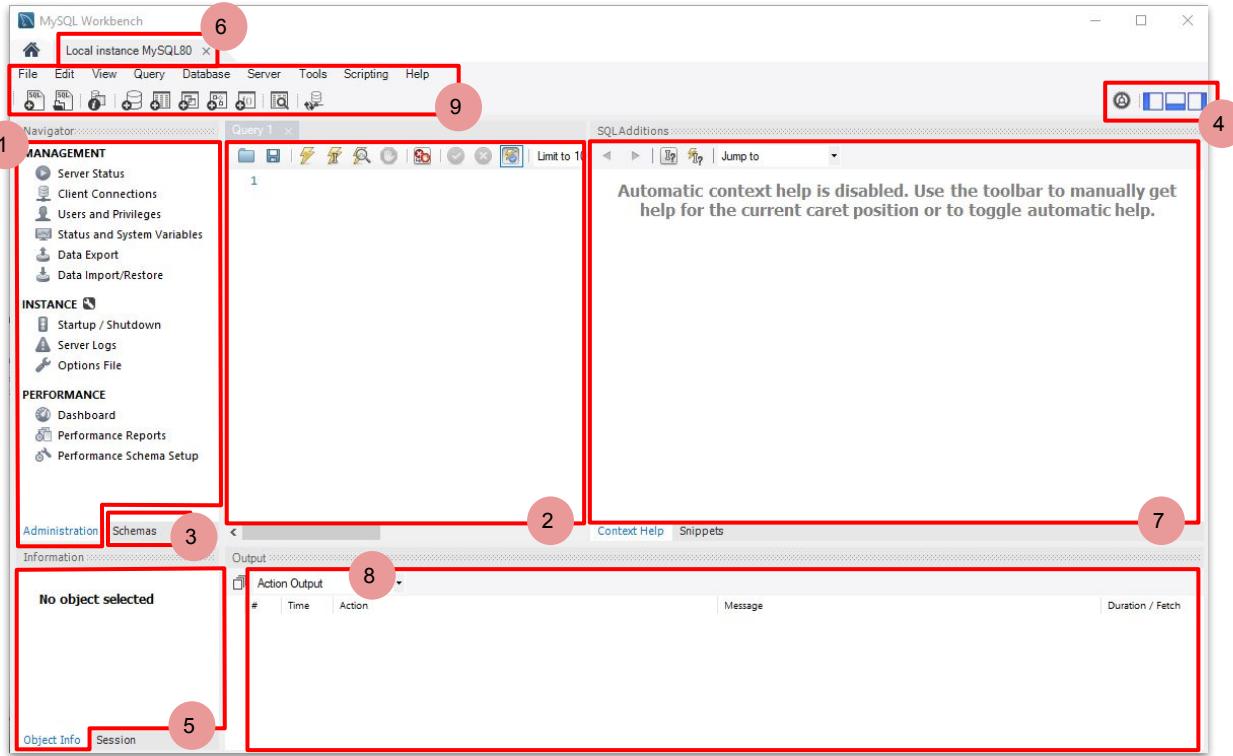
# MySQL Workbench Overview

After a successfully connecting, MySQL workbench will open on the following home screen:



# MySQL Workbench Overview

After a successfully connecting, MySQL workbench will open on the following home screen:



- Create or open SQL script file
- Create new database in active server connection
- Create new table in active database



Installing MySQL Workbench

Connecting to MySQL Server

MySQL Workbench Overview

## Table Relationships

Creating a Relational Database



# Table Relationships

Relational databases are collections of tables that are connected together. Each table represents an entity that has a collection of attributes (i.e. table columns). Tables in a database are connected to each other by means of relationships which are specified according to how different entities interact with each other. There are 4 main types of table relationships:

Relationship	ER diagram Symbol	Description
One to One		An instance in one table relates to a single instance in another table, e.g. a customer in the customers table can only have one address in the addresses table.
One to Many or Many to One		An instance in one table can correspond to multiple instances in another table, e.g. each customer from the customers table can have multiple invoices from the invoices table. For the “Many to One” relationship, the reverse is true.
Many to Many		Multiple instances of one table can correspond to multiple instances of another table, e.g. one invoice can contain multiple items and each item can be in multiple invoices.
Self Referencing Relationship	* depends on nature of relationship	An instance in one table can correspond to another instance in the same table, i.e. through a different attribute.

# Table Relationships

In order to connect a table to one or more tables in the database, we need to specify table relationships using key columns. Two tables are regarded as connected if one or both of the tables contain information that is related to one or more of the other tables columns. This way, information about an instance of an entity can be distributed across multiple tables in the database. There are two main types of key columns :

## Primary keys

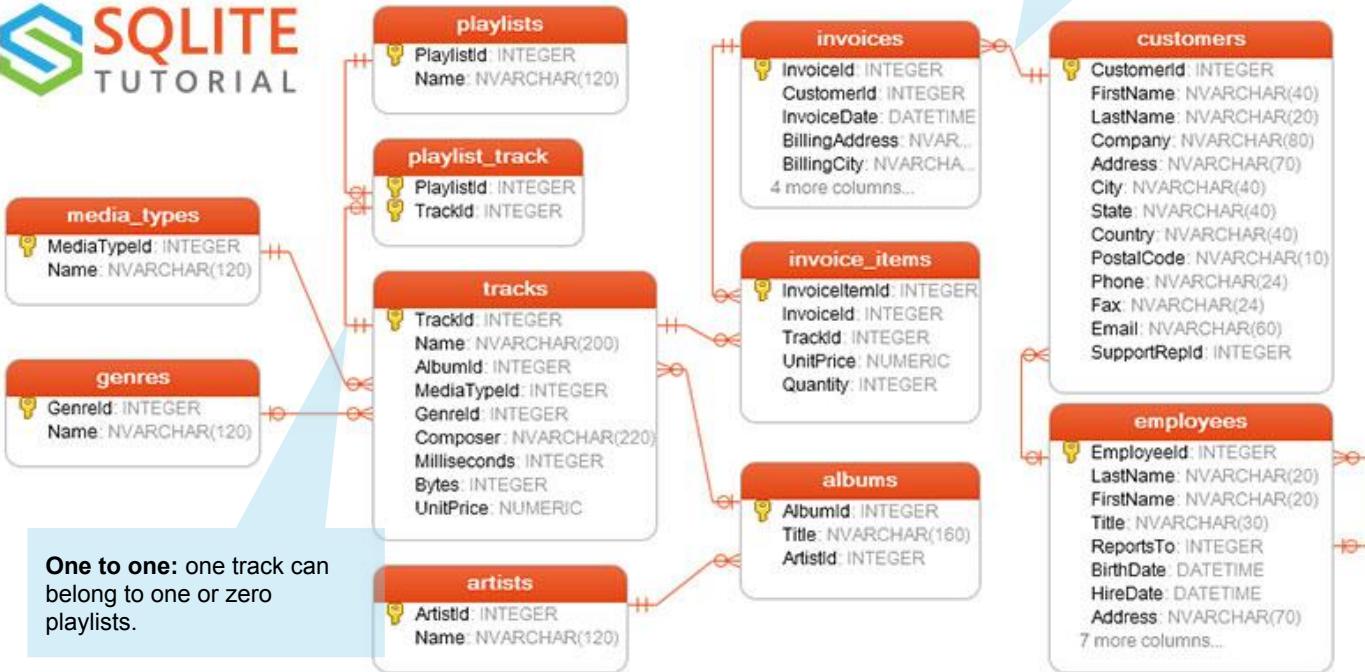
This is an attribute (i.e. column) that can be used to uniquely identify any row in the table. By definition, primary key columns cannot have duplicate or missing entries. As such, we usually auto-generate primary key values using special columns, e.g. INT AUTO\_INCREMENT, that generate a value each time we add a row to the table.

## Foreign keys

This is an attribute (or set of attributes) in a table whose value corresponds to a primary key in another table. Unlike primary keys, foreign keys can have duplicate entries. However, foreign keys columns must still obey the constraint that each entry in the column has to correspond to some value in primary key column of the connected table.

# Table Relationships

Chinook ER relationship examples:



**One to Many:** One and only one customer can have zero or many invoices.

Primary key columns in the database are indicated with the key icons

## More relationships

- Zero or many
- Zero or one
- One and only one

**Self reference (many to one):** Multiple employees can report to the same manager.



Installing MySQL Workbench

Connecting to MySQL Server

MySQL Workbench Overview

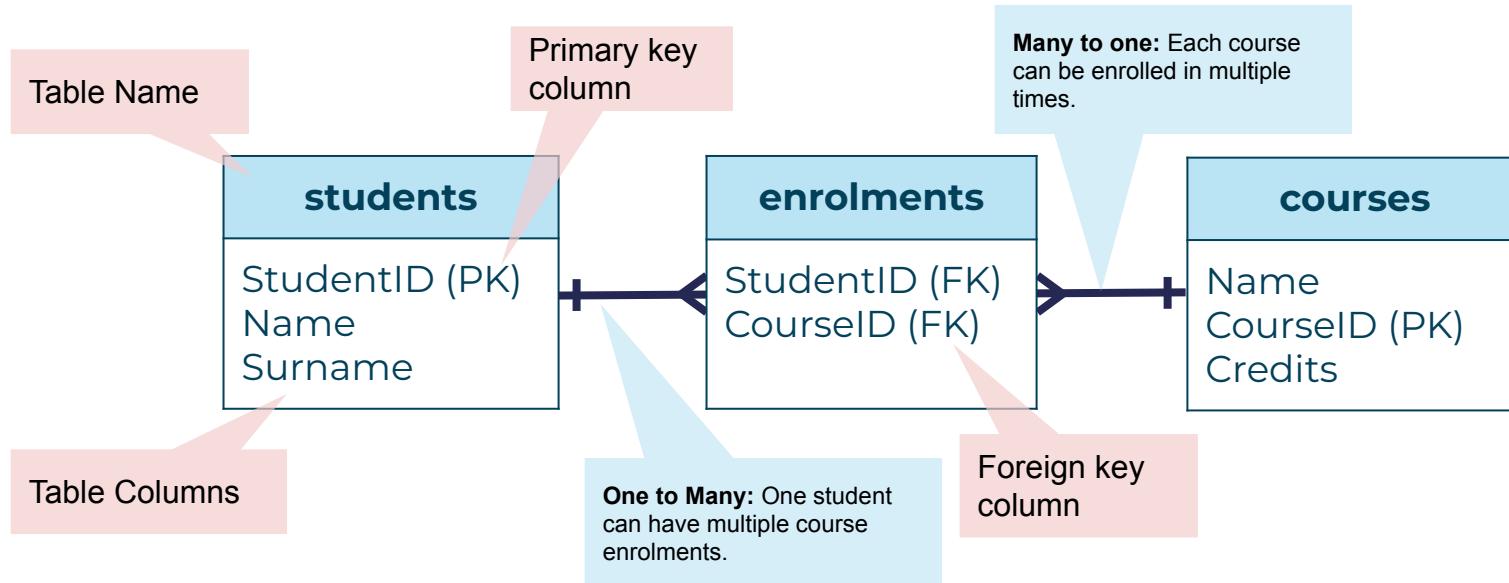
Table Relationships

## Creating a Relational Database



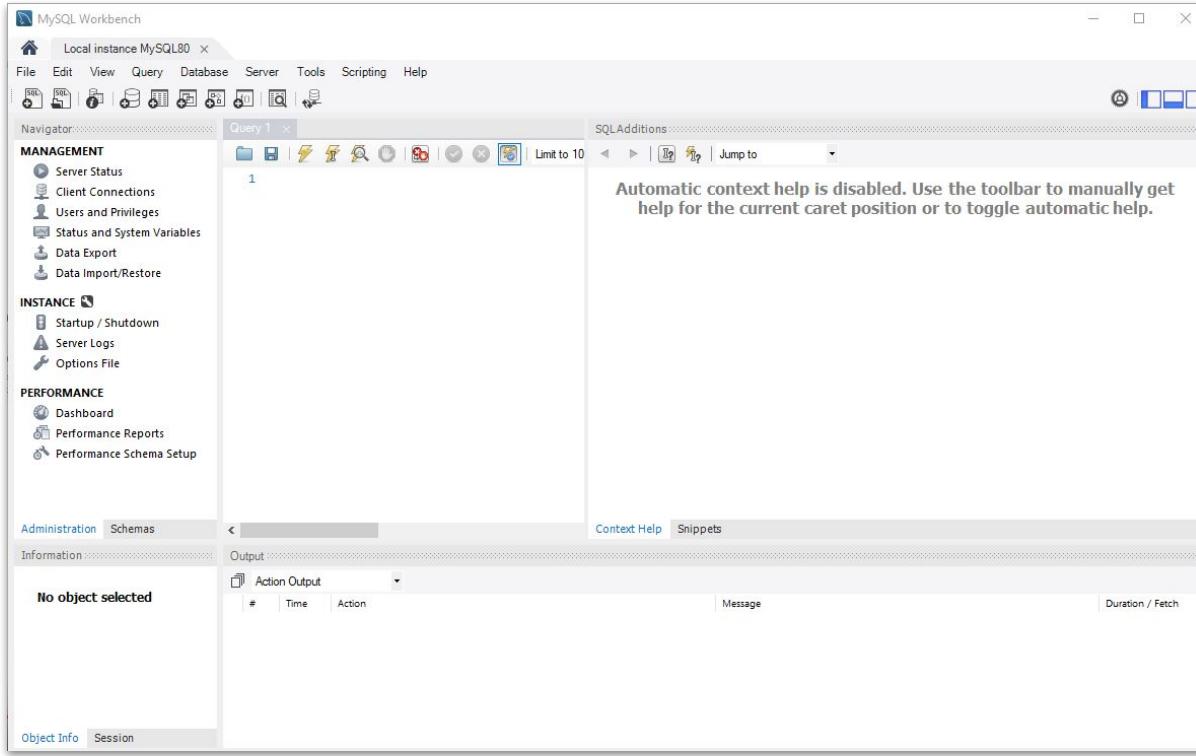
# Creating a Relational Database

Now that we understand table relationships, let's create our own database. We will use MySQL workbench to create a college database structured according to the following ER diagram:



# Creating a Relational Database

In this section, we cover how to create the database using the MySQL workbench interface:

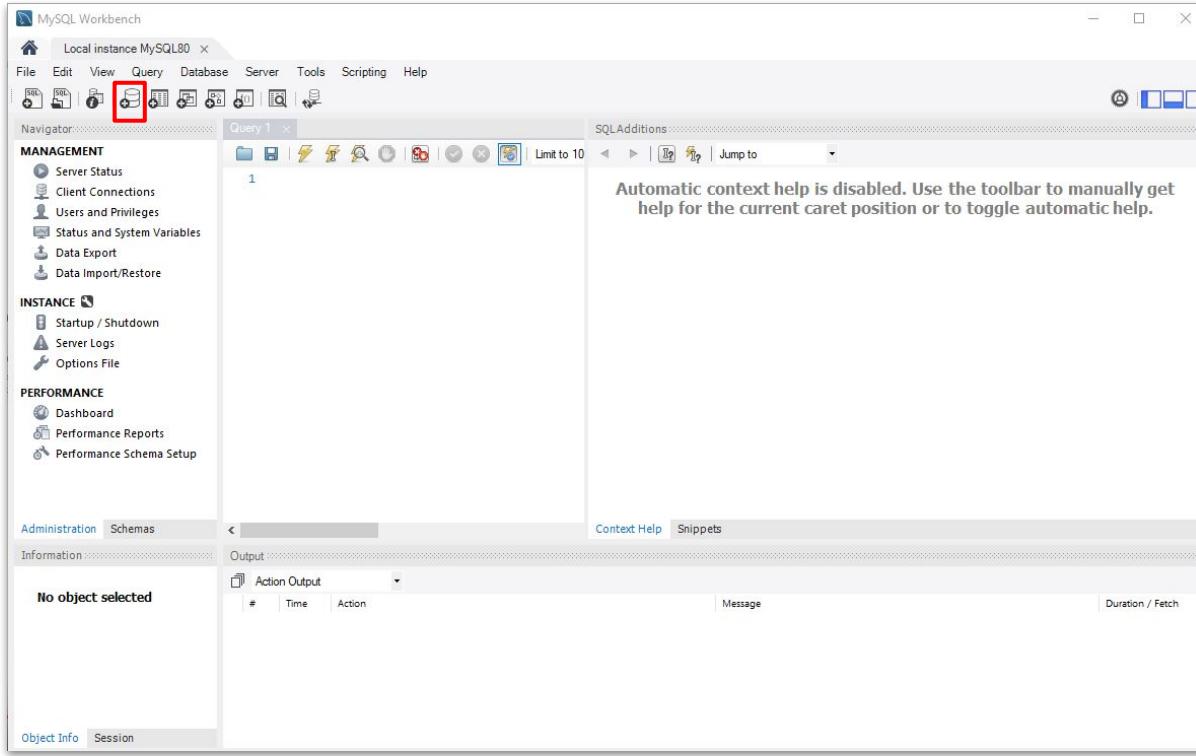


## Creating a Schema

1. Open MySQL workbench and connect to MySQL server as shown earlier in this train.

# Creating a Relational Database

In this section, we cover how to create the database using the MySQL workbench interface:

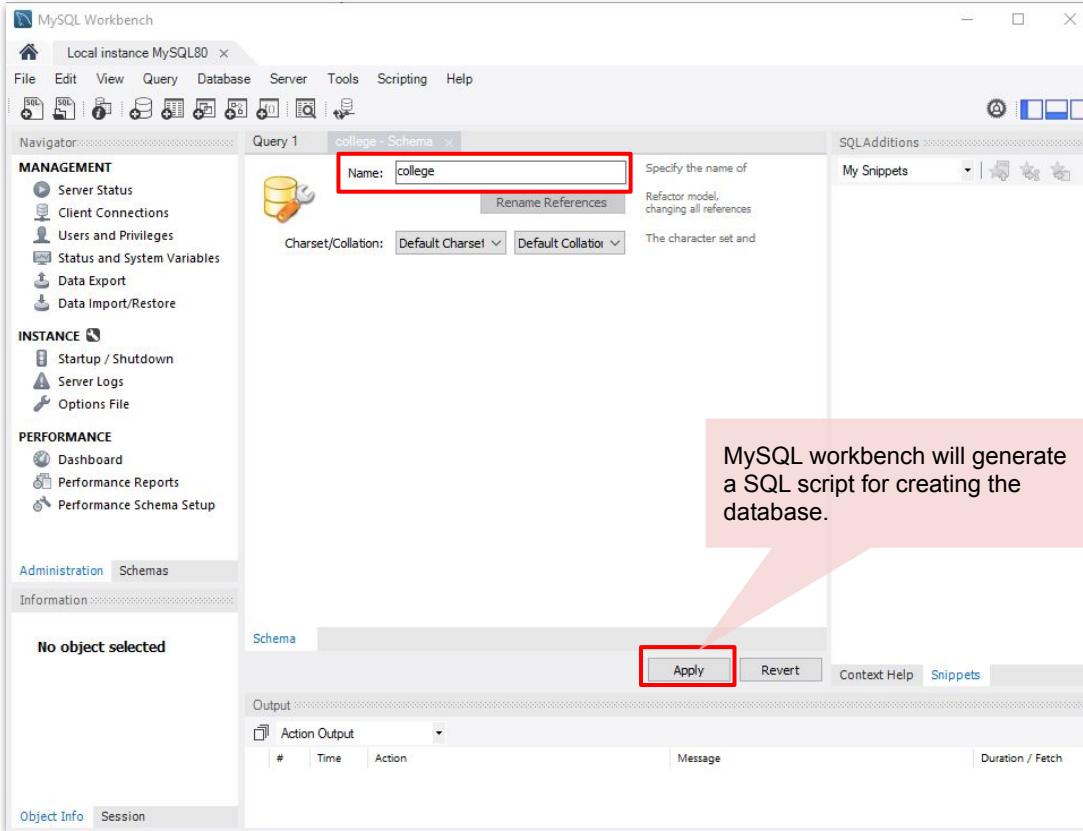


## Creating a Schema

1. Open MySQL workbench and connect to MySQL server as shown earlier in this train.
2. Click the create schema button , to create the database.

# Creating a Relational Database

In this section, we cover how to create the database using the MySQL workbench interface:

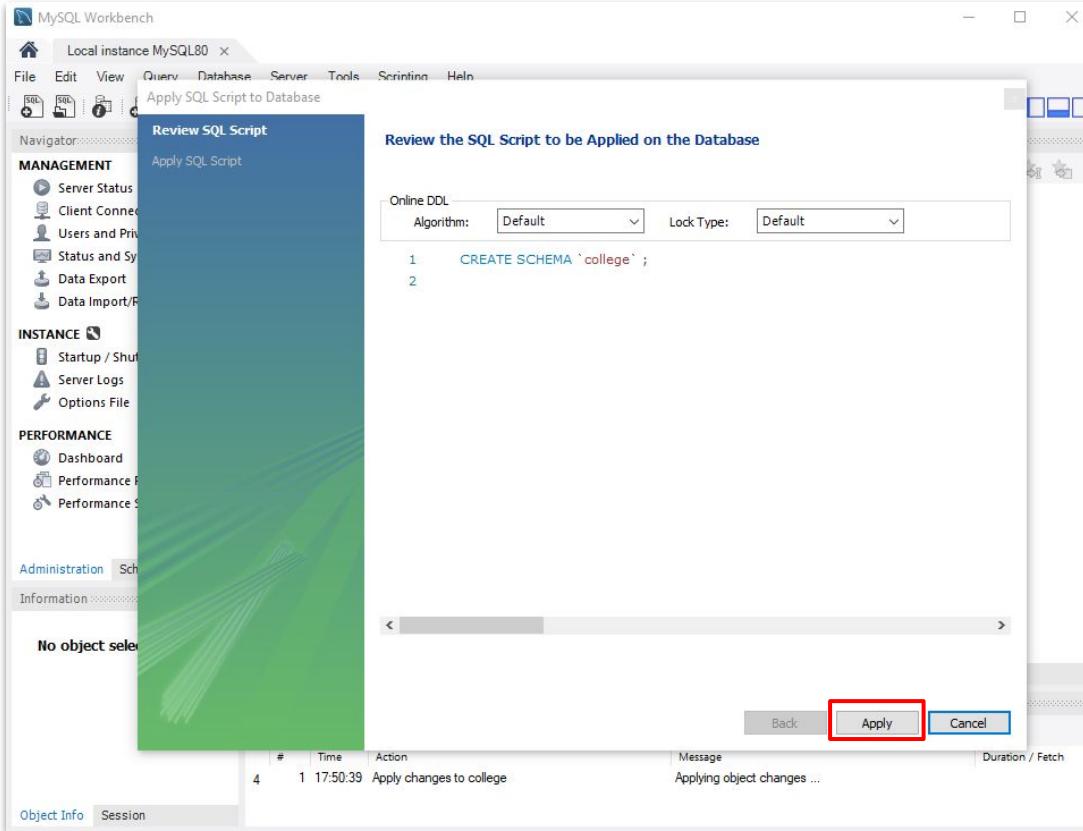


## Creating a Schema

1. Open MySQL workbench and connect to MySQL server as shown earlier in this train.
2. Click the create schema button to create the database.
3. Specify a name for the database in the wizard and click apply.

# Creating a Relational Database

In this section, we cover how to create the database using the MySQL workbench interface:

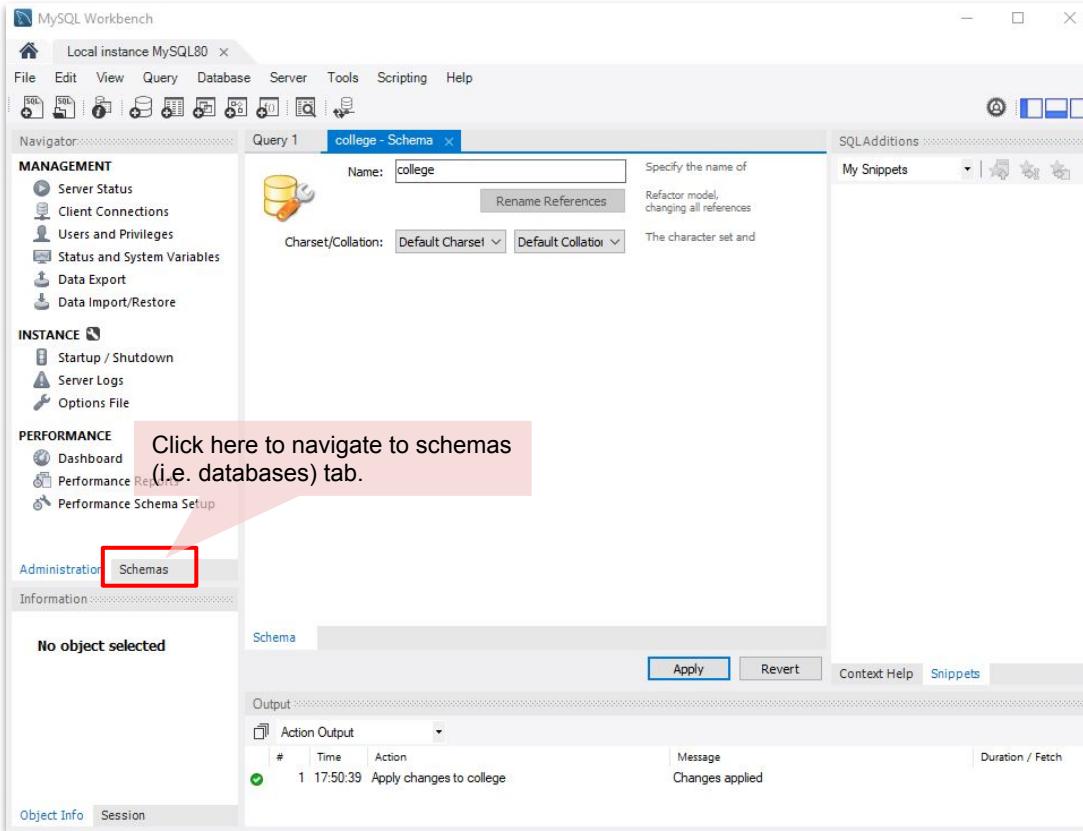


## Creating a Schema

1. Open MySQL workbench and connect to MySQL server as shown earlier in this train.
2. Click the create schema button , to create the database.
3. Specify a name for the database in the wizard and click apply.
4. Click apply and then finish.

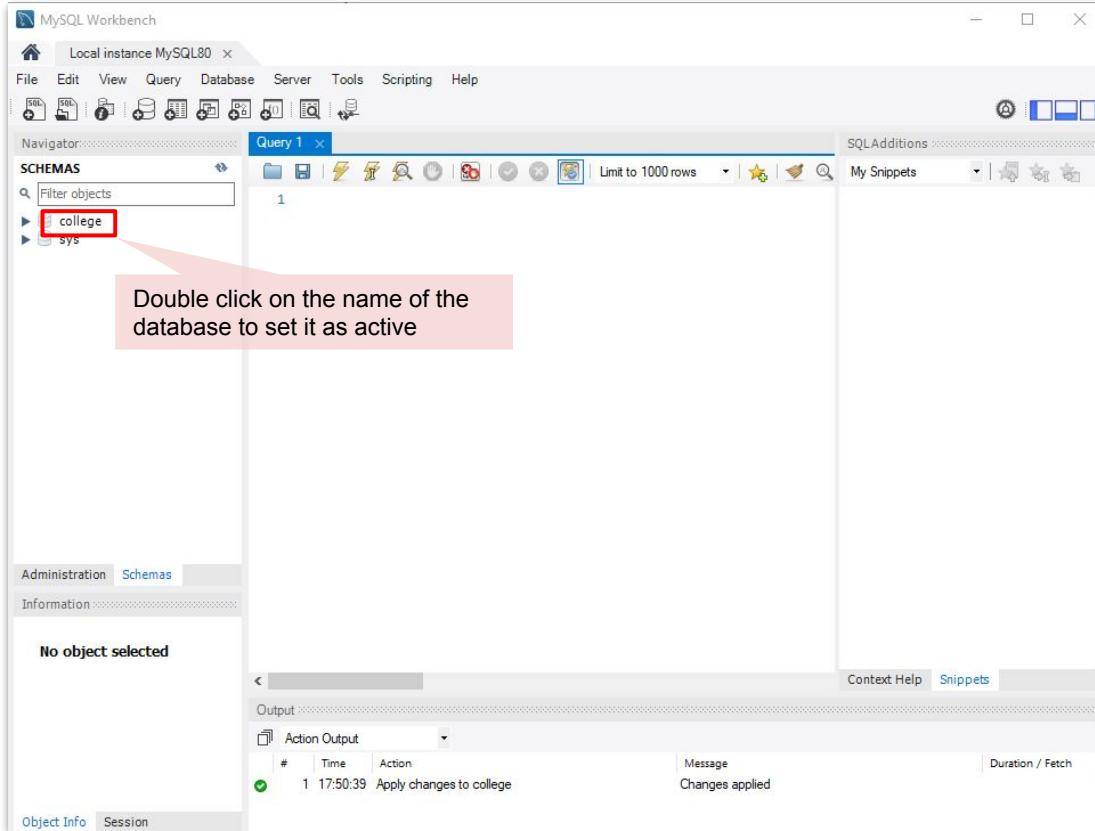
# Creating a Relational Database

After creating the database, we need to set it as active. To do this, double click its name in the the schemas tab.



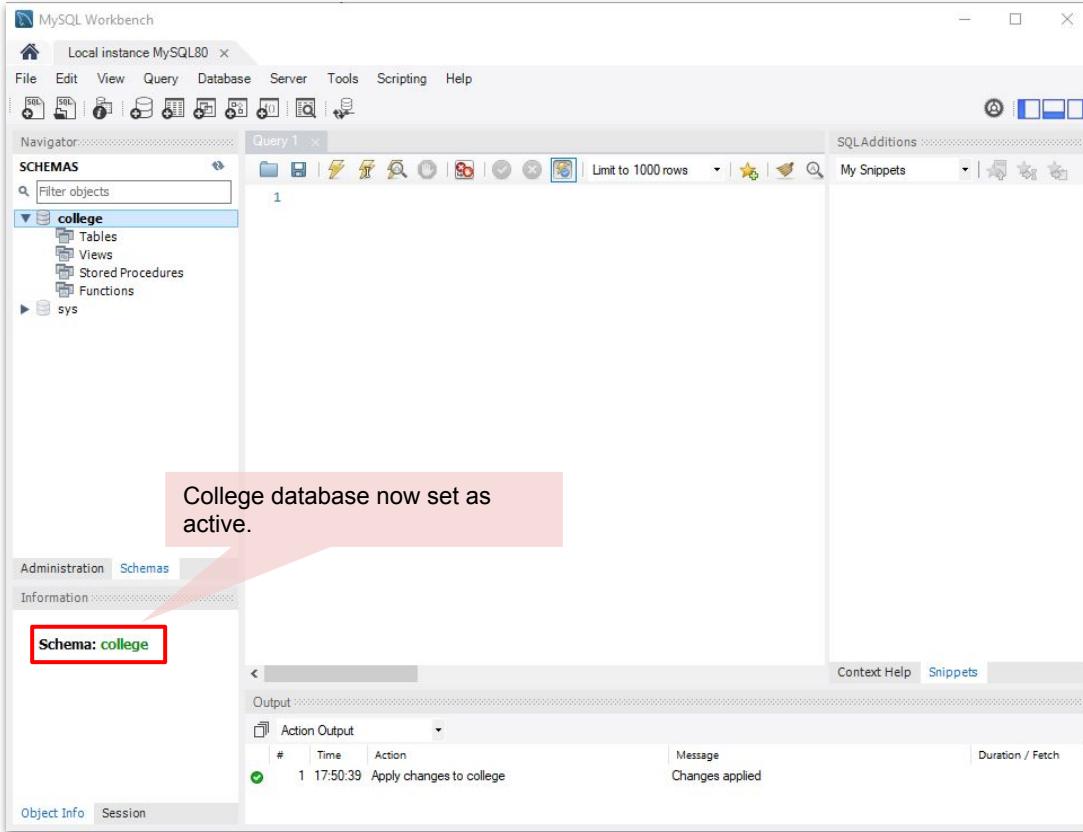
# Creating a Relational Database

After creating the database, we need to set it as active. To do this, double click its name in the the schemas tab.



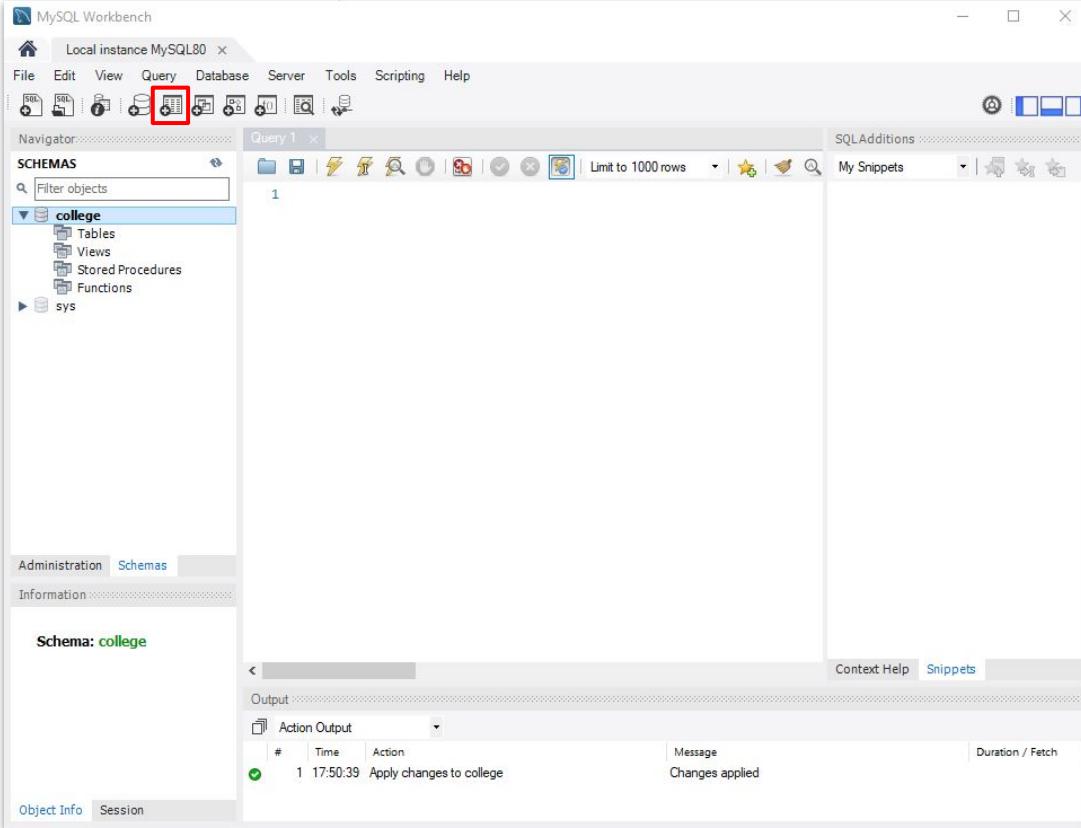
# Creating a Relational Database

After creating the database, we need to set it as active. To do this, double click its name in the the schemas tab.



# Creating a Relational Database

Next, we create tables under the active database:

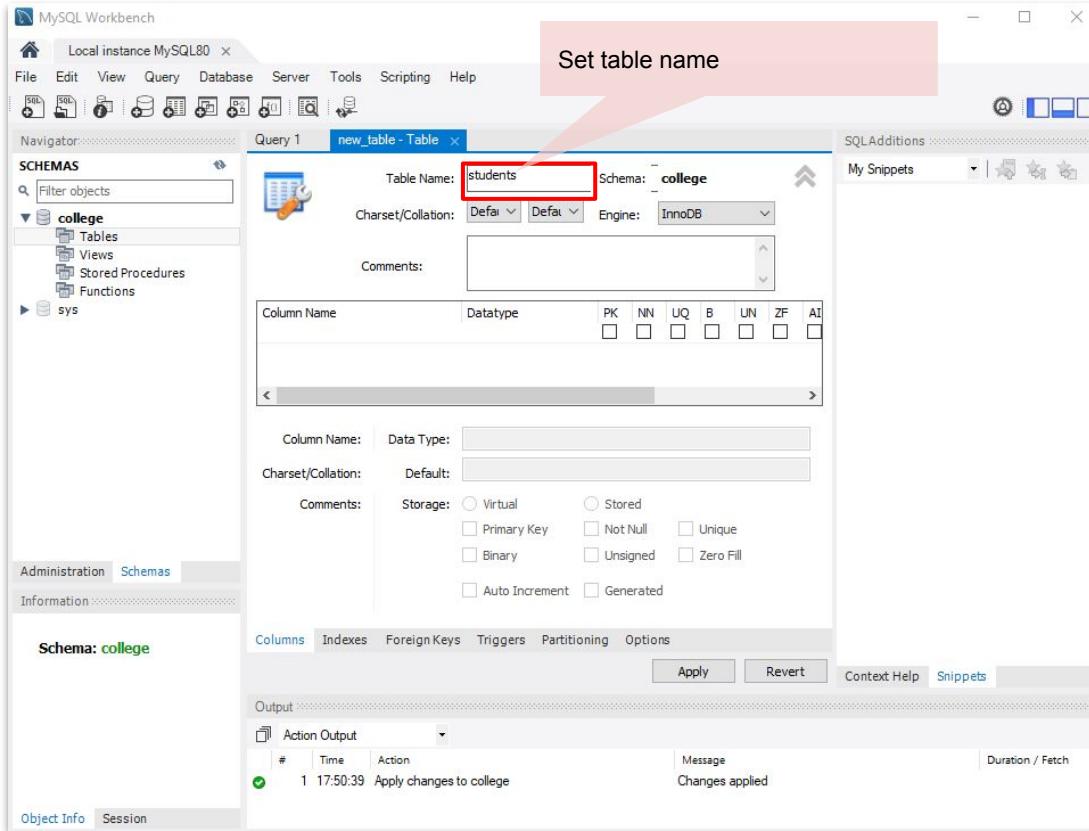


## Creating a Table

1. Click on the create new table button , to create a new table in the college database.
2. Set the table name, add columns, their datatypes, and constraints (i.e. set PK columns to Non-Null and Auto-Increment).

# Creating a Relational Database

Next, we create tables under the active database:

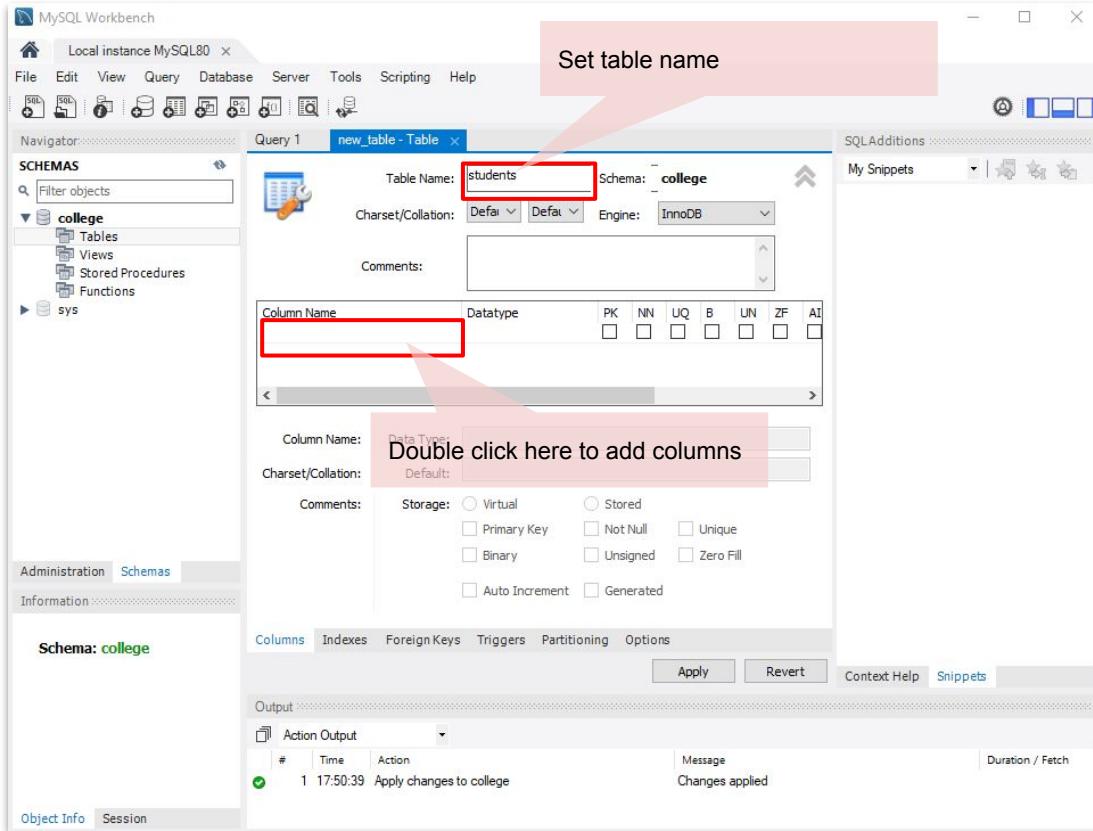


## Creating a Table

1. Click on the create new table button , to create a new table in the college database.
2. Set the table name, add columns, their datatypes, and constraints (i.e. set PK columns to Non-Null and Auto-Increment).

# Creating a Relational Database

Next, we create tables under the active database:

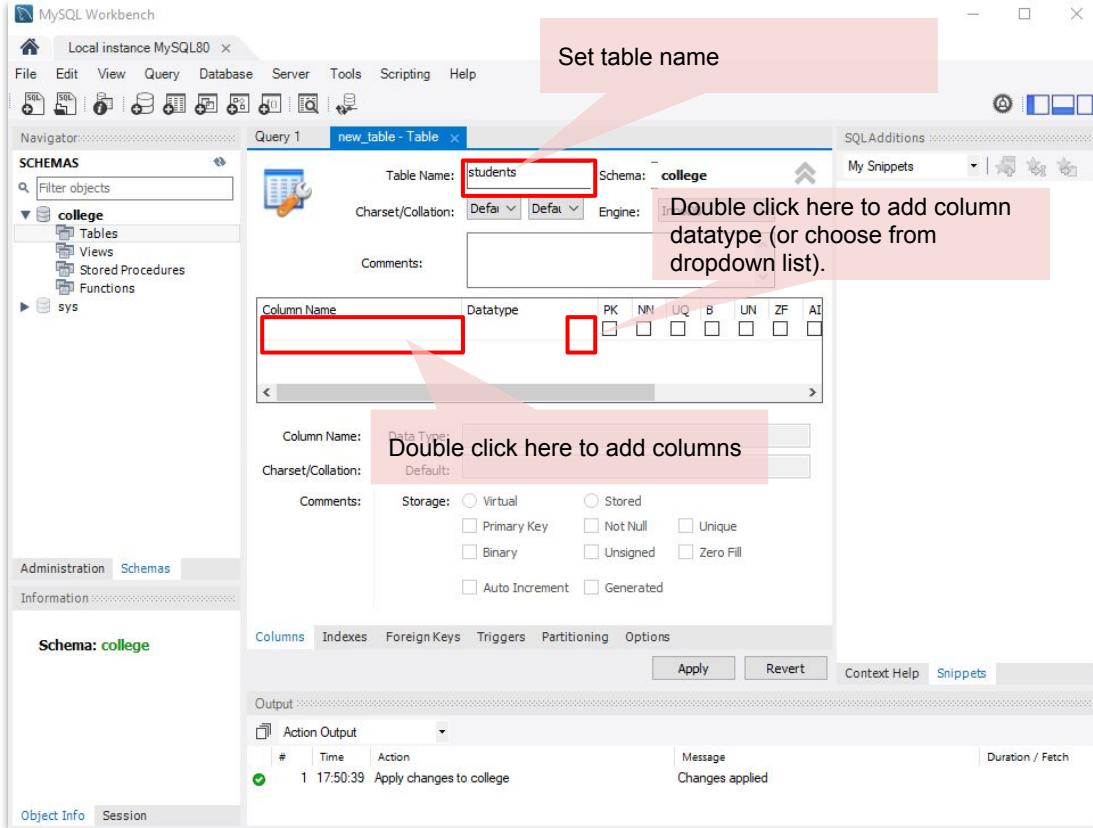


## Creating a Table

1. Click on the create new table button , to create a new table in the college database.
2. Set the table name, add columns, their datatypes, and constraints (i.e. set PK columns to Non-Null and Auto-Increment).

# Creating a Relational Database

Next, we create tables under the active database:



## Creating a Table

1. Click on the create new table button , to create a new table in the college database.
2. Set the table name, add columns, their datatypes, and constraints (i.e. set PK columns to Non-Null and Auto-Increment).

# Creating a Relational Database

Next, we create tables under the active database:

The screenshot shows the MySQL Workbench interface with the 'college' schema selected. A new table named 'students' is being created. The 'Columns' tab is selected, showing three columns: 'StudentID' (INT, PK, NN, AI), 'Name' (VARCHAR(100), NN), and 'Surname' (VARCHAR(100), NN). A callout box points to the primary key column settings, explaining that the primary key column must be set to PK - Primary key, NN - Non-Null, and AI - Auto Increment.

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI
StudentID	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>					<input checked="" type="checkbox"/>
Name	VARCHAR(100)		<input type="checkbox"/>					
Surname	VARCHAR(100)		<input type="checkbox"/>					

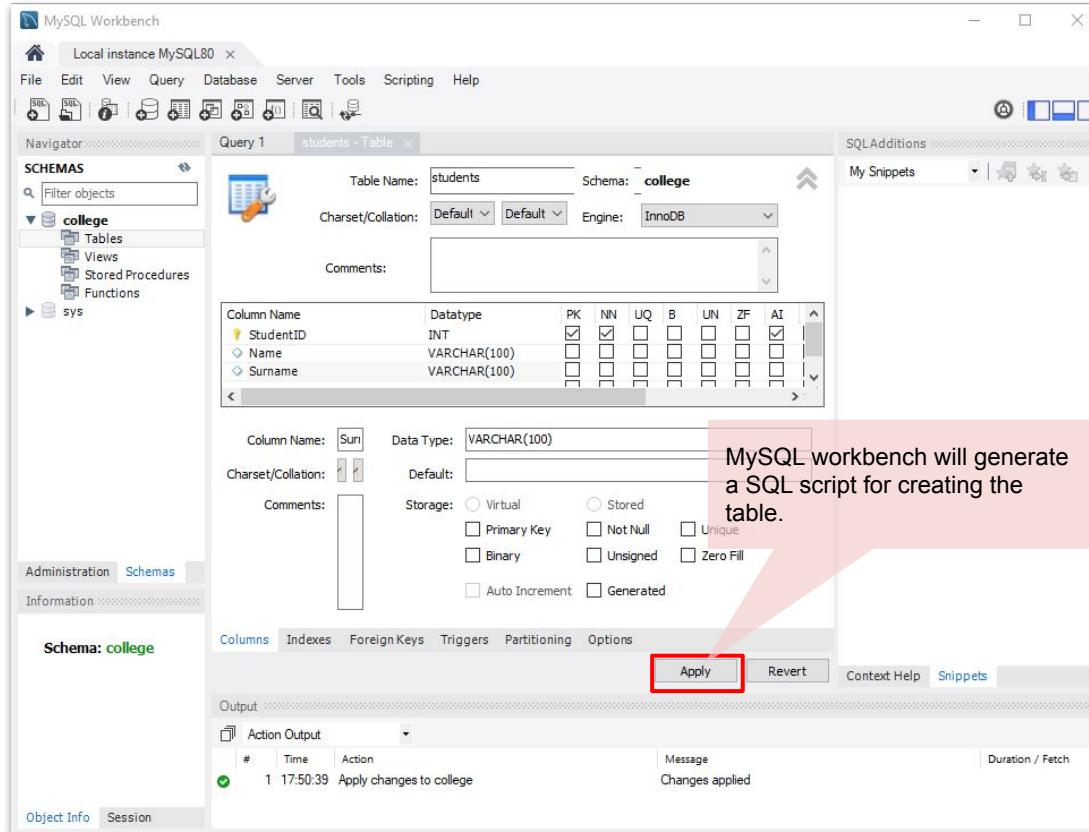
Note that the primary key column has to be set to  
**PK** - Primary key  
**NN** - Non-Null  
**AI** - Auto Increment

## Creating a Table

1. Click on the create new table button , to create a new table in the college database.
2. Set the table name, add columns, their datatypes, and constraints (i.e. set PK columns to Non-Null and Auto-Increment).

# Creating a Relational Database

Next, we create tables under the active database:

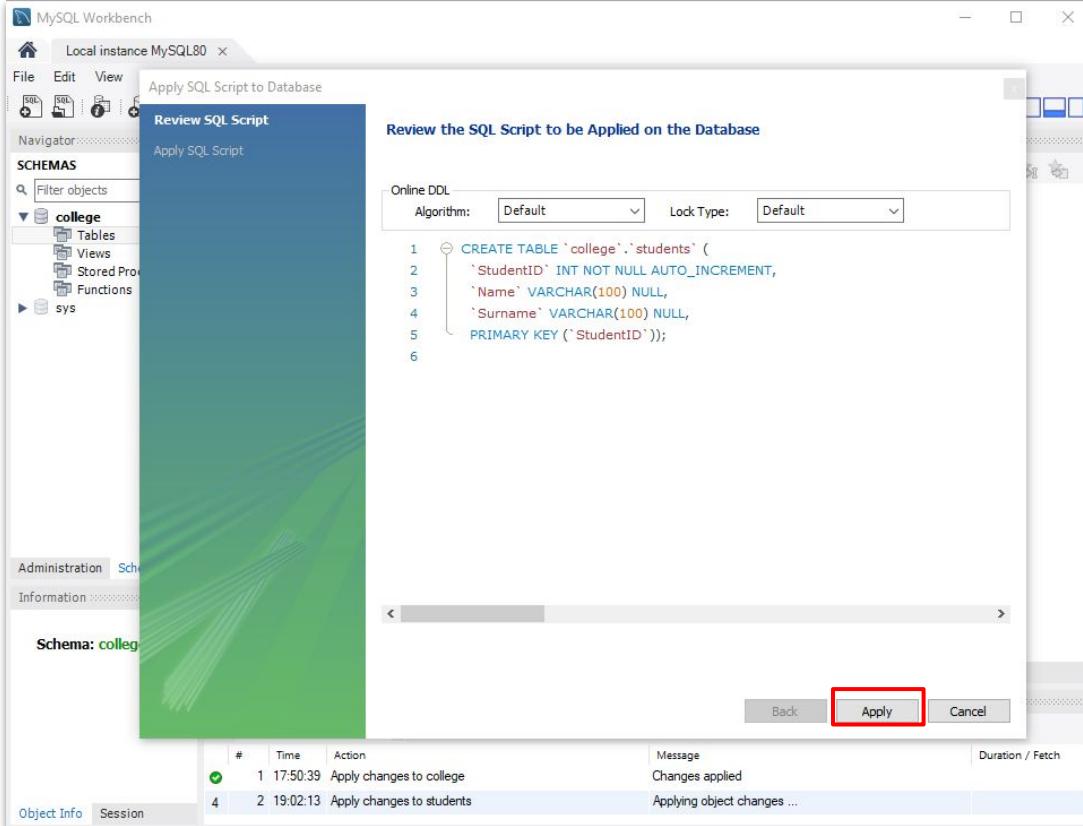


## Creating a Table

1. Click on the create new table button , to create a new table in the college database.
2. Set the table name, add columns, their datatypes, and constraints (i.e. set PK columns to Non-Null and Auto-Increment).
3. Click apply, then apply again on the wizard, and finish.

# Creating a Relational Database

Next, we create tables under the active database:

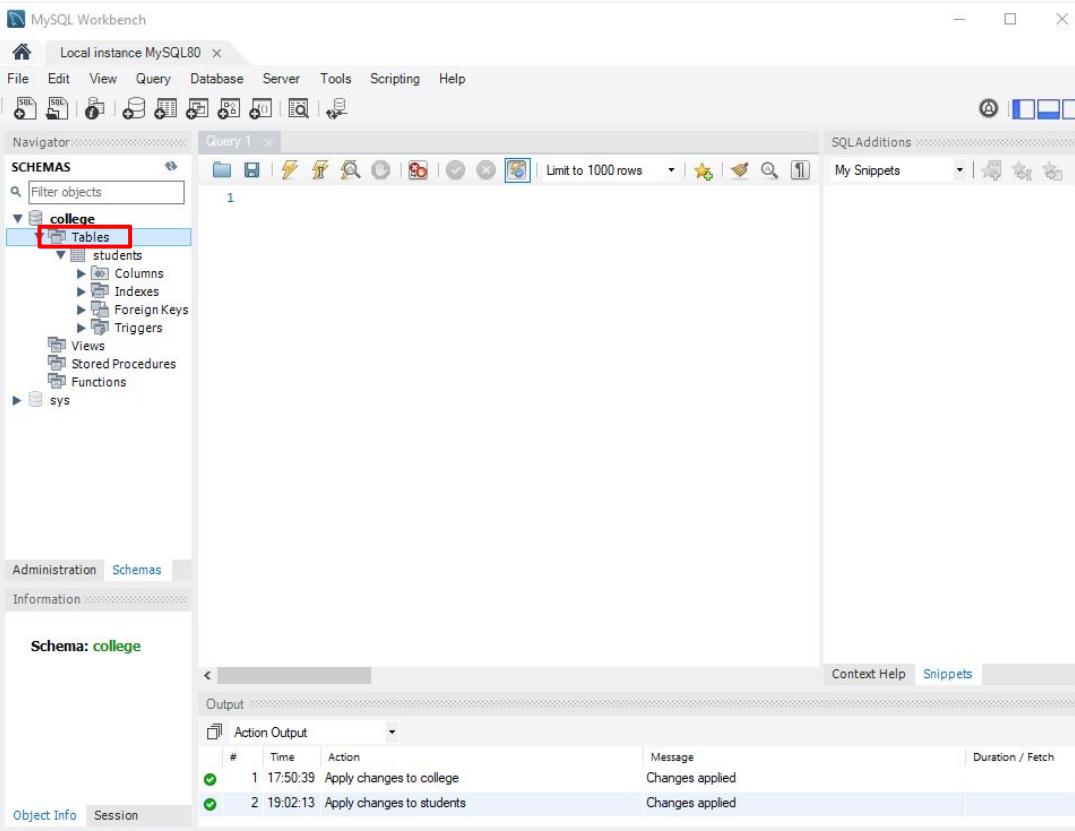


## Creating a Table

1. Click on the create new table button , to create a new table in the college database.
2. Set the table name, add columns, their datatypes, and constraints (i.e. set PK columns to Non-Null and Auto-Increment).
3. Click apply, then apply again on the wizard, and finish.

# Creating a Relational Database

Next, we create tables under the active database:

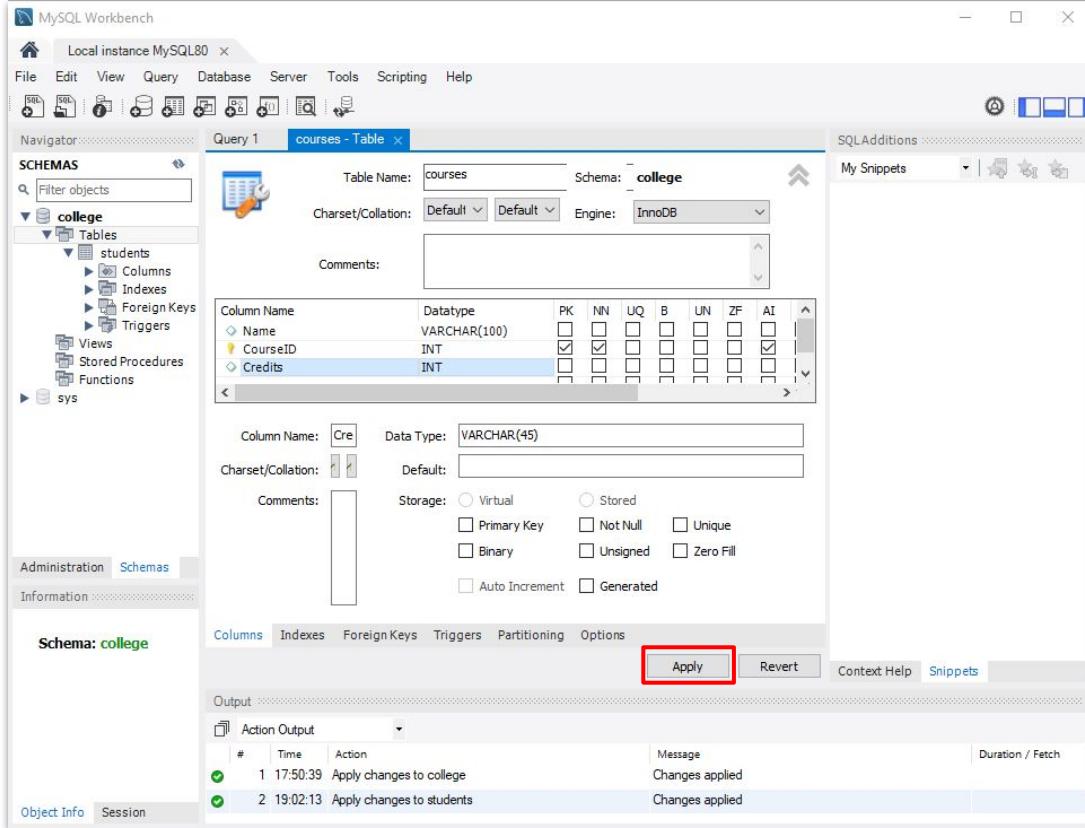


## Creating a Table

1. Click on the create new table button , to create a new table in the college database.
2. Set the table name, add columns, their datatypes, and constraints (i.e. set PK columns to Non-Null and Auto-Increment).
3. Click apply, then apply again on the wizard, and finish.
4. The new table can be viewed by clicking on the tables tab under the database name.

# Creating a Relational Database

Repeat the process to create other tables, let's do the **courses** table next:

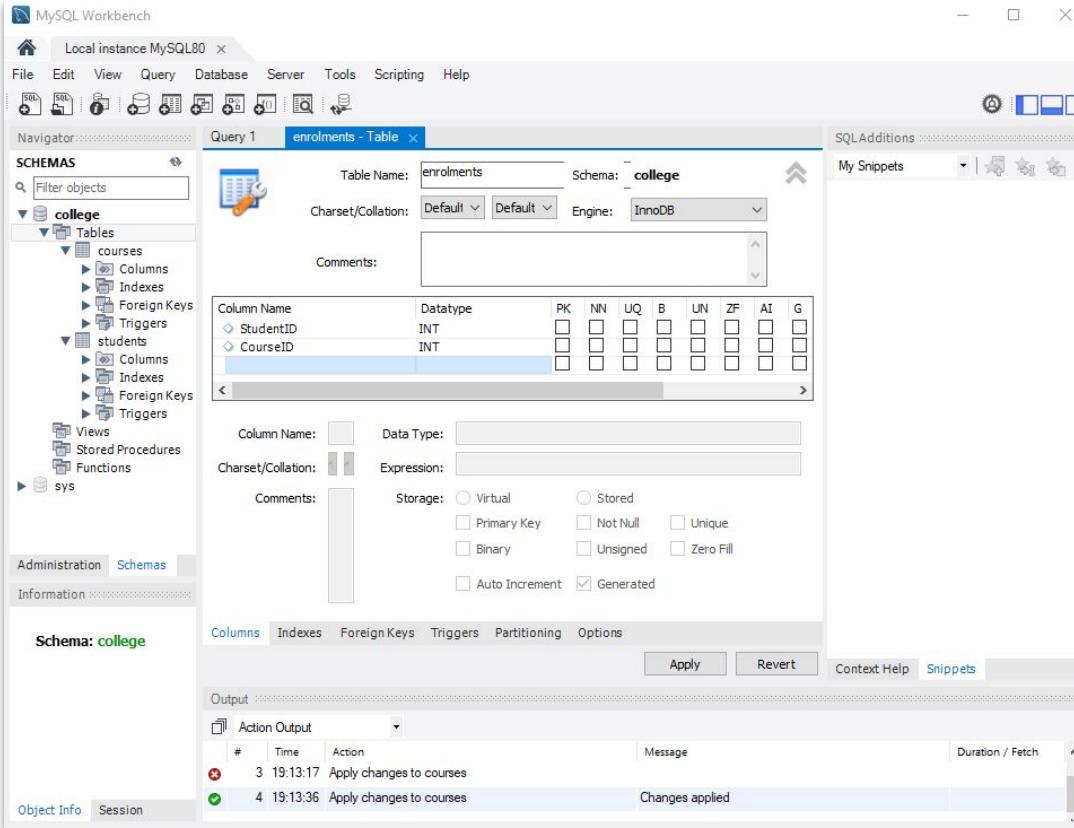


## Creating a Table

1. Click on the create new table button , to create a new table in the college database.
2. Set the table name, add columns, their datatypes, and constraints (i.e. set PK columns to Non-Null and Auto-Increment).
3. Click apply, then apply again on the wizard, and finish.
4. The new table can be viewed by clicking on the tables tab under the database name.

# Creating a Relational Database

Next, the **enrolments** table, the only difference here is that this table has foreign key columns:



## Foreign Key columns

1. Click on the create new table button , to create a new table in the college database.
2. Set the table name, add columns, their datatypes, and constraints (i.e. set PK columns to Non-Null and Auto-Increment).

# Creating a Relational Database

Next, the **enrolments** table, the only difference here is that this table has foreign key columns:

The screenshot shows the MySQL Workbench interface for creating a new table named 'enrolments' in the 'college' schema. The table has two columns: 'StudentID' (INT, PK, AI) and 'CourseID' (INT). The 'Foreign Keys' tab is selected at the bottom. The output pane shows the command 'Apply changes to courses' was successful.

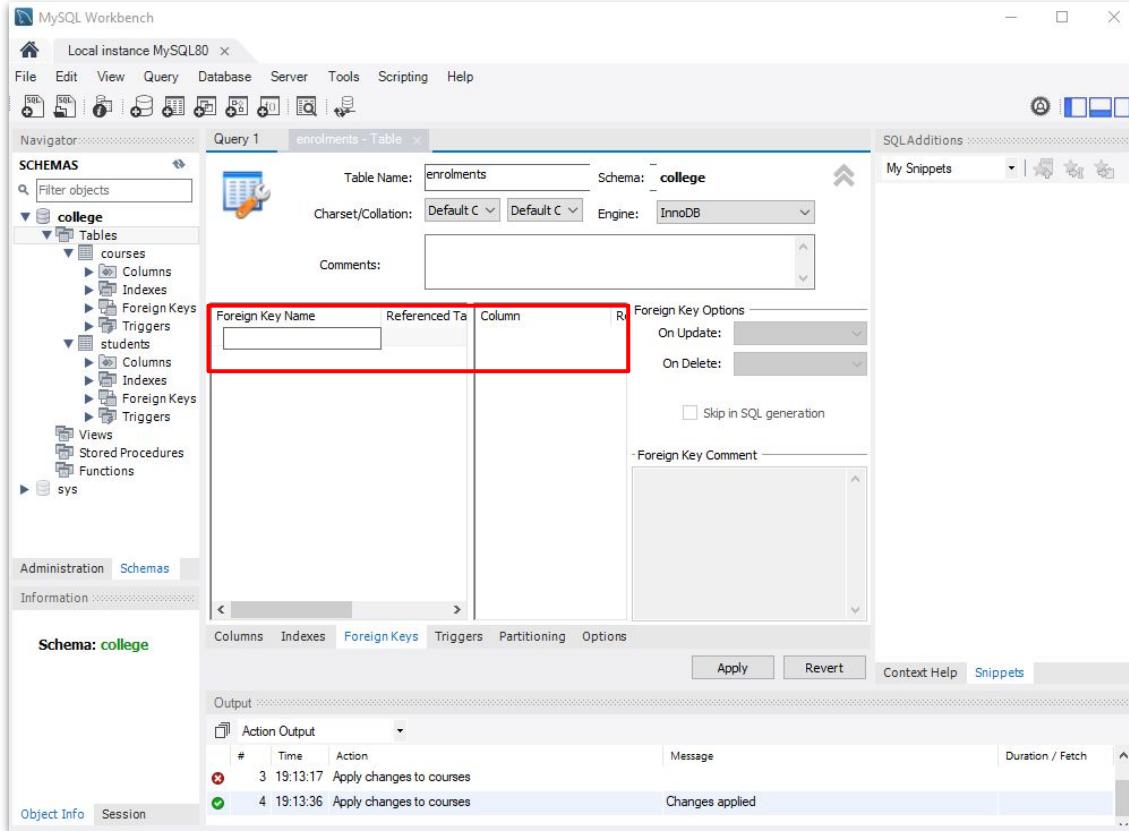
Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G
StudentID	INT	<input checked="" type="checkbox"/>	<input type="checkbox"/>						
CourseID	INT	<input checked="" type="checkbox"/>	<input type="checkbox"/>						

## Foreign Key columns

1. Click on the create new table button , to create a new table in the college database.
2. Set the table name, add columns, their datatypes, and constraints (i.e. set PK columns to Non-Null and Auto-Increment).
3. To add Foreign key columns, navigate to the foreign key tab.

# Creating a Relational Database

Next, the **enrolments** table, the only difference here is that this table has foreign key columns:



## Foreign Key columns

1. Click on the create new table button , to create a new table in the college database.
2. Set the table name, add columns, their datatypes, and constraints (i.e. set PK columns to Non-Null and Auto-Increment).
3. To add Foreign key columns, navigate to the foreign key tab.
4. Set Foreign key column name, the table it references, and the referenced table column.

# Creating a Relational Database

Next, the **enrolments** table, the only difference here is that this table has foreign key columns:

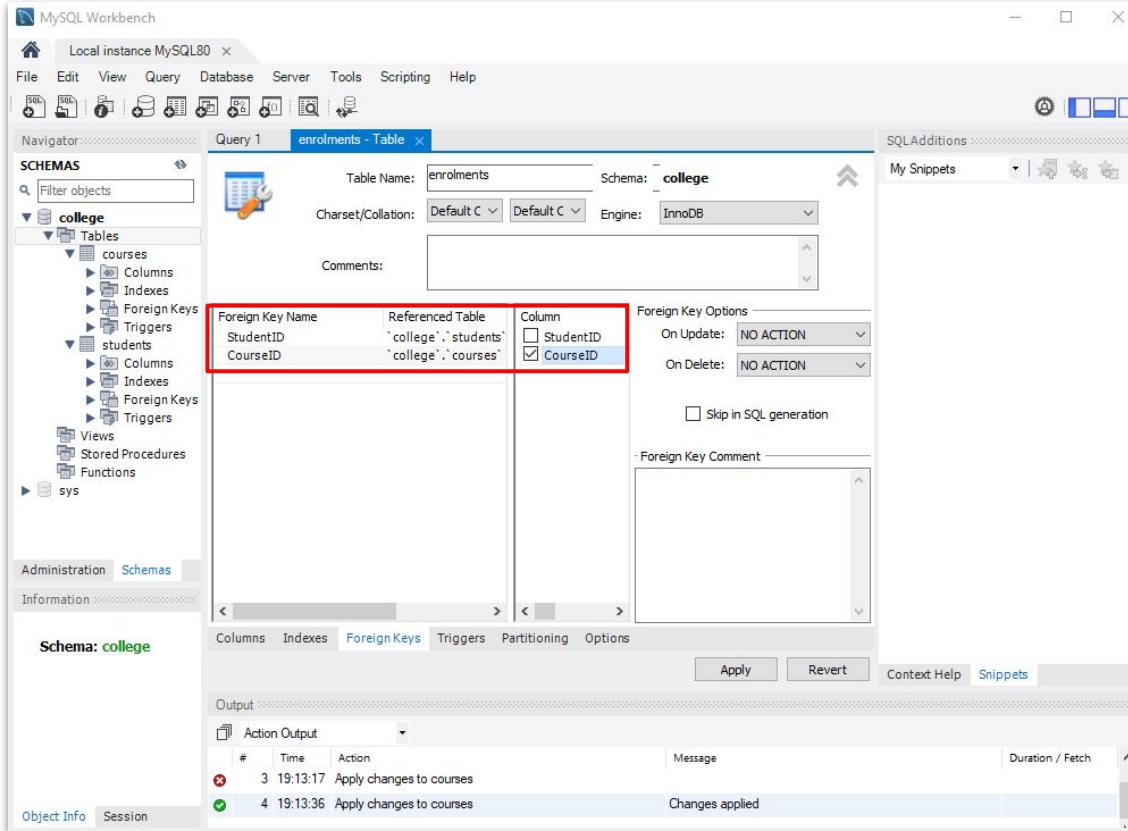
The screenshot shows the MySQL Workbench interface for creating a new table named 'enrolments' in the 'college' schema. The 'Foreign Keys' tab is selected. A red box highlights the 'Foreign Key Options' section where 'StudentID' is defined as a foreign key referencing the 'students' table's 'StudentID' column. The 'On Update' and 'On Delete' dropdowns both show 'NO ACTION'. The 'Skip in SQL generation' checkbox is unchecked. The 'Foreign Key Comment' field is empty.

## Foreign Key columns

1. Click on the create new table button , to create a new table in the college database.
2. Set the table name, add columns, their datatypes, and constraints (i.e. set PK columns to Non-Null and Auto-Increment).
3. To add Foreign key columns, navigate to the foreign key tab.
4. Set Foreign key column name, the table it references, and the referenced table column.

# Creating a Relational Database

Next, the **enrolments** table, the only difference here is that this table has foreign key columns:

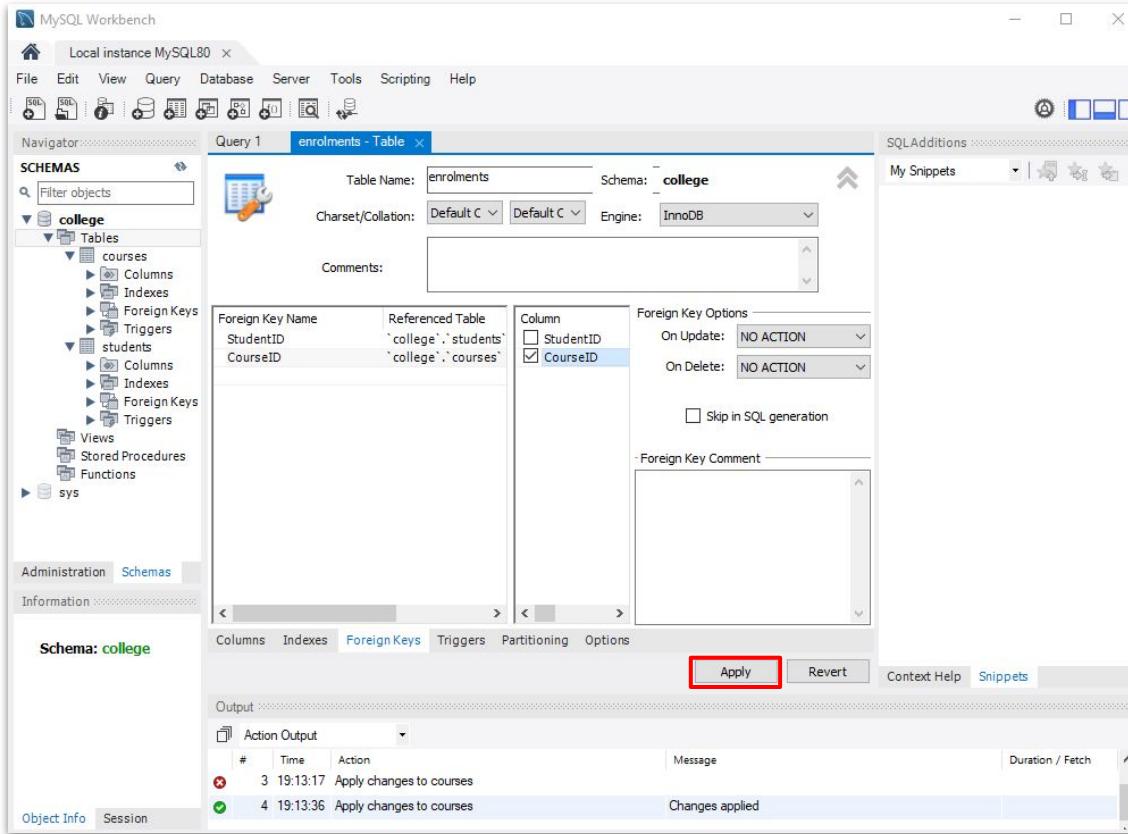


## Foreign Key columns

1. Click on the create new table button , to create a new table in the college database.
2. Set the table name, add columns, their datatypes, and constraints (i.e. set PK columns to Non-Null and Auto-Increment).
3. To add Foreign key columns, navigate to the foreign key tab.
4. Set Foreign key column name, the table it references, and the referenced table column.

# Creating a Relational Database

Next, the **enrolments** table, the only difference here is that this table has foreign key columns:

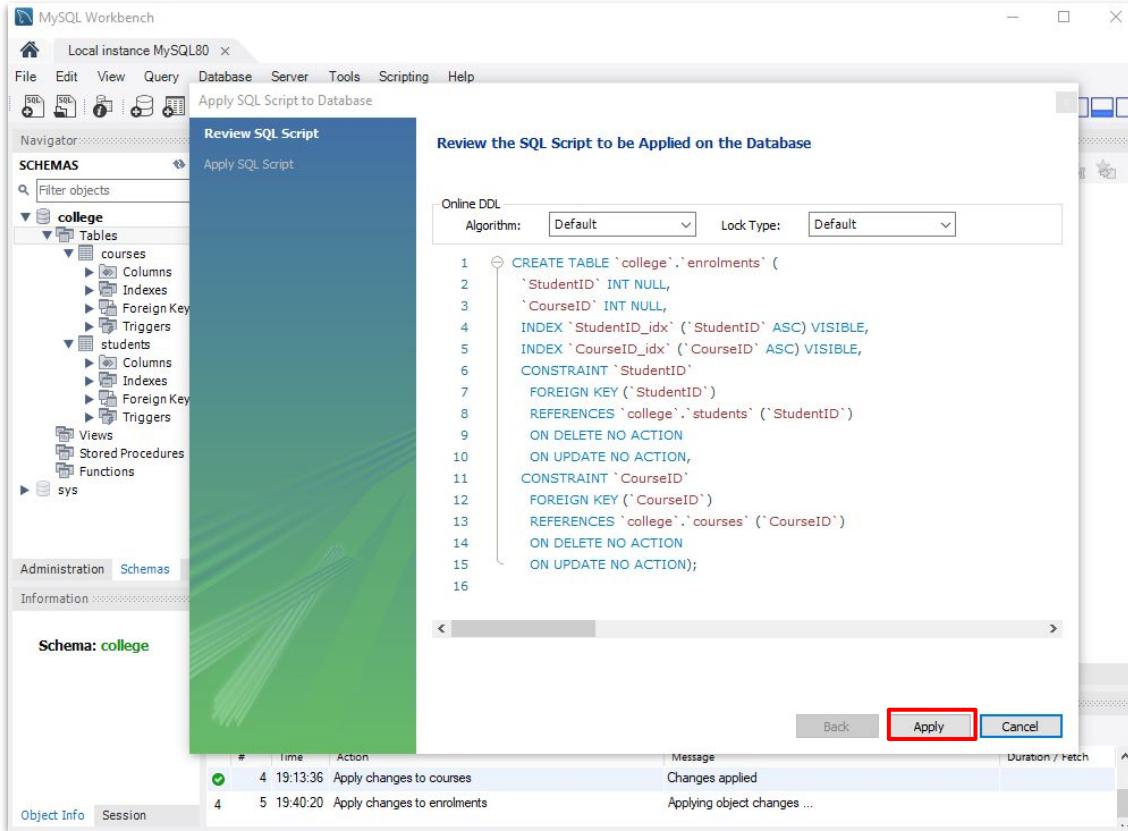


## Foreign Key columns

1. Click on the create new table button , to create a new table in the college database.
2. Set the table name, add columns, their datatypes, and constraints (i.e. set PK columns to Non-Null and Auto-Increment).
3. To add Foreign key columns, navigate to the foreign key tab.
4. Set Foreign key column name, the table it references, and the referenced table column.
5. Click apply, then apply again on the wizard, and finish.

# Creating a Relational Database

Next, the **enrolments** table, the only difference here is that this table has foreign key columns:

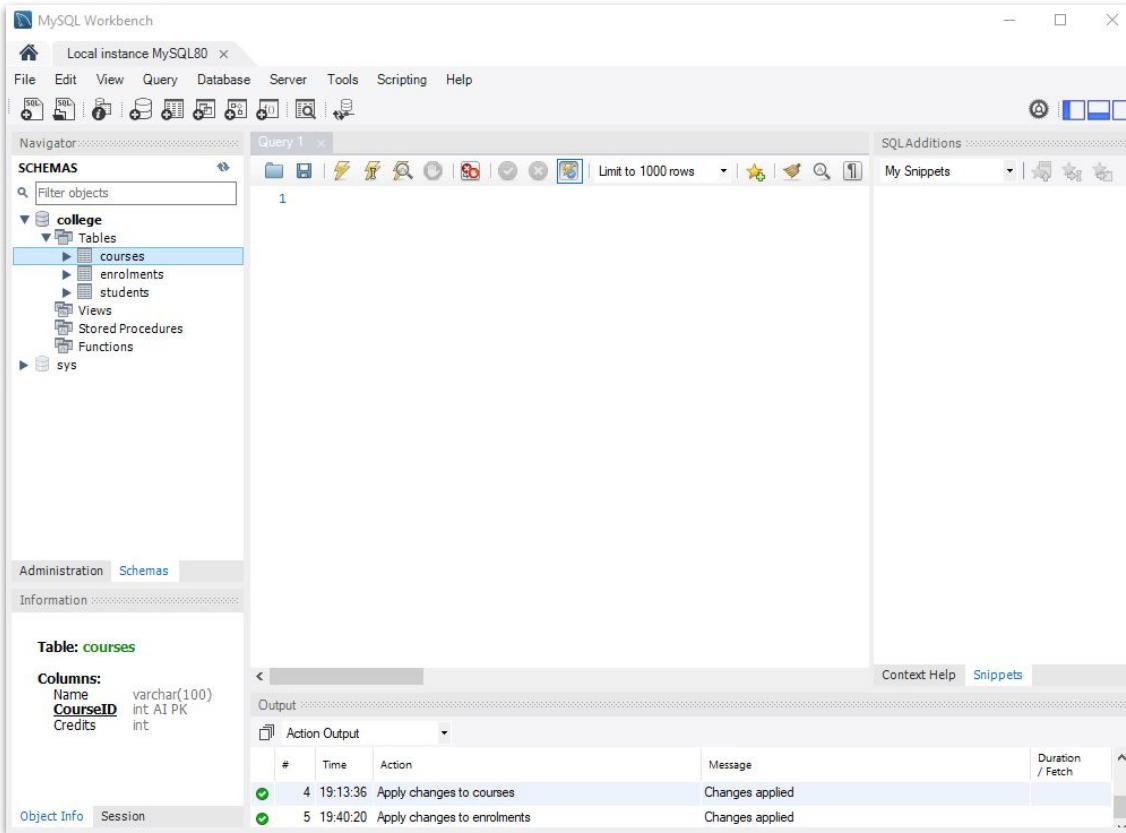


## Foreign Key columns

1. Click on the create new table button , to create a new table in the college database.
2. Set the table name, add columns, their datatypes, and constraints (i.e. set PK columns to Non-Null and Auto-Increment).
3. To add Foreign key columns, navigate to the foreign key tab.
4. Set Foreign key column name, the table it references, and the referenced table column.
5. Click apply, then apply again on the wizard, and finish.

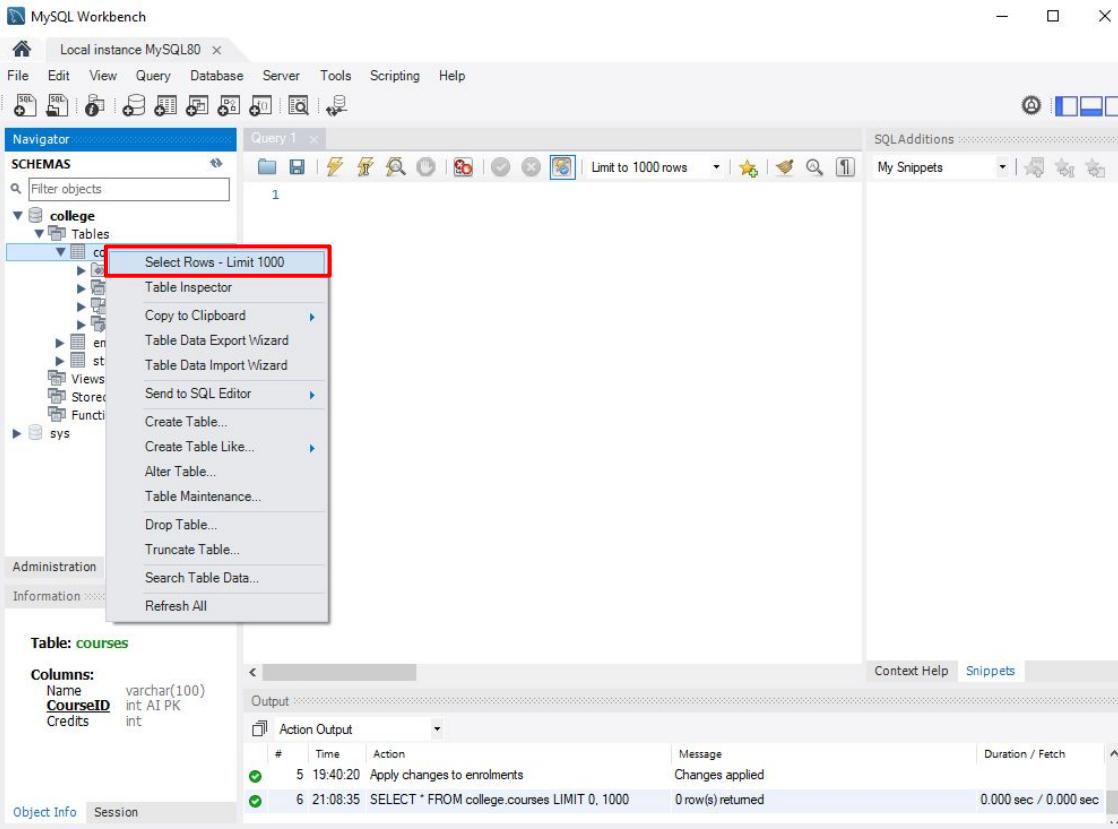
# Creating a Relational Database

At this we have successfully created our database, the last step here, is to **add data**:



# Creating a Relational Database

At this we have successfully created our database, the last step here, is to **add data**:



## Adding data to tables

1. Right click on the table name and click **select rows**.

# Creating a Relational Database

At this we have successfully created our database, the last step here, is to **add data**:

The screenshot shows the MySQL Workbench interface. In the top-left, the 'Schemas' tree shows a 'college' schema containing a 'courses' table. A query window titled 'Query 1' displays the result of the SQL command: 'SELECT \* FROM college.courses;'. The result grid shows three columns: 'Name', 'CourseID', and 'Credits'. All three rows in the grid are highlighted with a red border. Below the grid, the 'Object Info' tab is selected, showing the table structure: Name (varchar(100)), CourseID (int AI PK), and Credits (int). The bottom status bar indicates two recent actions: 'SELECT \* FROM college.courses LIMIT 0, 1000' at 21:08:35 and 'SELECT \* FROM college.courses LIMIT 0, 1000' at 21:14:29.

## Adding data to tables

1. Right click on the table name and click **select rows**.
2. In the result grid, double click on a row and column to add a value.

# Creating a Relational Database

At this we have successfully created our database, the last step here, is to **add data**:

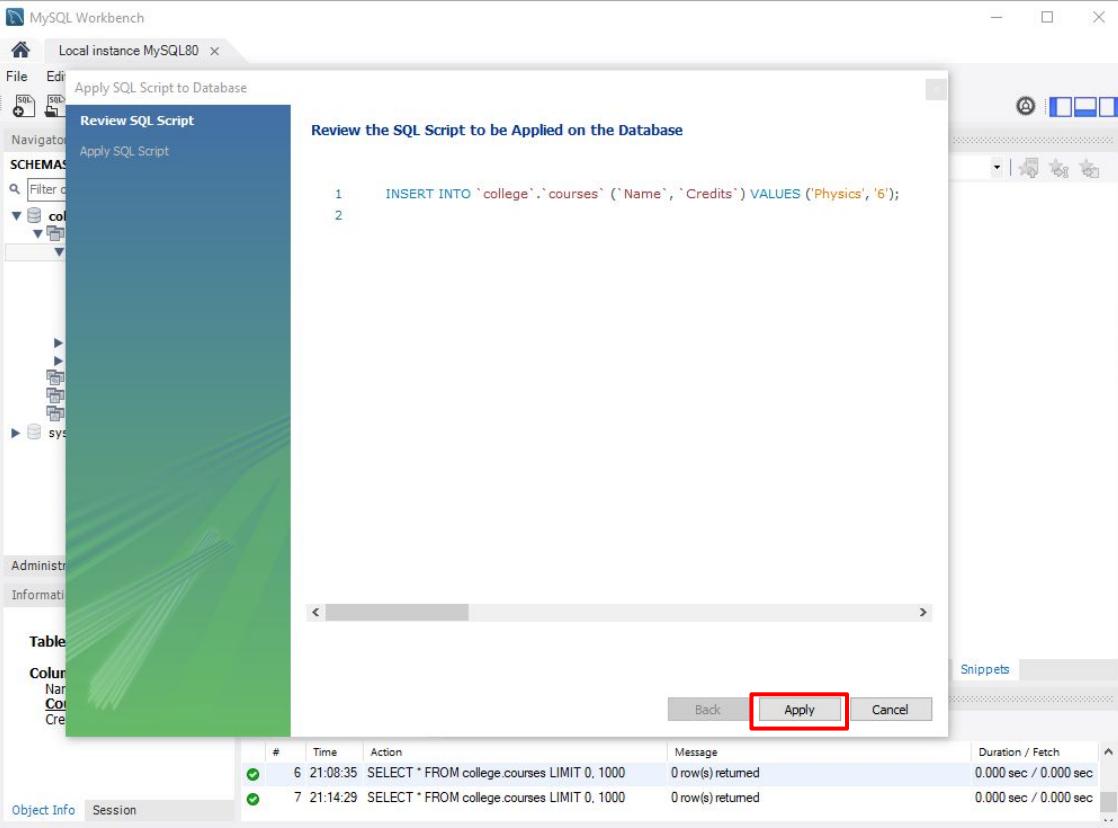
The screenshot shows the MySQL Workbench interface. In the Navigator pane, under the 'college' schema, the 'courses' table is selected. The 'Result Grid' shows one row of data: Name (Physics), CourseID (NULL), and Credits (6). The 'SQL Editor' pane contains the query: `1 • SELECT * FROM college.courses;`. The bottom status bar shows two recent queries: # 6 21:08:35 and # 7 21:14:29.

## Adding data to tables

1. Right click on the table name and click **select rows**.
2. In the result grid, double click on a row and column to add a value.
3. After adding all required column values (with the exception of **auto-incrementing** primary key columns), click **apply**.

# Creating a Relational Database

At this we have successfully created our database, the last step here, is to **add data**:



## Adding data to tables

1. Right click on the table name and click **select rows**.
2. In the result grid, double click on a row and column to add a value.
3. After adding all required column values (with the exception of **auto-incrementing** primary key columns), click **apply**.
4. Click **apply**.

# Creating a Relational Database

At this we have successfully created our database, the last step here, is to **add data**:

The screenshot shows the MySQL Workbench interface. In the top-left, the Navigator pane displays the 'college' schema with its tables: 'courses', 'enrolments', 'students', 'Views', 'Stored Procedures', and 'Functions'. The 'sys' table is also listed. In the center, a 'Query 1' tab is open with the SQL command: 'SELECT \* FROM college.courses;'. The results are displayed in a 'Result Grid' table:

Name	CourseID	Credits
Physics	1	6

A red box highlights the first row of the grid. A callout bubble points to this row with the text: 'Even though the primary key column was not specified, a value has been auto-generated.' At the bottom of the interface, the 'Object Info' and 'Session' tabs are visible.

## Adding data to tables

1. Right click on the table name and click **select rows**.
2. In the result grid, double click on a row and column to add a value.
3. After adding all required column values (with the exception of **auto-incrementing** primary key columns), click **apply**.
4. Click **apply**.

# Creating a Relational Database

At this we have successfully created our database, the last step here, is to **add data**:

The screenshot shows the MySQL Workbench interface. In the Navigator pane, under the 'college' schema, the 'courses' table is selected. A red box highlights the first row of the 'Result Grid' which contains the values: Name (Physics), CourseID (1), and Credits (6). A callout bubble points to this row with the text: "Even though the primary key column was not specified, a value has been auto-generated." Below the grid, the 'Table: courses' section shows the columns: Name, CourseID (with a note 'int AI PK'), and Credits. The 'Columns' table shows the definitions: Name (varchar(100)), CourseID (int AI PK), and Credits (int). The 'Output' pane at the bottom shows two rows of action output corresponding to the SELECT query results.

Name	CourseID	Credits
Physics	1	6

Table: courses

Columns:

Name	Type	Notes
Name	varchar(100)	
<u>CourseID</u>	int	AI PK
Credits	int	

Output:

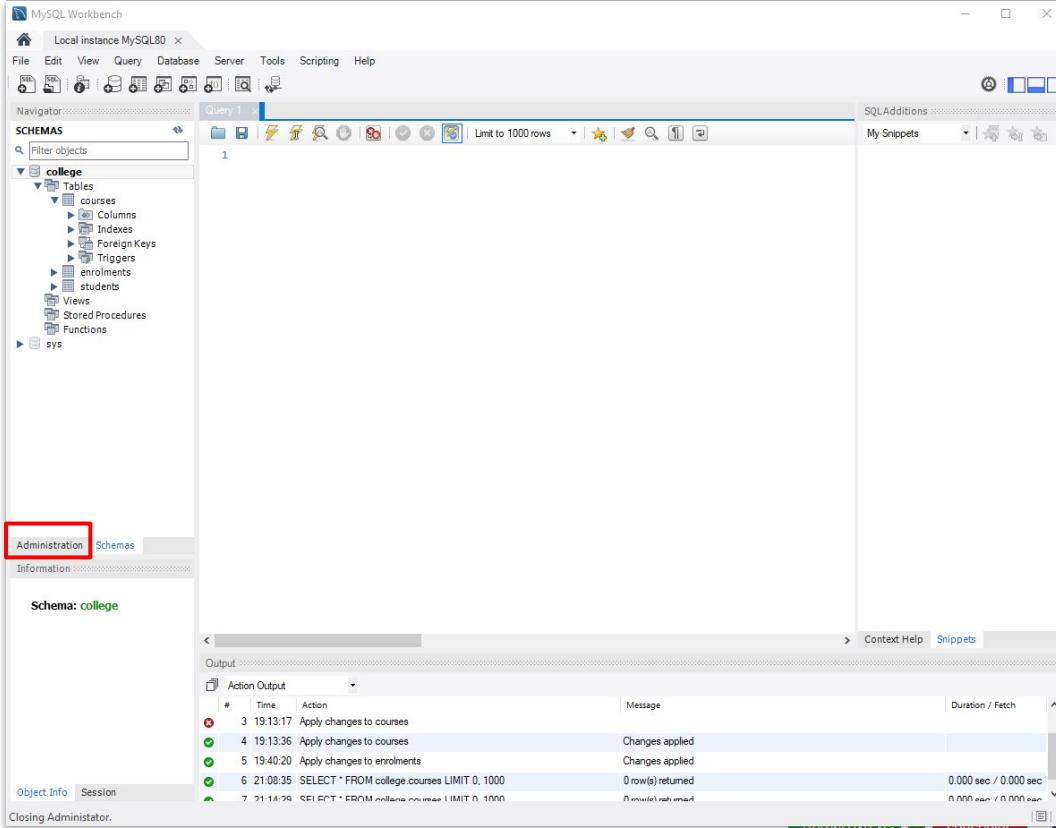
#	Time	Action	Message	Duration / Fetch
6	21:08:35	SELECT * FROM college.courses LIMIT 0, 1000	0 row(s) returned	0.000 sec / 0.000 sec
7	21:14:29	SELECT * FROM college.courses LIMIT 0, 1000	0 row(s) returned	0.000 sec / 0.000 sec

## Adding data to tables

1. Right click on the table name and click **select rows**.
2. In the result grid, double click on a row and column to add a value.
3. After adding all required column values (with the exception of **auto-incrementing** primary key columns), click **apply**.  
Click **apply**.
4. Repeat process to add data to other tables.

# Creating a Relational Database

To save the database to a file, we use the **Data Export** wizard in the administration wizard:

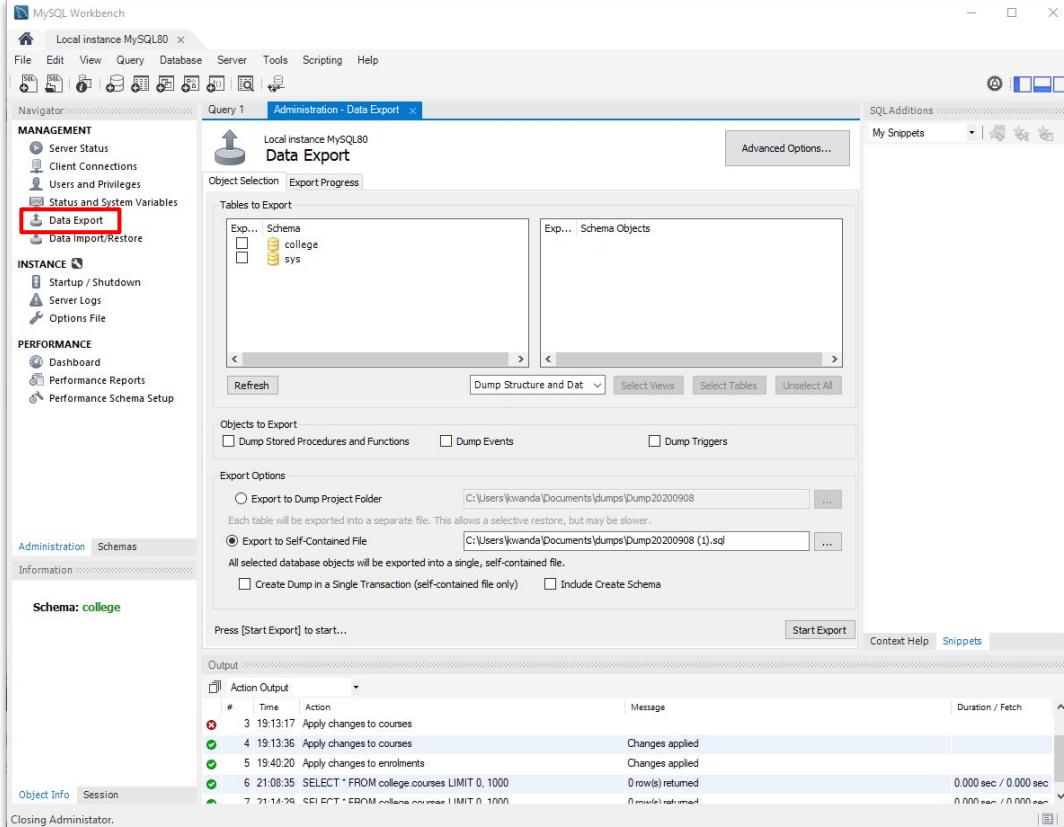


## Saving the DB to file

1. Click on the administration tab.

# Creating a Relational Database

To save the database to a file, we use the **Data Export** wizard in the administration wizard:

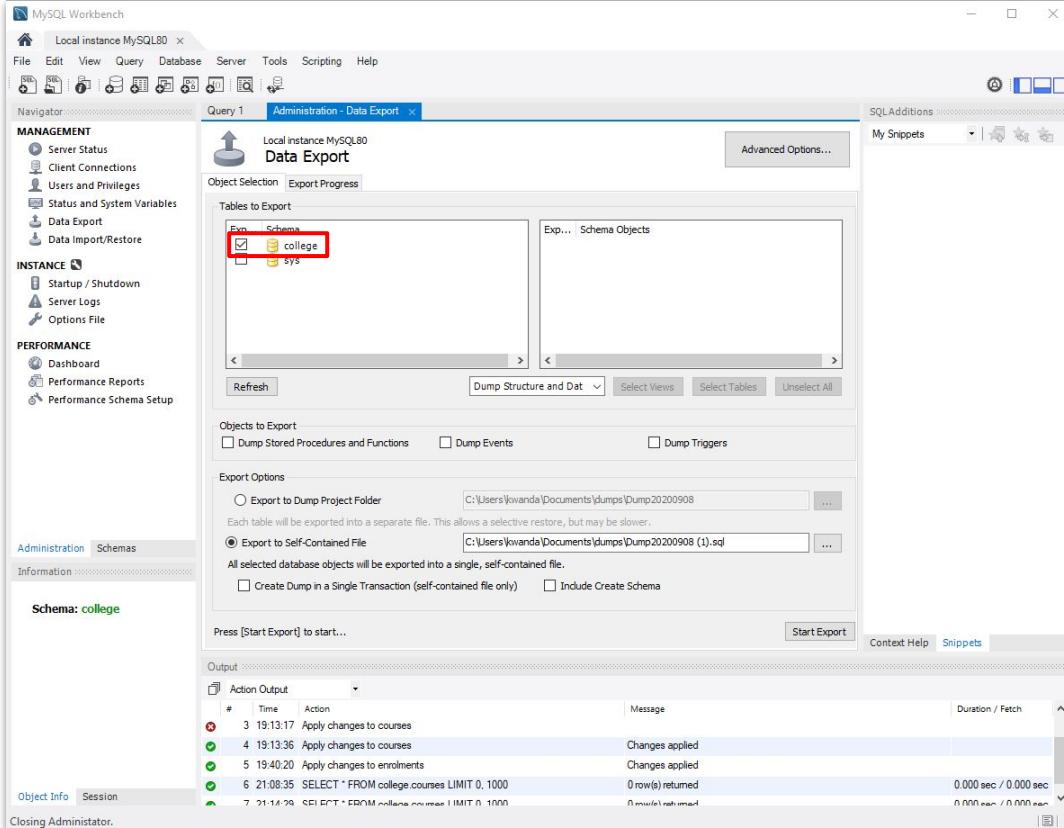


## Saving the DB to file

1. Click on the administration tab.
2. Select **Data Export**.

# Creating a Relational Database

To save the database to a file, we use the **Data Export** wizard in the administration wizard:

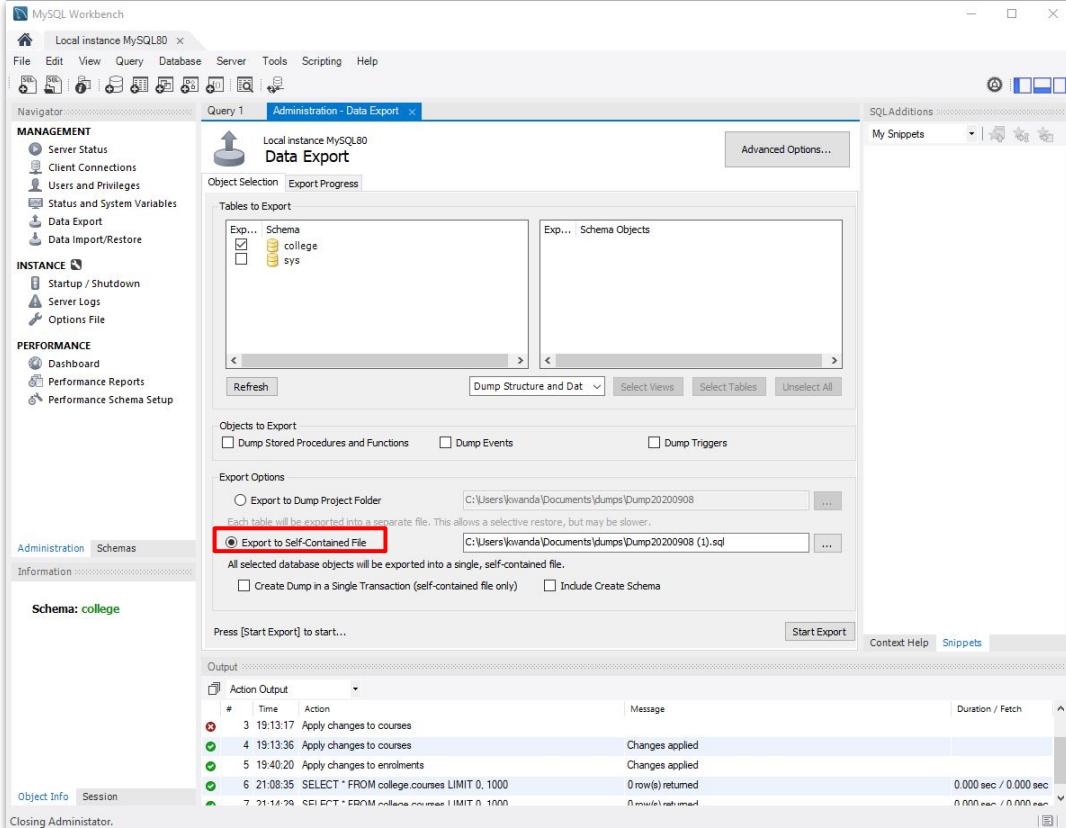


## Saving the DB to file

1. Click on the administration tab.
2. Select **Data Export**.
3. Select database to export.

# Creating a Relational Database

To save the database to a file, we use the **Data Export** wizard in the administration wizard:

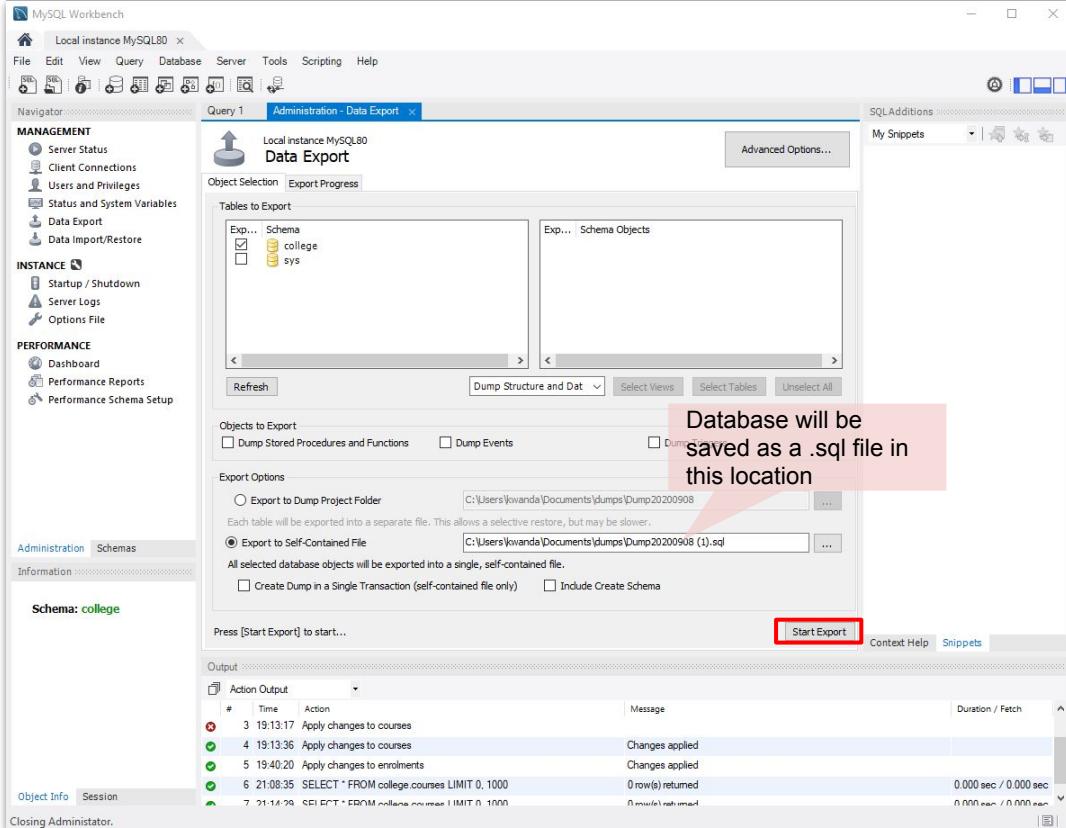


## Saving the DB to file

1. Click on the administration tab.
2. Select **Data Export**.
3. Select database to export.
4. Select “Export to self-contained file”

# Creating a Relational Database

To save the database to a file, we use the **Data Export** wizard in the administration wizard:

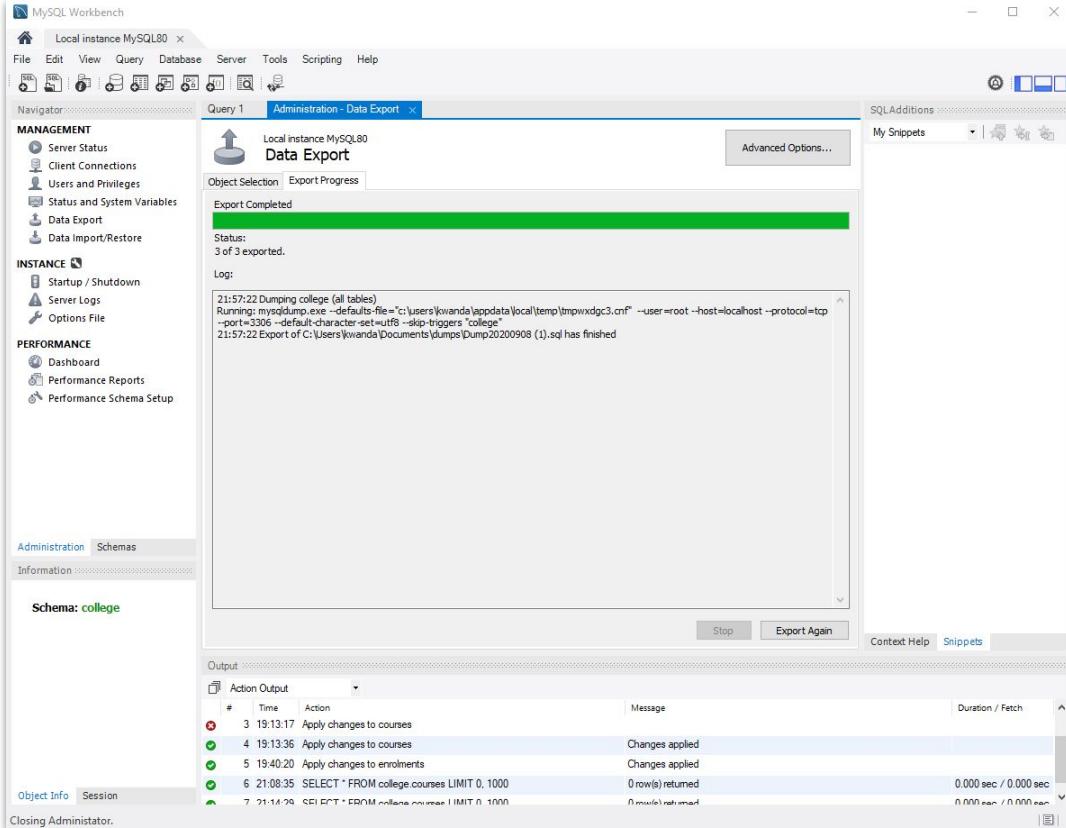


## Saving the DB to file

1. Click on the administration tab.
2. Select **Data Export**.
3. Select database to export.
4. Select “Export to self-contained file”
5. Click the **Start Export** button.

# Creating a Relational Database

To save the database to a file, we use the **Data Export** wizard in the administration wizard:



## Saving the DB to file

1. Click on the administration tab.
2. Select **Data Export**.
3. Select database to export.
4. Select “Export to self-contained file”
5. Click the **Start Export** button.

The exported .sql file can be used as a backup file that can be reloaded into MySQL or a different database management system.

# Conclusion

In this tutorial, we learnt how to:

- Install and configure MySQL workbench;
- Connect to a running MySQL server;
- Create a simple database;
- Create and connect tables using primary/foreign key relationships; and
- Backup the simple database.

