Stockholms universitet

Introduction to MySQL Community Edition

Course: SUPCOM, Hösten 2019

Module: Database Management Systems

DBMS Tool: MySQL version 8.0.18

Goal

- To familiarize yourself with basic MySQL Workbench (GUI)

Learning outcome

- Ability to navigate and use Workbench
- Ability to create database (schema), tables, constraints
- Ability to use DDL and DML

Part one – Introduction

MySQL Community with Workbench

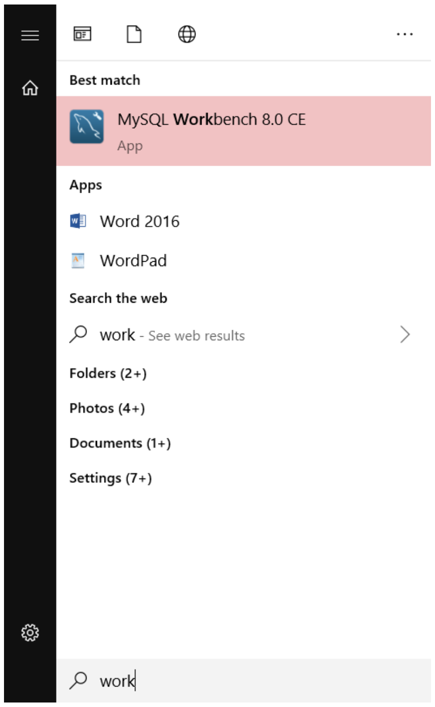
1. Introduction

MySQL is a database server claimed to be very fast, multithreaded, multi-user, and robust SQL (Structured Query Language) server according to MySQL™. MySQL is a trademark of Oracle Corporation and its affiliates. The MySQL software has Dual Licenses[1] therefore users can use the open source licensed version under the terms of the GNU General Public License[2] (Community Server) or can purchase a standard commercial license from Oracle (Enterprise Edition).

MySQL Workbench provides a graphical tool for working with MySQL servers and databases. MySQL Workbench fully supports MySQL versions 5.5 and higher[3]. MySQL Workbench provides functionalities to enable SQL Development, Data Modeling, Server Administration, Data Migration, and MySQL Enterprise Support for backup and audit.

Starting MySQL Workbench

- Click on **Start** menu then enter or type MySQL
- Then the **Start** menu displays **MySQL Workbench 8.0 CE** as illustrated below
- Launch MySQL by selecting **MySQL Workbench**

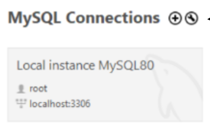


Creating connections to MySQL Server locally

- Start MySQL Workbench

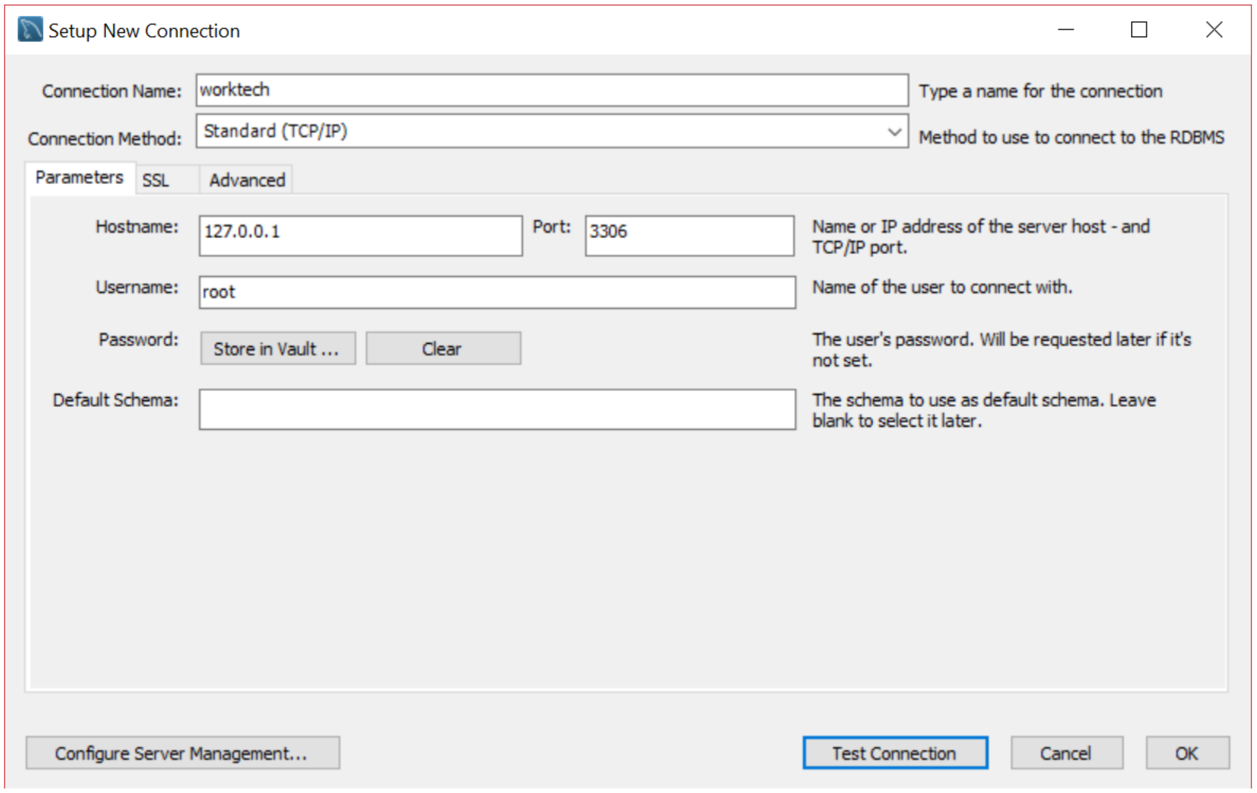
[1] http://www.mysql.com/company/legal/licensing/
[2] http://www.fsf.org/licenses/
[3] https://dev.mysql.com/doc/refman/8.0/en/workbench.html

- Click on the + sign next to MySQL Connections as illustrated here



- Enter connection name – here **worktech** is connection name and do not change anything else as illustrated below
- Click on **OK**

  Note: (*Remember that your MySQL server is installed locally, so under hostname loop back IP address is specified 127.0.0.1, if you want to connect your Workbench to any MySQL server you can enter fully qualified address for example (mysql.dsv.su.se)*)



Setup New Connection

Connection Name: worktech — Type a name for the connection

Connection Method: Standard (TCP/IP) — Method to use to connect to the RDBMS

Hostname: 127.0.0.1 Port: 3306 — Name or IP address of the server host - and TCP/IP port.

Username: root — Name of the user to connect with.

Password: — The user's password. Will be requested later if it's not set.

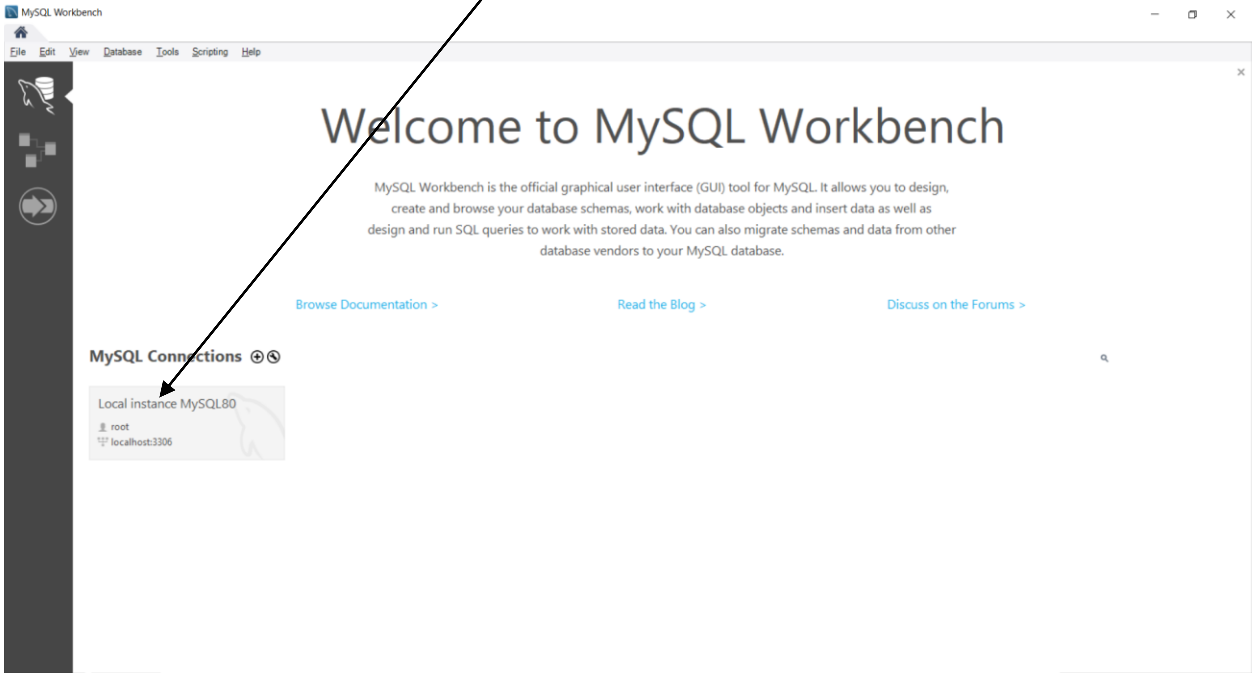
Default Schema: — The schema to use as default schema. Leave blank to select it later.

- Now your new connection is created so you can simply click on it to connect to your server and enter your password if you are prompted.

Connecting to your MySQL server

(*you can have several connections here*)

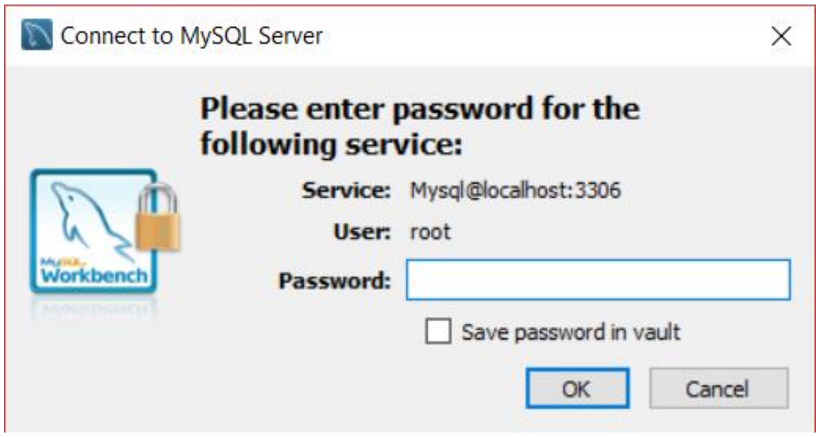
- Select the available connection: Local instance MySQL80
- Enter the password (supcom) then click **OK**
- MySQL will be opened select the connection already created as illustrated below.



Welcome to MySQL Workbench

MySQL Workbench is the official graphical user interface (GUI) tool for MySQL. It allows you to design, create and browse your database schemas, work with database objects and insert data as well as design and run SQL queries to work with stored data. You can also migrate schemas and data from other database vendors to your MySQL database.

- Enter your password (supcom) and click **OK**



Please enter password for the following service:

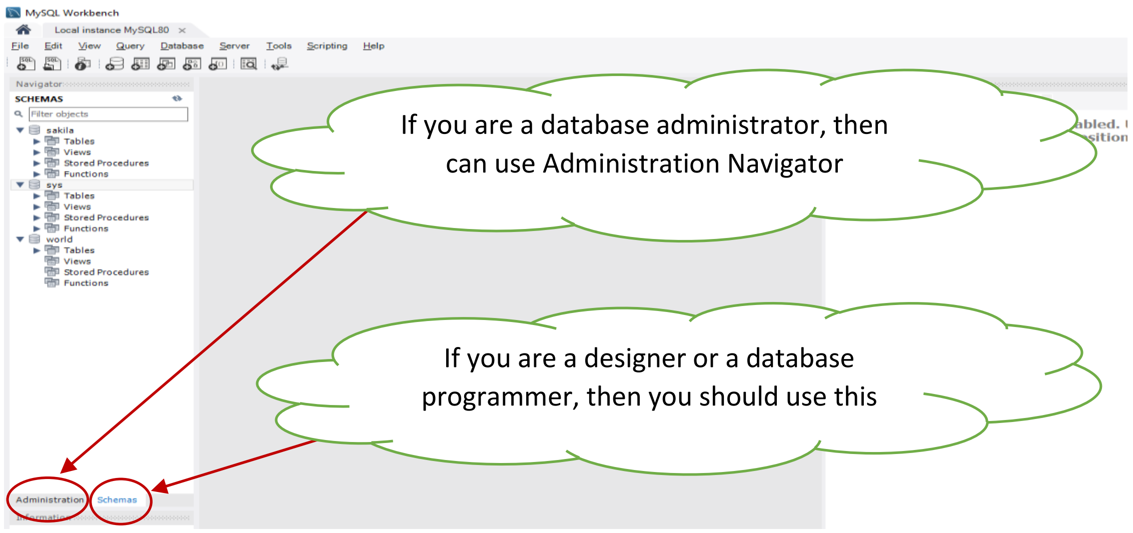
Service: Mysql@localhost:3306

User: root

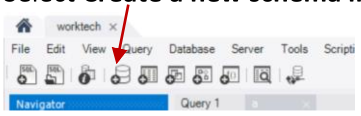
Password:

Creating and deleting databases

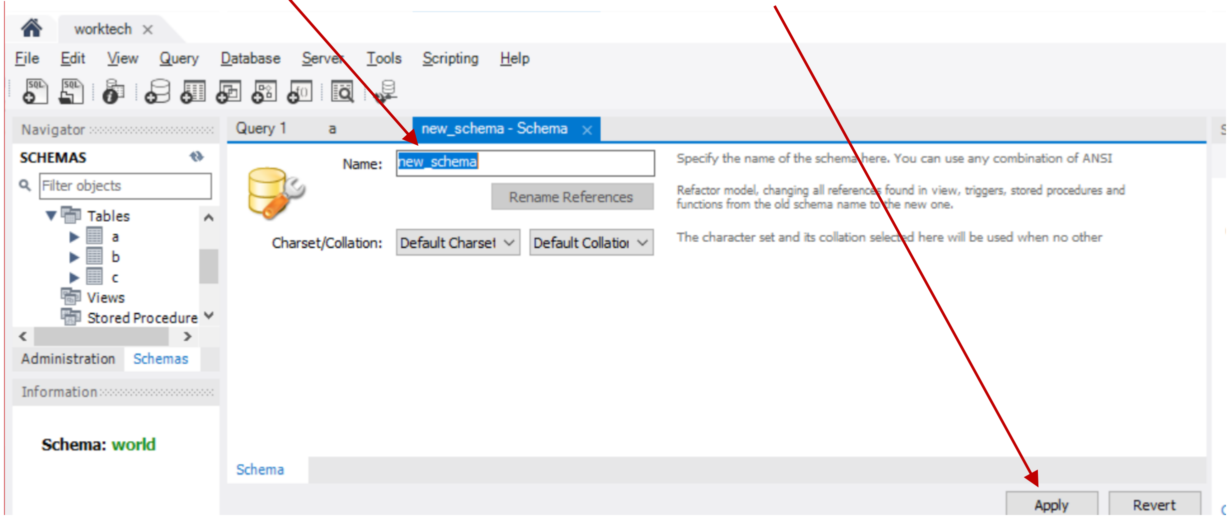
- Select **Schemas** under the navigator panel (you have Administrative or Schemas here)


If you are a database administrator, then can use Administration Navigator
If you are a designer or a database programmer, then you should use this
Administration
Schemas

- Select **Create a new schema in the connected server** from the toolbar



- Enter your Schema **Name** (type *schoolregistrar*) then click on **Apply**



Name: new_schema

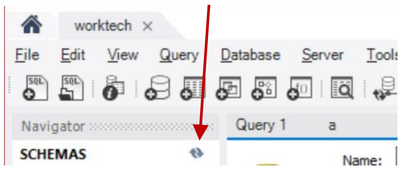
Charset/Collation: Default Charset Default Collation

Specify the name of the schema here. You can use any combination of ANSI

Refactor model, changing all references found in view, triggers, stored procedures and functions from the old schema name to the new one.

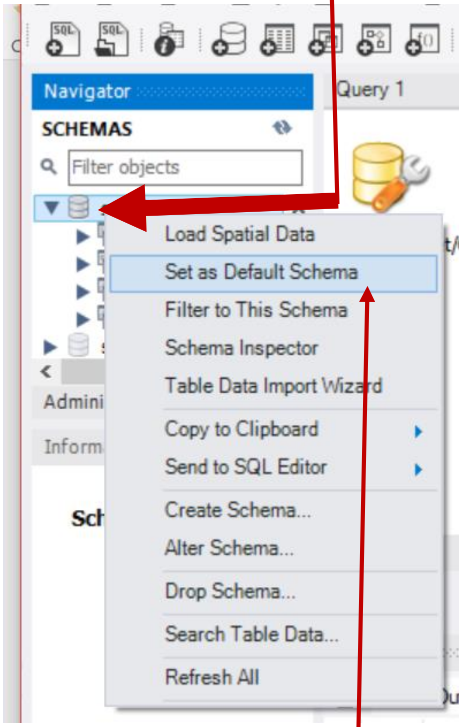
The character set and its collation selected here will be used when no other

- Now click on **Refresh** button to see the latest schemas or databases



Working with databases (schemas) – Setting your default schema (this is important especially when you work with SQL and if you want to create tables (DDL), use select query and other DMLs on this database)

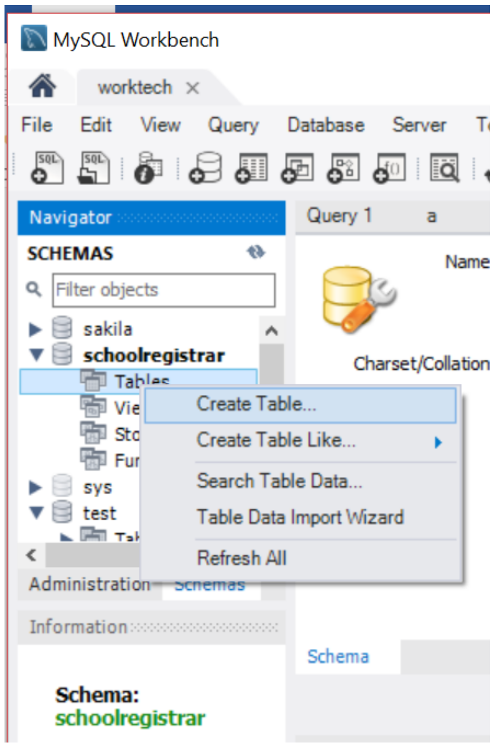
- Right click on the database you want to work with (here it is **schoolregistrar**)



- Select **Set as Default Schema** from the shortcut

Creating Tables

- Select the database to work with (expand **schoolregistrar**)
- Right click on **Tables,** then Select **Create Table…** as illustrated below



Schema: schoolregistrar

student
idStudent int (PK)
SFName varchar(45)
SLName varchar(45)
SAddress varchar(45)

- Enter your table name (student) and enter Column names (idStudent, SFName, SLName, SAddress) > Check PK under idStudent to make it **primary key**

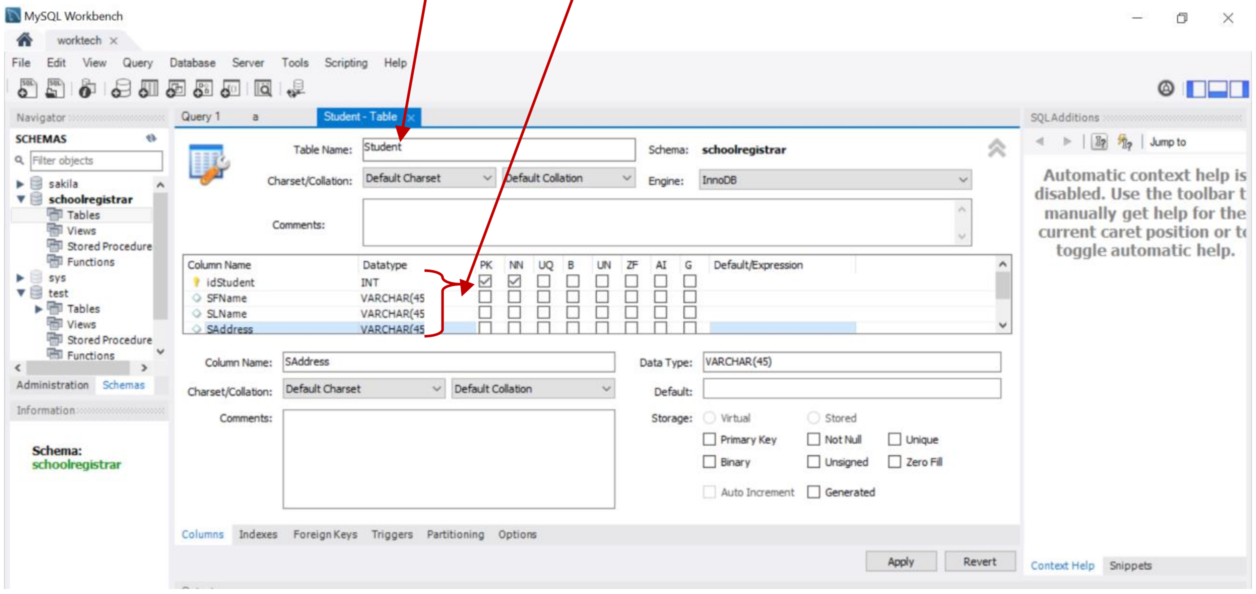


Table Name: Student
Schema: schoolregistrar
Charset/Collation: Default Charset / Default Collation
Engine: InnoDB
Comments:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
idStudent	INT	✓	✓							
SFName	VARCHAR(45									
SLName	VARCHAR(45									
SAddress	VARCHAR(45									

Column Name: SAddress
Data Type: VARCHAR(45)
Charset/Collation: Default Charset / Default Collation
Default:
Comments:
Storage: Virtual / Stored
Primary Key / Not Null / Unique
Binary / Unsigned / Zero Fill
Auto Increment / Generated

Automatic context help is disabled. Use the toolbar t manually get help for the current caret position or to toggle automatic help.

- Click on **Apply** when you are done and close the current table

Exercise: Create a new table and name it **course** with **courseId** as *primary key* (PK) with data type integer (or int), courseTitle with data type varchar(50), courseCredit with data type varchar(10).

course
courseId int (PK)
courseTitle varchar(50)
courseCredit varchar(10)

Tip – follow the steps described under **Creating Tables**.

Creating tables with foreign keys

In this step you are going to create a table with two foreign keys connecting to **course** and **student** tables created before. (**registration** table will have foreign keys connecting to course and student tables using their primary keys as foreign keys in the new table. So the new table will have registrationNo as primary key, courseId and IdStudent as foreign keys.

- Select the database to work with (expand **schoolregistrar**) if it is not selected
- Right click on **Tables,** then Select **Create Table…**
- Enter table name (registraion), columns (registrationNO primary key integer, courseId int, IdStudent int)

registation
registrationNo int (PK)
courseId int(FK)
idStudent int(FK)

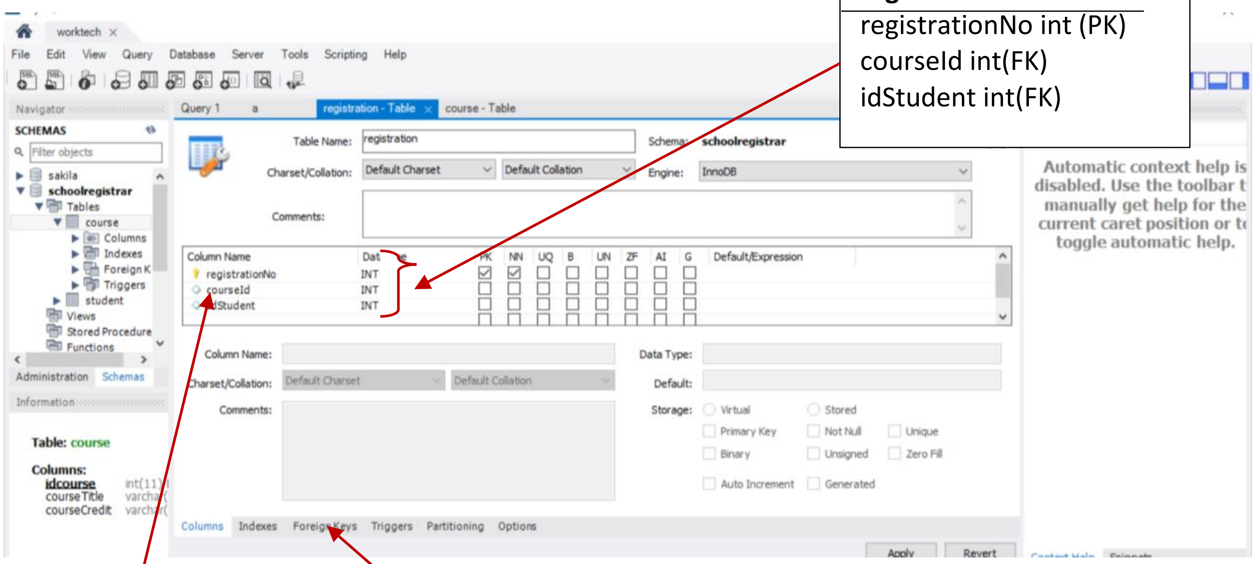


Table Name: registration
Schema: schoolregistrar
Charset/Collation: Default Charset / Default Collation
Engine: InnoDB

Column Name	Datatype	PK	NN
registrationNo	INT	✓	✓
courseId	INT		
idStudent	INT		

- Select **couseId** then click on Foreign Keys
- Select referenced column and corresponding table as illustrated below

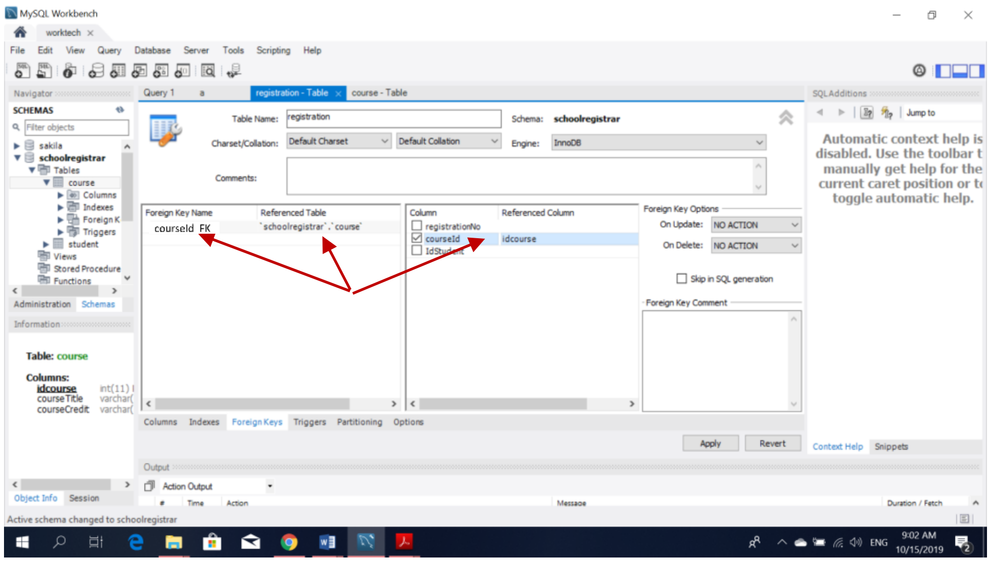


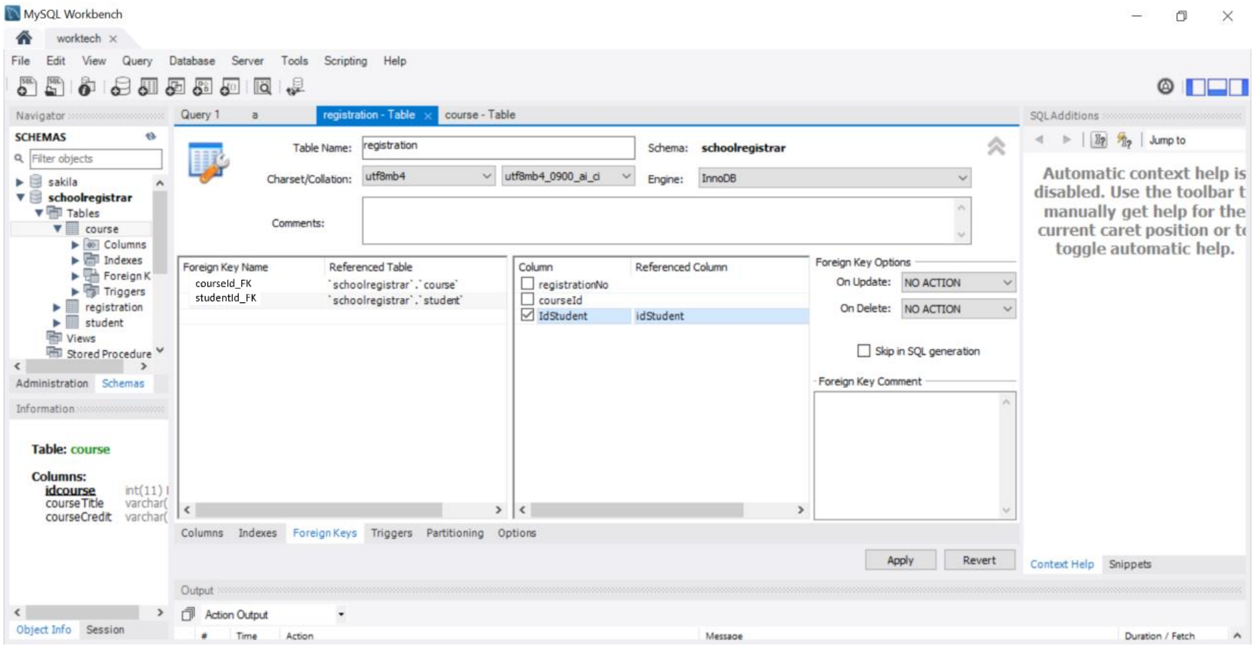
Table Name: registration
Schema: schoolregistrar

Foreign Key Name	Referenced Table
courseId_FK	`schoolregistrar`.`course`

Column	Referenced Column
☐ registrationNo	
☑ courseId	idcourse
☐ IdStudent	

Foreign Key Options
On Update: NO ACTION
On Delete: NO ACTION

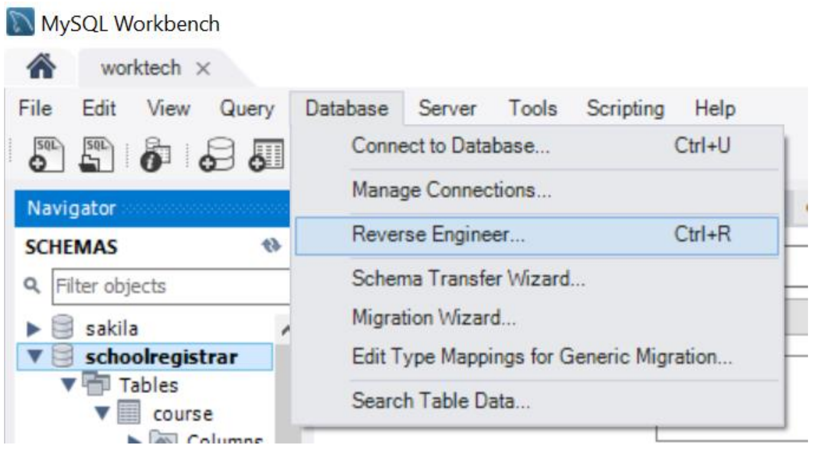
- Add a foreign key for student table as illustrated below



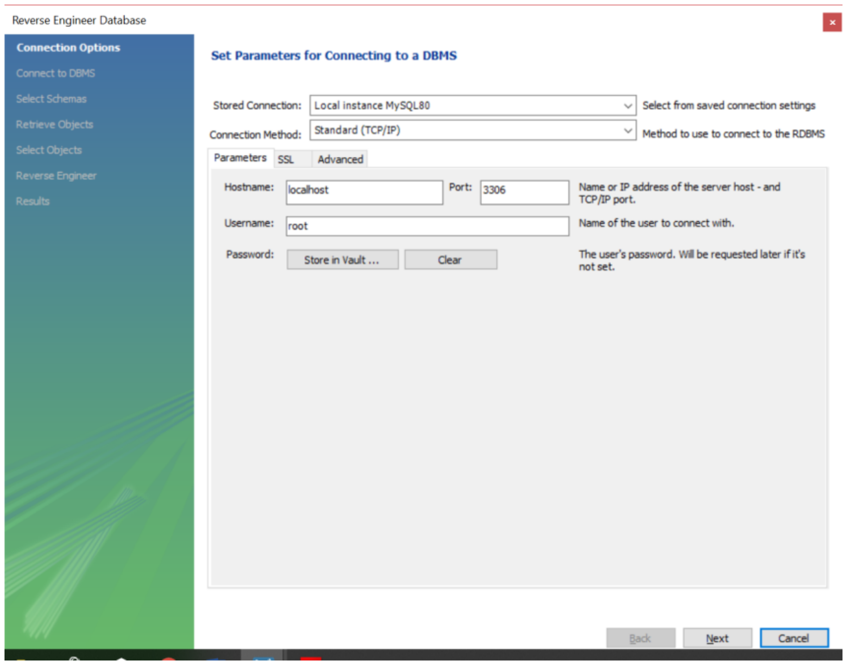
- Click on Apply, congratulations now you have created your table with its foreign keys.

Generating your database model (Enhanced Entity Relationship Diagram)

- Select the database to generate its Enhanced Entity Relationship Model (**schoolregistrar**)
- From the **Database** menu select **Reverse Engineer…**


MySQL Workbench
worktech
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
sakila
schoolregistrar
Tables
course
Connect to Database... Ctrl+U
Manage Connections...
Reverse Engineer... Ctrl+R
Schema Transfer Wizard...
Migration Wizard...
Edit Type Mappings for Generic Migration...
Search Table Data...

- Click don not change anything and click **Next**



Set Parameters for Connecting to a DBMS

Stored Connection: Local instance MySQL80 — Select from saved connection settings

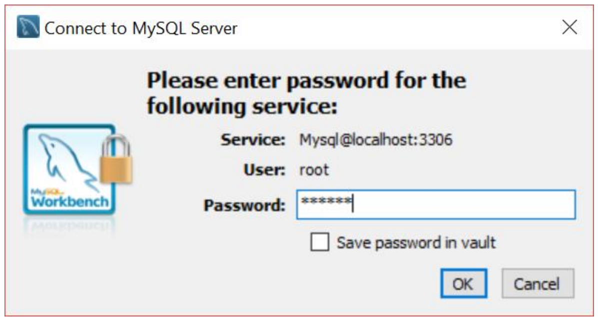
Connection Method: Standard (TCP/IP) — Method to use to connect to the RDBMS

Hostname: localhost Port: 3306 — Name or IP address of the server host - and TCP/IP port.

Username: root — Name of the user to connect with.

Password: Store in Vault ... Clear — The user's password. Will be requested later if it's not set.

- Enter **your password** and click **OK**



Please enter password for the following service:

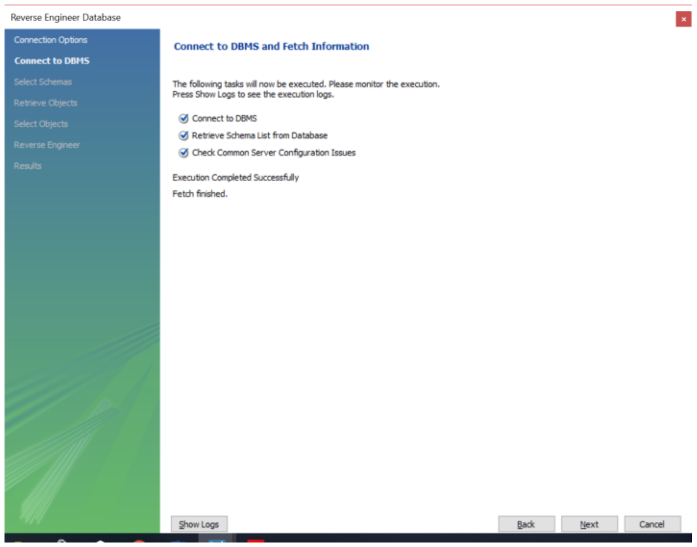
Service: Mysql@localhost:3306

User: root

Password: ******

Save password in vault

- Now you will see it is successfully fetched the information as illustrated below and Click **Next**



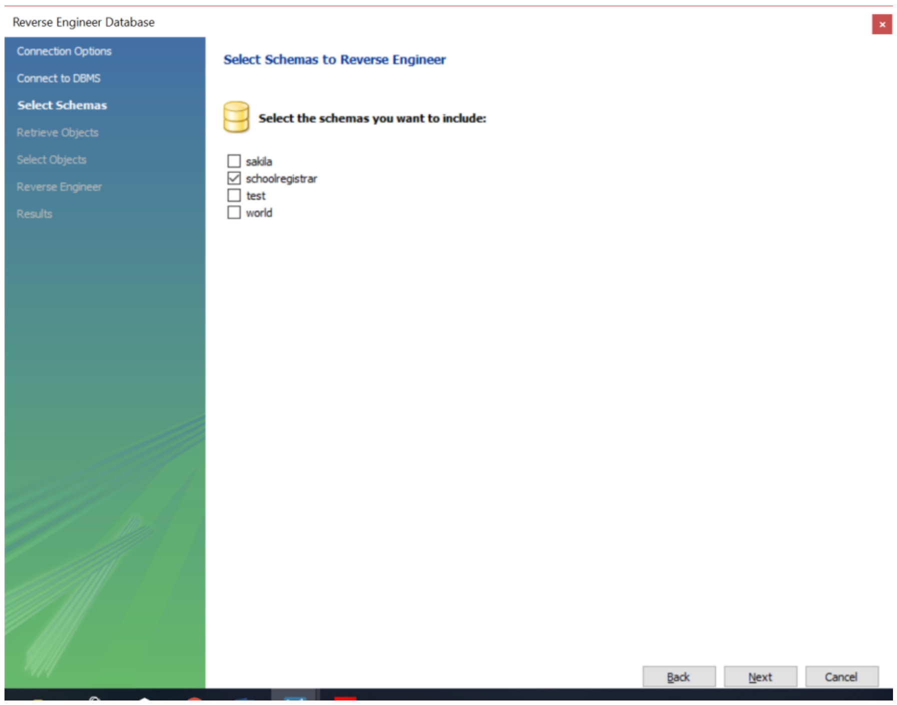
Connect to DBMS and Fetch Information

The following tasks will now be executed. Please monitor the execution.
Press Show Logs to see the execution logs.

✓ Connect to DBMS
✓ Retrieve Schema List from Database
✓ Check Common Server Configuration Issues

Execution Completed Successfully
Fetch finished.

- Select the database (**schoolregistrar**) you want to generate the EER model for as illustrated below and click **Next**

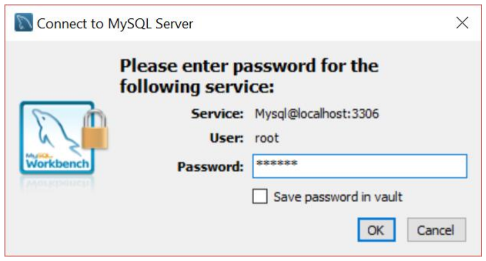

Reverse Engineer Database

Select Schemas to Reverse Engineer

Select the schemas you want to include:

- [ ] sakila
- [x] schoolregistrar
- [ ] test
- [ ] world

- Enter **your password**


Connect to MySQL Server

Please enter password for the following service:

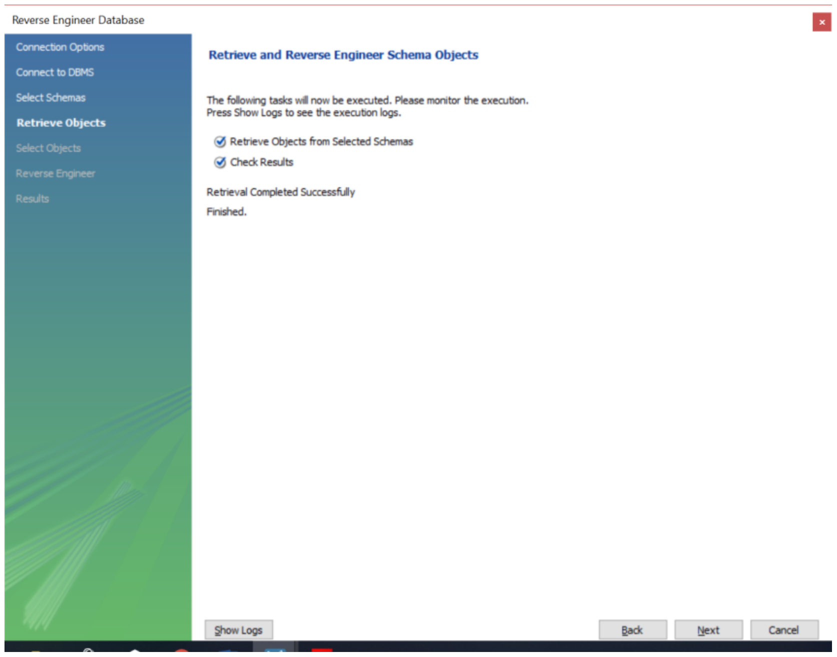
Service: Mysql@localhost:3306

User: root

Password: ******

- [ ] Save password in vault

- You will see that the objects are retrieved successfully then as illustrated below Click **Next**


Reverse Engineer Database

Retrieve and Reverse Engineer Schema Objects

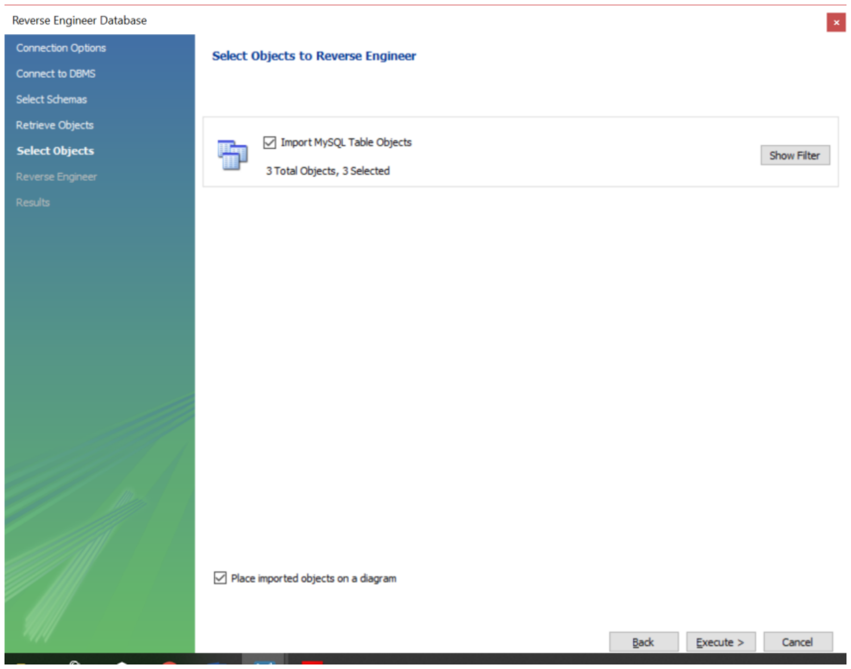
The following tasks will now be executed. Please monitor the execution.
Press Show Logs to see the execution logs.

- ✓ Retrieve Objects from Selected Schemas
- ✓ Check Results

Retrieval Completed Successfully

Finished.

- Make sure that Import **MySQL Table Objects** is checked as illustrated below then Click on **Execute**



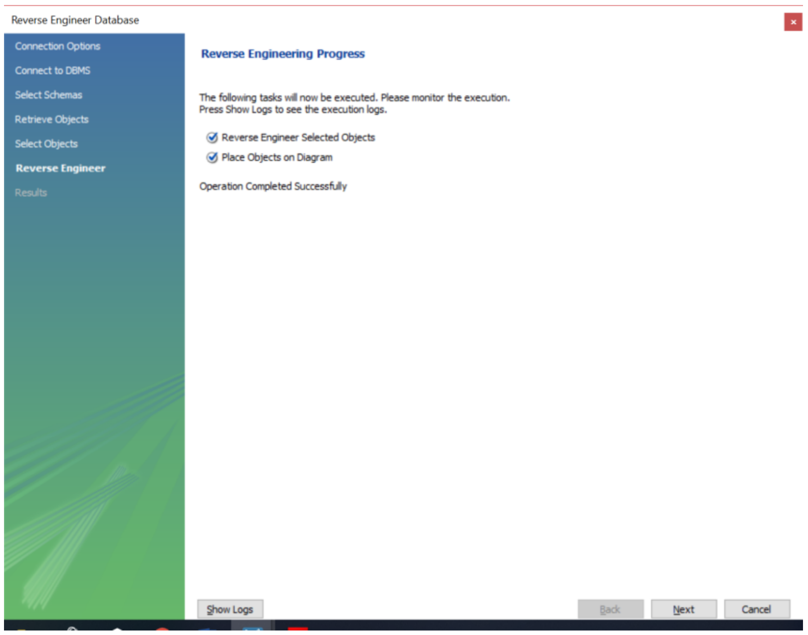
Select Objects to Reverse Engineer

☑ Import MySQL Table Objects

3 Total Objects, 3 Selected

☑ Place imported objects on a diagram

- You will see that the reverse engineering process completed successfully as illustrated below, click **Next**



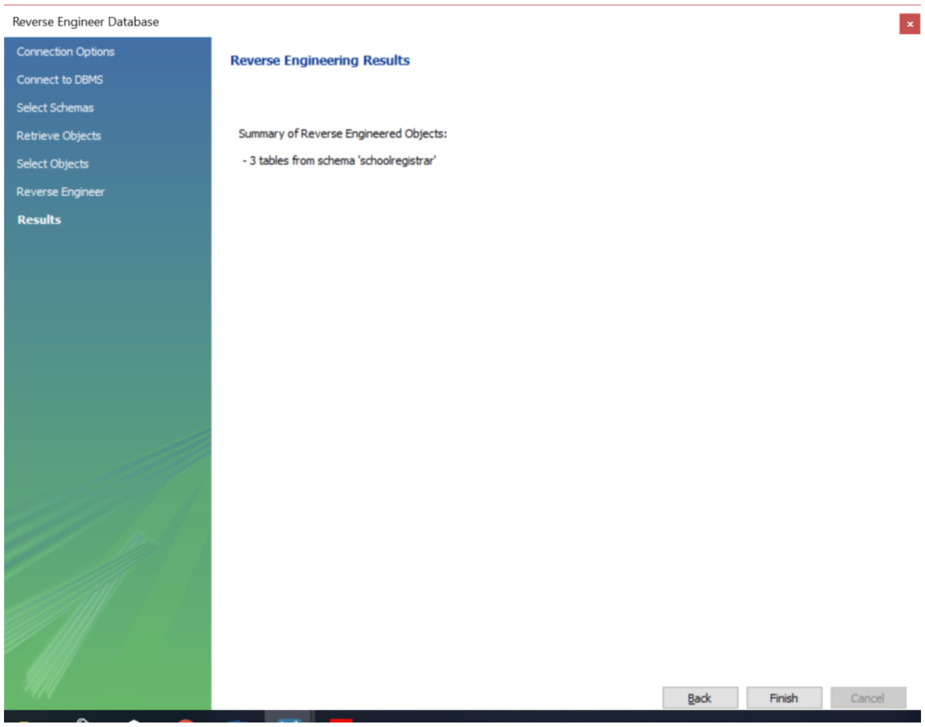
Reverse Engineering Progress

The following tasks will now be executed. Please monitor the execution.
Press Show Logs to see the execution logs.

✓ Reverse Engineer Selected Objects
✓ Place Objects on Diagram

Operation Completed Successfully

- Finally, you will see Reverse Engineering Results finished successfully then Click on **Finish**

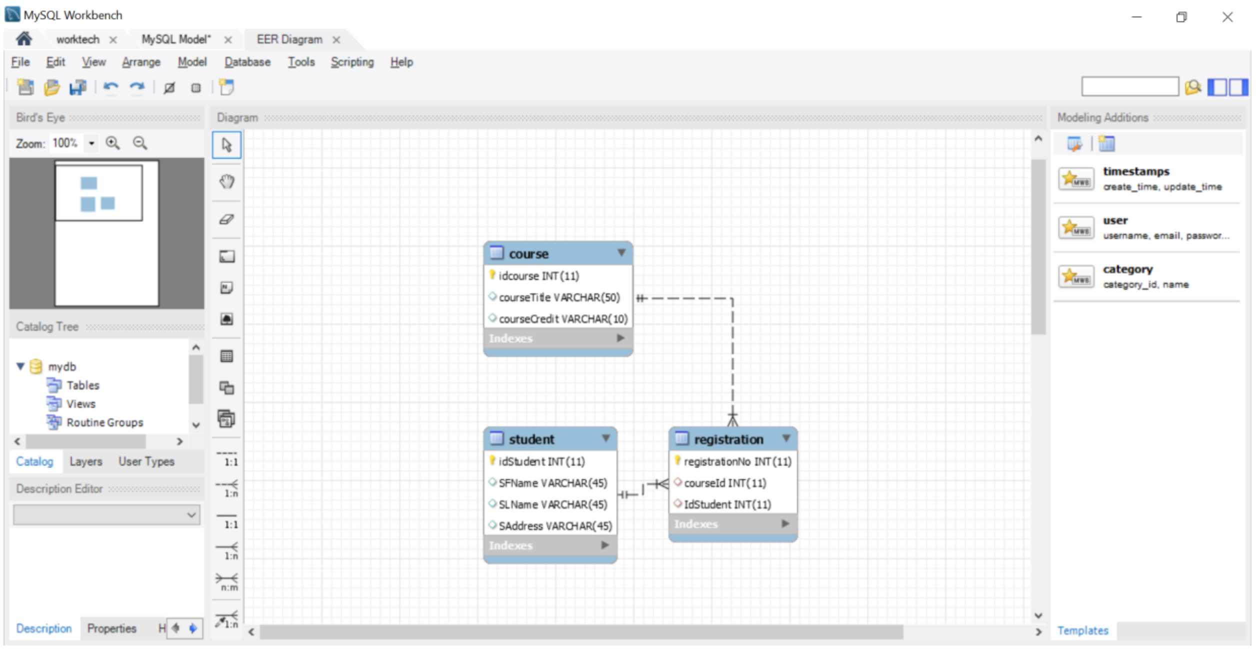

Reverse Engineer Database

Reverse Engineering Results

Summary of Reverse Engineered Objects:

- 3 tables from schema 'schoolregistrar'

- You will see the relational model generated illustrated below, if it is not visible you can use the scroll bars



course
- idcourse INT(11)
- courseTitle VARCHAR(50)
- courseCredit VARCHAR(10)

student
- idStudent INT(11)
- SFName VARCHAR(45)
- SLName VARCHAR(45)
- SAddress VARCHAR(45)

registration
- registrationNo INT(11)
- courseId INT(11)
- IdStudent INT(11)

Exercise:

1. Create a new database (schema) – football
2. Create the following tables
   a. Table name - teams
      i. Column1 - teamId int primary key

        ii. Column2 – teamName varchar(50)
        iii. Column3 – teamCity varchar(20)

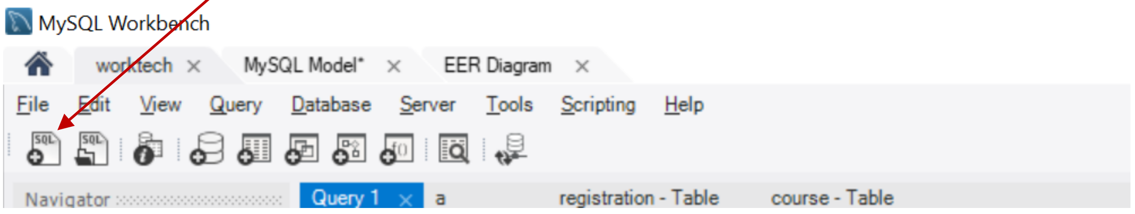
b. Table name – players
    i. Column1 - playerId int primary key
    ii. Column2 – playerName varchar(25)
    iii. Column3 – playerArea varchar(20)
    iv. Column4 - teamId int Foregin Key referencing table (team) (enter the name of the Foreign Key as – fk_players_coaches)
c. Table name – coahes
    i. Column1 - coachId int primary key
    ii. Column2 – coachName varchar(25)
    iii. Column3 – yearsOfcoaching int
    iv. Column4 - teamId int Foregin Key referencing table (team) (Note that **Foreign Key** names should be unique so here name it as – fk_teams_coaches)

3. Generate the relational model (EER Model as you did in the exercise above)
4. Copy a snapshot to show it to your teacher

Part two – Using SQL to create database objects

1. Creating databases using SQL

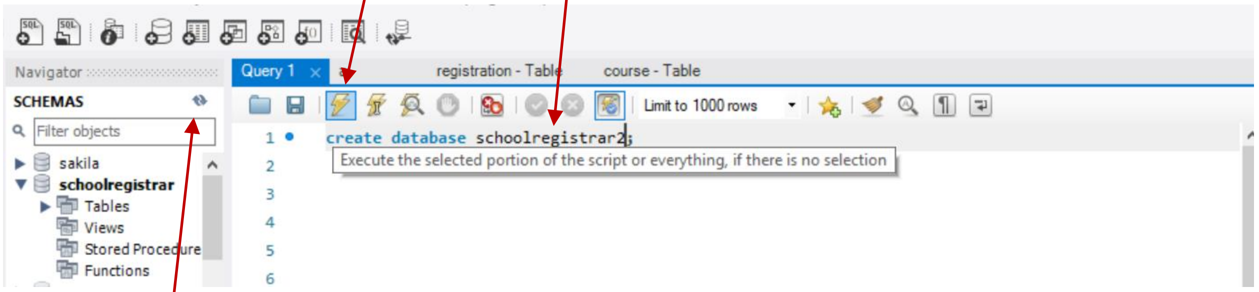
- Select **Create a new SQL tab for executing queries**



- Enter SQL statement in your query window as illustrated below

create database schoolregistrar2

- Select **Execute the selected portion of the script for everything**
  (Note: When you execute SQL statements, you should be careful. If you have several SQL statements, then MYSQL will run all statements unless you make a selection of the desired SQL statement you want to run)



```
create database schoolregistrar2;
```

- Refresh the Schemas to see the database just created
- Set **schoolregistrar2** as the default schema

Use the following query to create these tables:

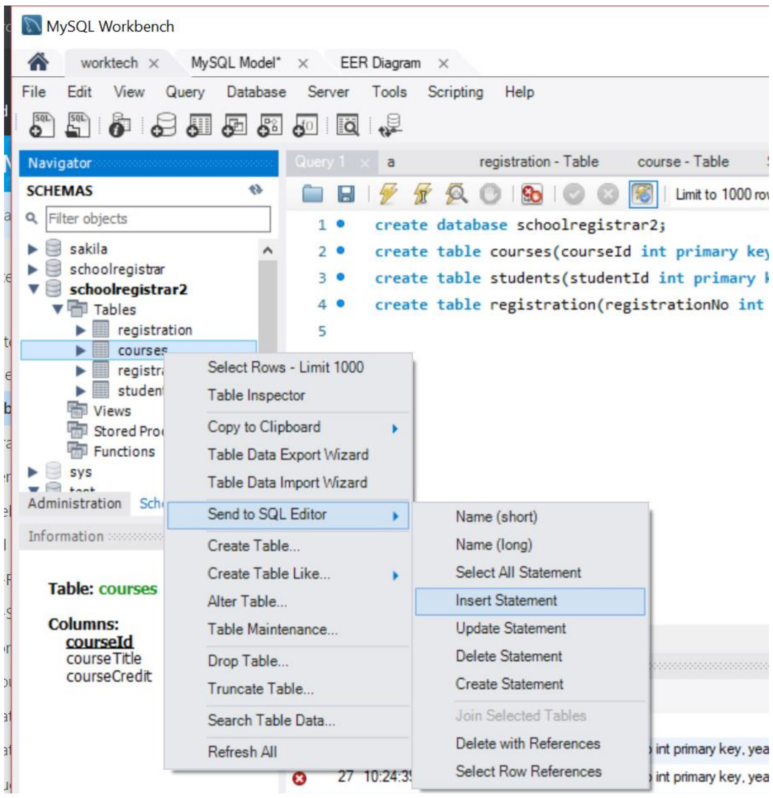
- create table courses(courseId int primary key, courseTitle varchar(23), courseCredit varchar(20));
- create table students(studentId int primary key, sFname varchar(25), sLname varchar(25));
- create table registration(registrationNo int primary key, yearOfRegistration int, courseId int, studentId int, constraint Foreign key fk_reg_to_courses1(courseId) references courses(courseId), constraint Foreign key fk_reg_to_students1(studentId) references students(studentId));

2. Inserting data into tables DML

In this step we will insert data into three tables: courses, students, and registration

- Insert statement – use insert statement to insert data into all three tables

- Right click on the table – course > pint to **Send to SQL Editor** > select **Insert Statement**

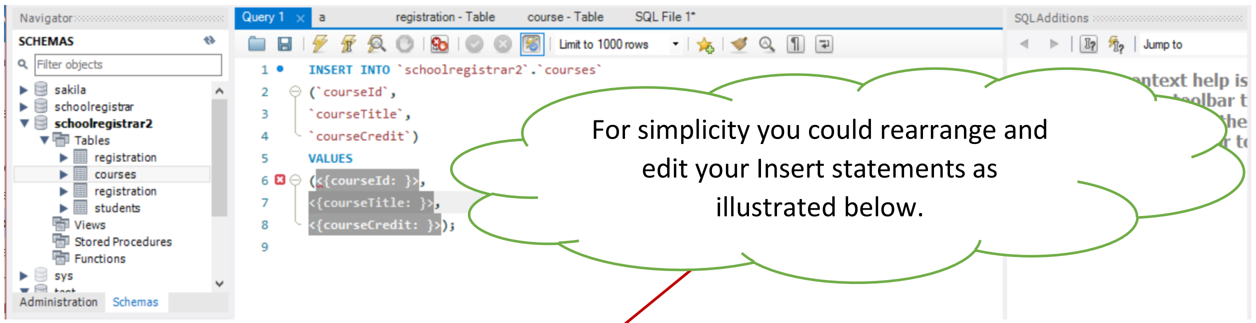


```
create database schoolregistrar2;
create table courses(courseId int primary key
create table students(studentId int primary k
create table registration(registrationNo int
```

Table: courses

Columns:
courseId
courseTitle
courseCredit

- Now you will have the ***Insert Statement*** created for you, Note that you need to replace place holders of values with real values, and rearrange your insert statement as illustrated in the next step



```
INSERT INTO `schoolregistrar2`.`courses`
(`courseId`,
`courseTitle`,
`courseCredit`)
VALUES
(<{courseId: }>,
<{courseTitle: }>,
<{courseCredit: }>);
```

- Add the values into **course** table using the values specified in following SQL statements, you can run each insert statement only once because of the Primary Key constraint. So select each of the following insert statements and execute them separately.

INSERT INTO `schoolregistrar2`.`courses` (`courseId`,`courseTitle`,`courseCredit`) VALUES(1, "DBMS", "7.5");

INSERT INTO `schoolregistrar2`.`courses` (`courseId`,`courseTitle`,`courseCredit`) VALUES(2, "Computer Security", "7.5");

```
INSERT INTO `schoolregistrar2`.`courses`
(`courseId`,`courseTitle`,`courseCredit`) VALUES(3, "HCI", "7.5");

INSERT INTO `schoolregistrar2`.`courses`
(`courseId`,`courseTitle`,`courseCredit`) VALUES(4, "Software Engineering", "7.5");

INSERT INTO `schoolregistrar2`.`courses`
(`courseId`,`courseTitle`,`courseCredit`) VALUES(5, "Networking", "7.5");
```

- Add the values into **students** table using the values specified in following SQL statements

```
INSERT INTO `schoolregistrar2`.`students`
(`studentId`, `sFname`, `sLname`) VALUES (1, "Isac","James");

INSERT INTO `schoolregistrar2`.`students`
(`studentId`, `sFname`, `sLname`) VALUES (2, "Erik","Snidders");

INSERT INTO `schoolregistrar2`.`students`
(`studentId`, `sFname`, `sLname`) VALUES (3, "Manhatan","George");

INSERT INTO `schoolregistrar2`.`students`
(`studentId`, `sFname`, `sLname`) VALUES (4, "Suleuman","Mohamed");

INSERT INTO `schoolregistrar2`.`students`
(`studentId`, `sFname`, `sLname`) VALUES (5, "Christian","Alfred");

INSERT INTO `schoolregistrar2`.`students`
(`studentId`, `sFname`, `sLname`) VALUES (6, "Susan","Rice");

INSERT INTO `schoolregistrar2`.`students`
(`studentId`, `sFname`, `sLname`) VALUES (7, "Zuckberg","Dickson");
```

- Add the values into **registration** table using the values specified in following SQL statements

```
INSERT INTO `schoolregistrar2`.`registration`
(`registrationNo`, `yearOfRegistration`, `courseId`, `studentId`)
VALUES (2, 2014, 2,2);

INSERT INTO `schoolregistrar2`.`registration`
(`registrationNo`, `yearOfRegistration`, `courseId`, `studentId`)
VALUES (3, 2014, 2,2);

INSERT INTO `schoolregistrar2`.`registration`
(`registrationNo`, `yearOfRegistration`, `courseId`, `studentId`)
VALUES (4, 2014, 2,2);

INSERT INTO `schoolregistrar2`.`registration`
(`registrationNo`, `yearOfRegistration`, `courseId`, `studentId`)
```

VALUES (5, 2014, 1,1);

INSERT INTO `schoolregistrar2`.`registration`
(`registrationNo`, `yearOfRegistration`, `courseId`, `studentId`)
VALUES (6, 2014, 1,1);

3. Querying tables DML

You can use Select statement to query any table in this database

Example: Select * From Students

4. Updating tables DML

Change all course credits from 7.5 to 15 for courseIDs 1 and 5

UPDATE `schoolregistrar2`.`courses`
SET `courseCredit` = "15"
WHERE `courseId` = 1 and `courseId` = 5;

Exercise: Change all credits back to 7.5

Exercise: Change Networking to Network Security for *`courseId`* = 5

5. Deleting rows in tables DML

Be careful! – you might accidentally delete everything unless you specify the where clause as illustrated in the example below.

Example:

DELETE FROM `schoolregistrar2`.`students`
WHERE studentId = 6;

Exercise – create a new database – football2, then create all tables available in football database created before using sql statements. **Caution** – Foreign Key constrains no matter where they are applied must be uniquely named so you need to make sure that the foreign keys are different from the foreign keys in football database.