University of New York Tirana
Faculty of Engineering and Architecture
Rruga e Kavajës, pranë 21 Dhjetorit (Sheshi Ataturk)
Tirane, Shqipëri

Master of Science in Computer Science

Distributed Systems
Manual for Laboratory Practice

PART III - Remote Method Invocation
Full Application with Three Tiers

Prof. Dr. Marenglen Biba
Department of Computer Science
E-mail: marenglenbiba@unyt.edu.al

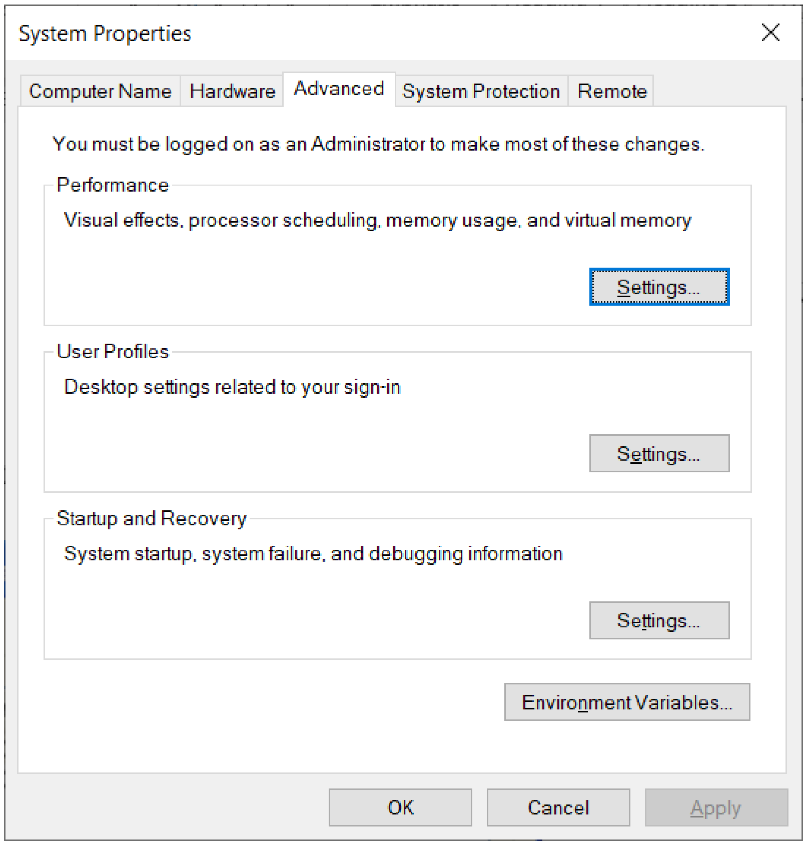
1. Document Purpose

This document contains explanations on how to run the following programs: RMI Servers, RMI Client and Database Server.

For running the programs, a correct configuration of the running environment is necessary (path and classpath variables).

- Install JAVA JDK 21
- Install Apache Netbeans 21
- Install MySQL
- Install MySQL Workbench
- Set path system variable in the operating system

For running the programs, a correct configuration of the running environment is necessary (path variable).



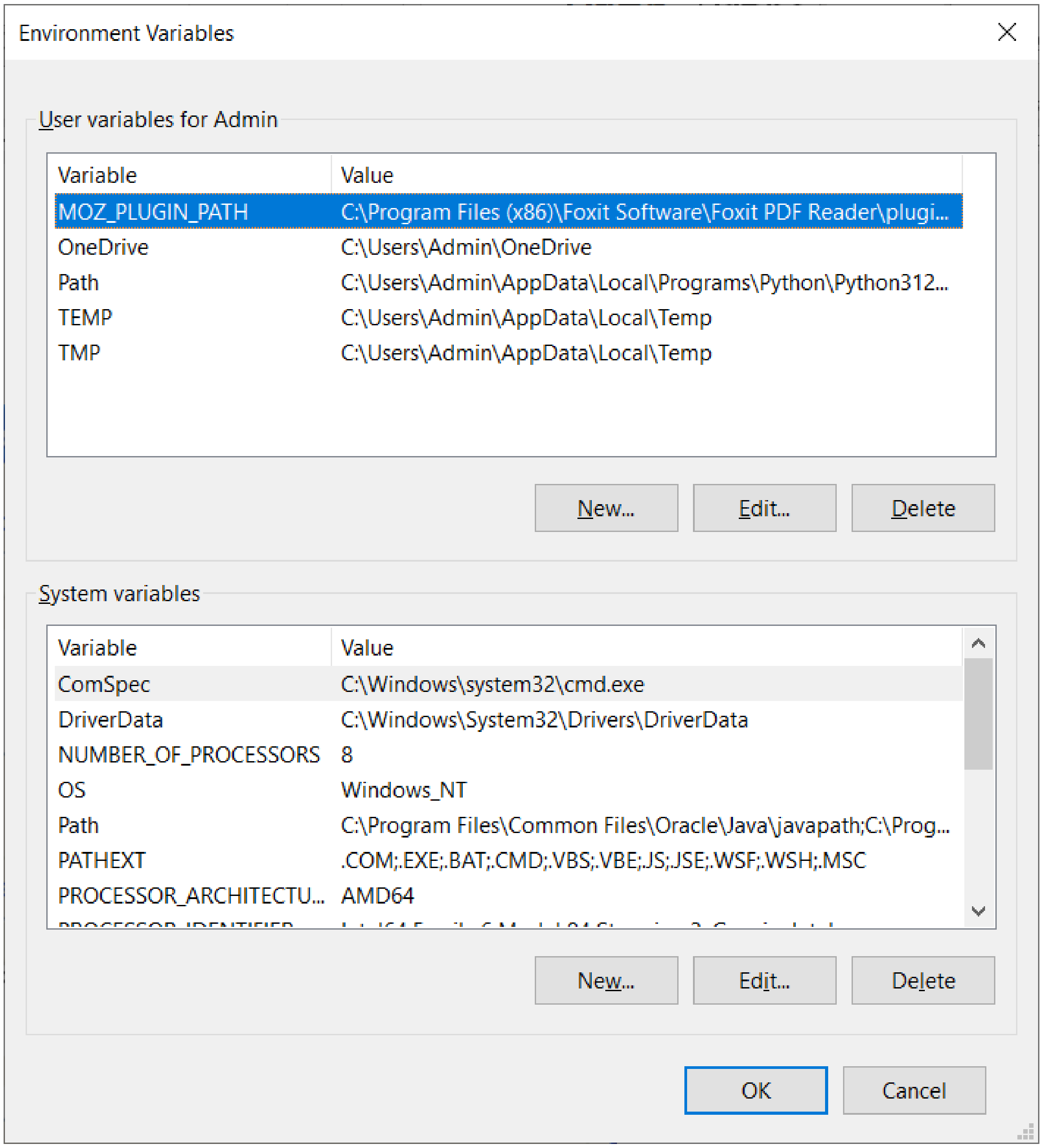
You must be logged on as an Administrator to make most of these changes.

Performance
Visual effects, processor scheduling, memory usage, and virtual memory

User Profiles
Desktop settings related to your sign-in

Startup and Recovery
System startup, system failure, and debugging information

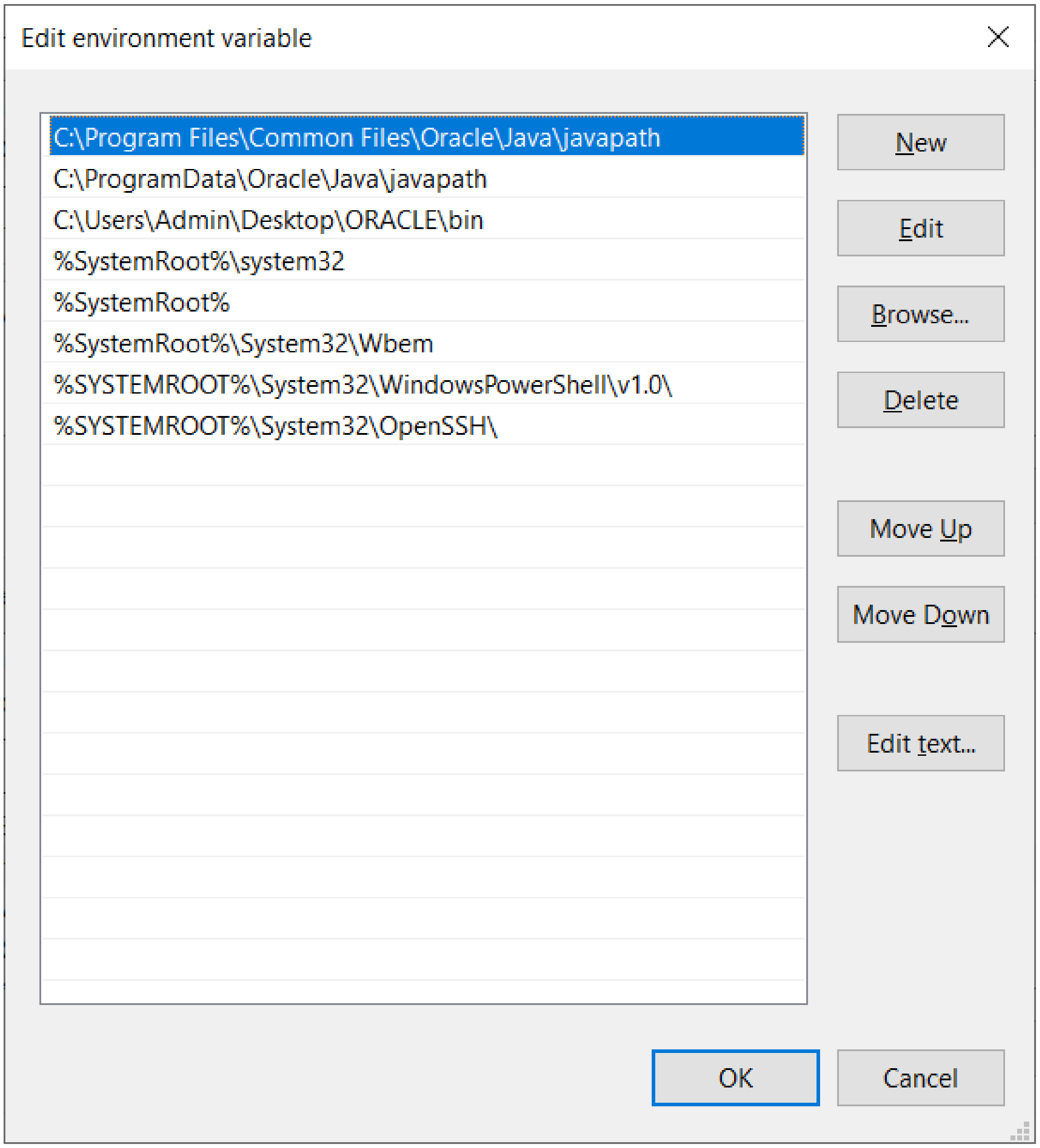
Click on Environment Variables.



Variable	Value
MOZ_PLUGIN_PATH	C:\Program Files (x86)\Foxit Software\Foxit PDF Reader\plugi...
OneDrive	C:\Users\Admin\OneDrive
Path	C:\Users\Admin\AppData\Local\Programs\Python\Python312...
TEMP	C:\Users\Admin\AppData\Local\Temp
TMP	C:\Users\Admin\AppData\Local\Temp

Variable	Value
ComSpec	C:\Windows\system32\cmd.exe
DriverData	C:\Windows\System32\Drivers\DriverData
NUMBER_OF_PROCESSORS	8
OS	Windows_NT
Path	C:\Program Files\Common Files\Oracle\Java\javapath;C:\Prog...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC
PROCESSOR_ARCHITECTU...	AMD64

Find the Path system variable and click Edit.

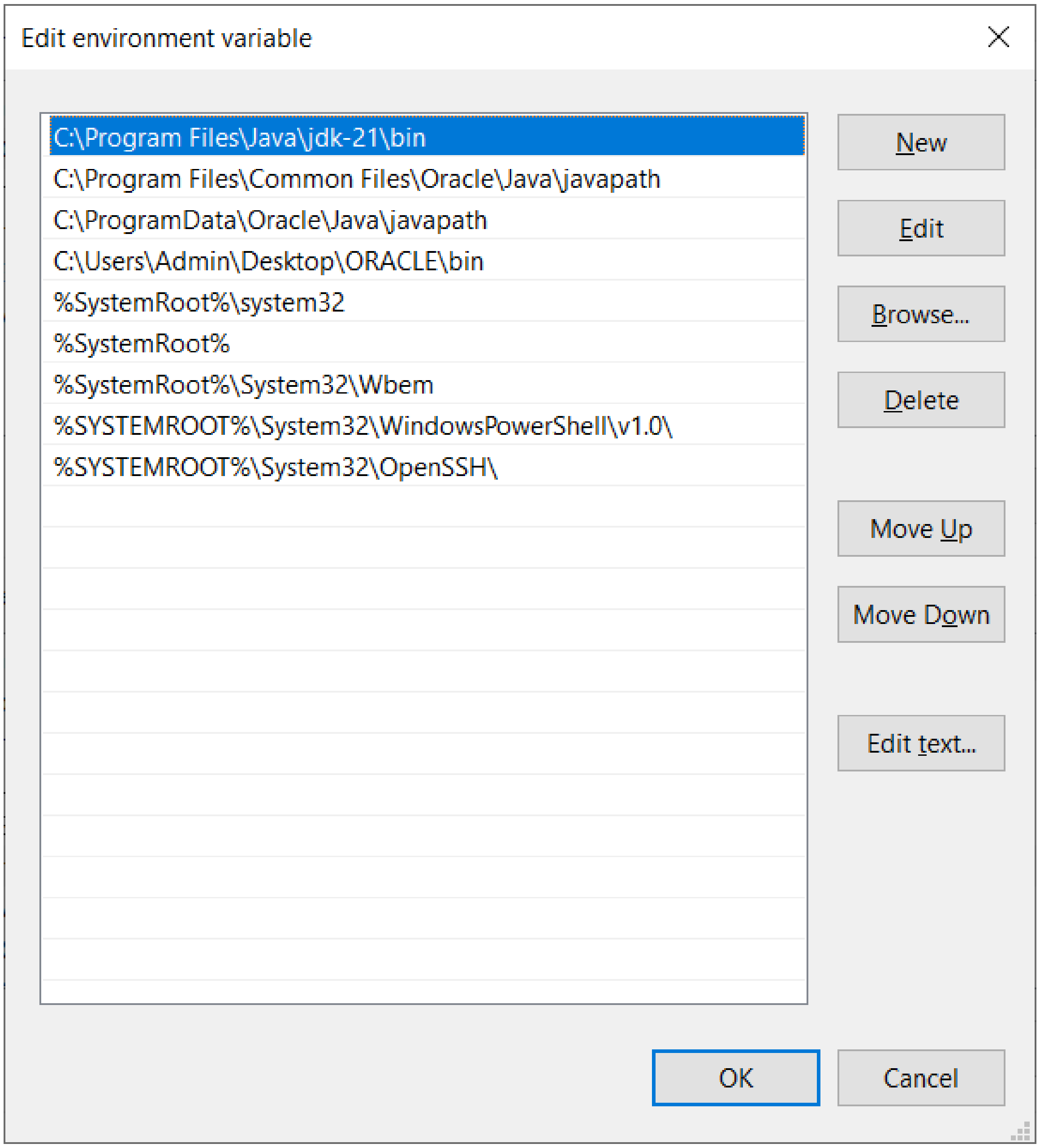

Edit environment variable

```
C:\Program Files\Common Files\Oracle\Java\javapath
C:\ProgramData\Oracle\Java\javapath
C:\Users\Admin\Desktop\ORACLE\bin
%SystemRoot%\system32
%SystemRoot%
%SystemRoot%\System32\Wbem
%SYSTEMROOT%\System32\WindowsPowerShell\v1.0\
%SYSTEMROOT%\System32\OpenSSH\
```

Select New and set the value of the variable to the directory where you have installed Java, for example:

C:\Program Files\Java\jdk-21\bin

Move the item up as follows:



Edit environment variable

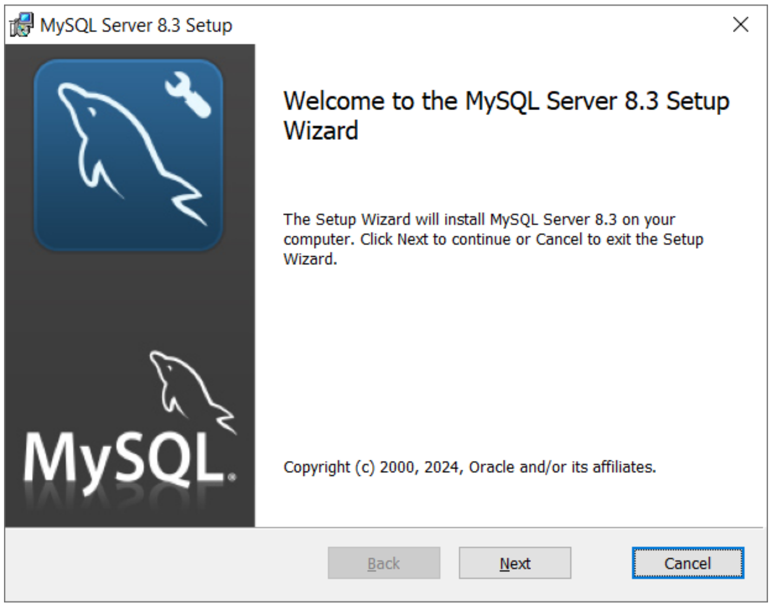
```
C:\Program Files\Java\jdk-21\bin
C:\Program Files\Common Files\Oracle\Java\javapath
C:\ProgramData\Oracle\Java\javapath
C:\Users\Admin\Desktop\ORACLE\bin
%SystemRoot%\system32
%SystemRoot%
%SystemRoot%\System32\Wbem
%SYSTEMROOT%\System32\WindowsPowerShell\v1.0\
%SYSTEMROOT%\System32\OpenSSH\
```

Download and Install MySQL Server

MySQL Community Server 8.3.0

https://dev.mysql.com/downloads/installer/

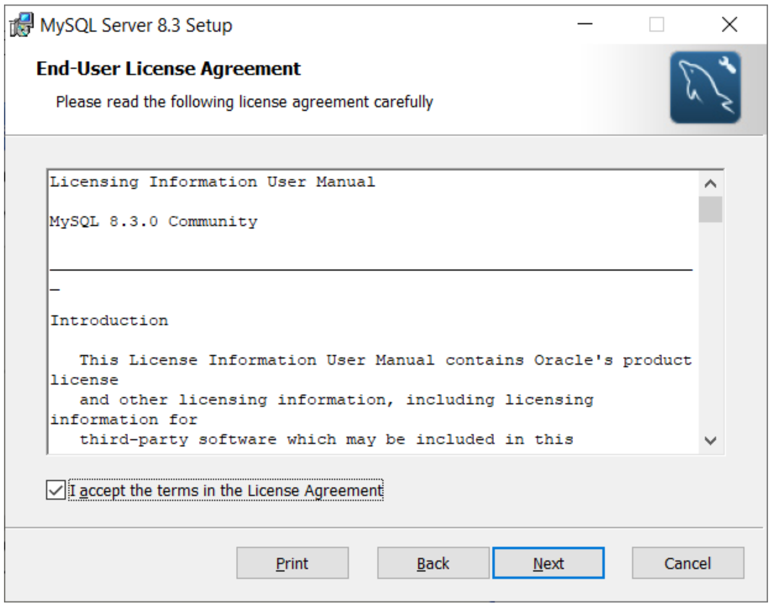
After you download use MySQL Server Instance Config Wizard



Welcome to the MySQL Server 8.3 Setup Wizard

The Setup Wizard will install MySQL Server 8.3 on your computer. Click Next to continue or Cancel to exit the Setup Wizard.

Copyright (c) 2000, 2024, Oracle and/or its affiliates.



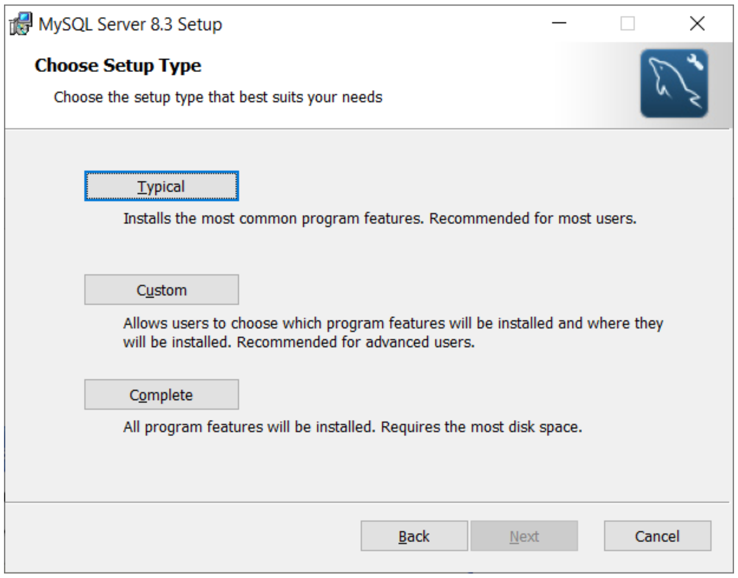
End-User License Agreement

Please read the following license agreement carefully

```
Licensing Information User Manual

MySQL 8.3.0 Community

________________________________________________________________
_

Introduction

   This License Information User Manual contains Oracle's product
license
   and other licensing information, including licensing
information for
   third-party software which may be included in this
```

[✓] I accept the terms in the License Agreement

Choose the Custom configuration as follows:


MySQL Server 8.3 Setup

Choose Setup Type

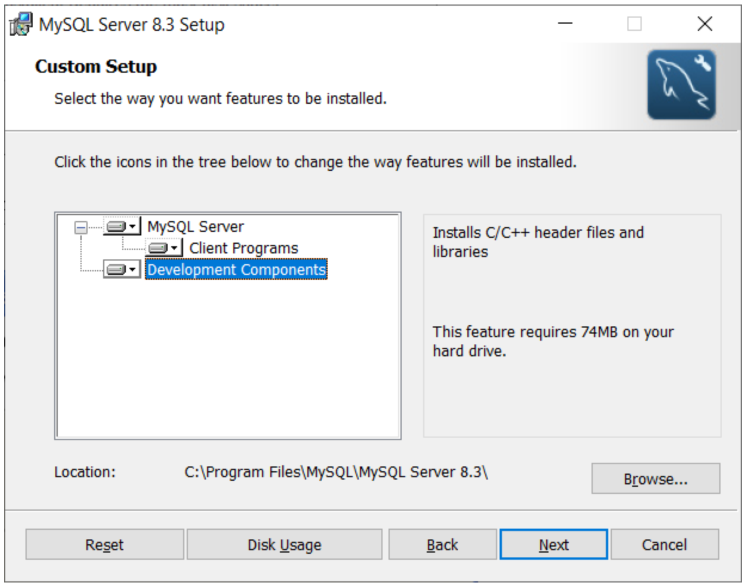
Choose the setup type that best suits your needs

Installs the most common program features. Recommended for most users.

Allows users to choose which program features will be installed and where they will be installed. Recommended for advanced users.

All program features will be installed. Requires the most disk space.

Install all on local hard drive in the following window:


MySQL Server 8.3 Setup

Custom Setup

Select the way you want features to be installed.

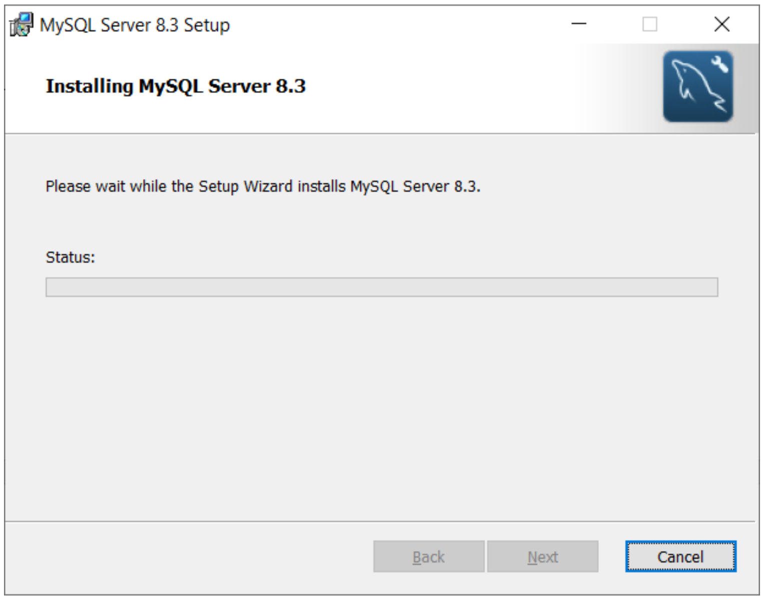
Click the icons in the tree below to change the way features will be installed.

Installs C/C++ header files and libraries

This feature requires 74MB on your hard drive.

Location: C:\Program Files\MySQL\MySQL Server 8.3\

Proceed with next step as follows:

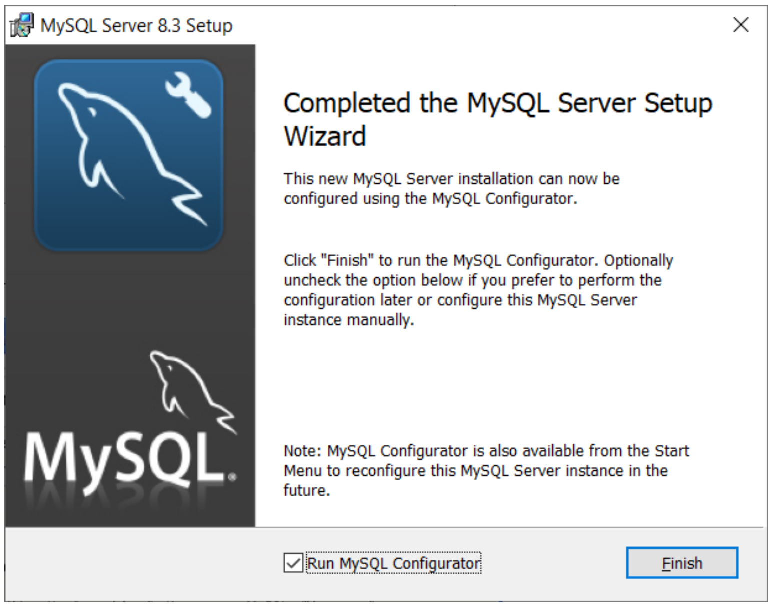

MySQL Server 8.3 Setup

Installing MySQL Server 8.3

Please wait while the Setup Wizard installs MySQL Server 8.3.

Status:

When you have finished copying the files, proceed with the configuration as follows:


MySQL Server 8.3 Setup

Completed the MySQL Server Setup Wizard

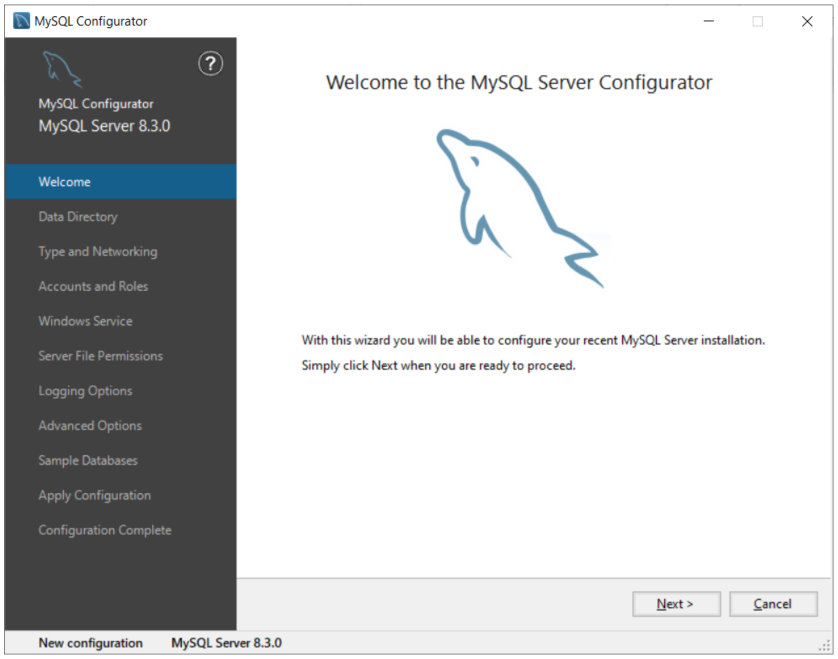
This new MySQL Server installation can now be configured using the MySQL Configurator.

Click "Finish" to run the MySQL Configurator. Optionally uncheck the option below if you prefer to perform the configuration later or configure this MySQL Server instance manually.

Note: MySQL Configurator is also available from the Start Menu to reconfigure this MySQL Server instance in the future.

[✓] Run MySQL Configurator

Click finish and continue with the configuration instance as follows:

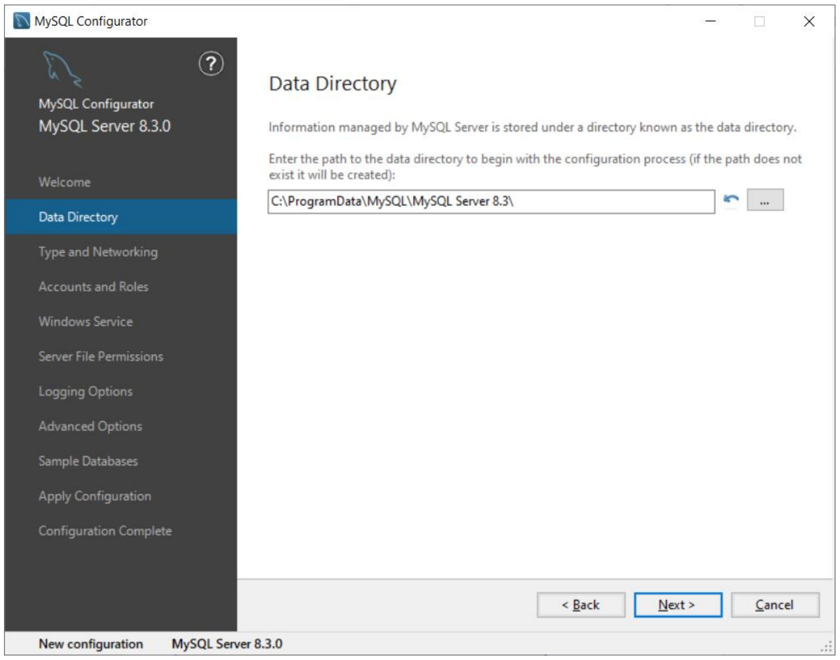


Welcome to the MySQL Server Configurator

With this wizard you will be able to configure your recent MySQL Server installation.

Simply click Next when you are ready to proceed.

Choose the path of data storage:



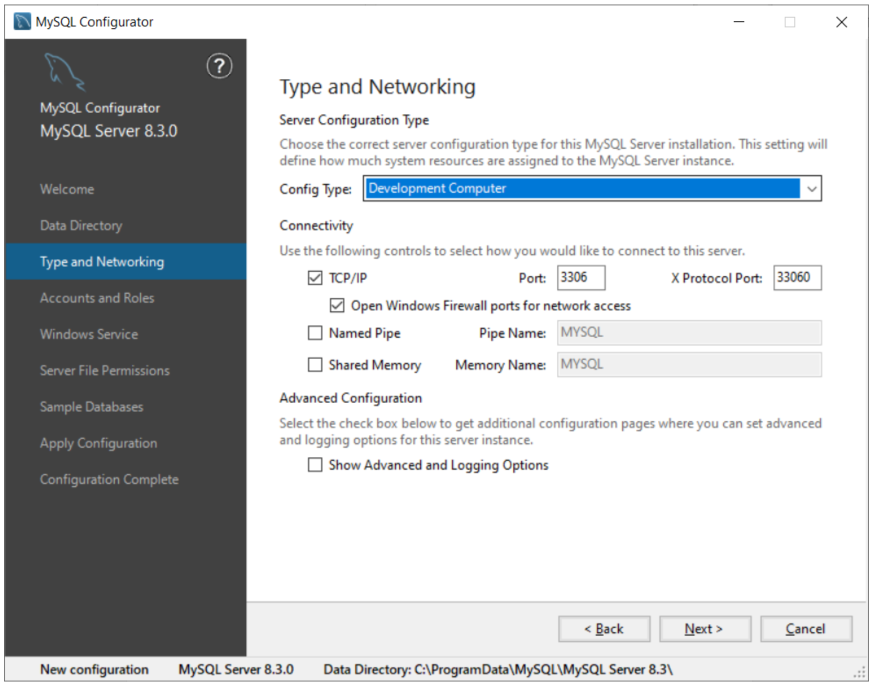
Data Directory

Information managed by MySQL Server is stored under a directory known as the data directory.

Enter the path to the data directory to begin with the configuration process (if the path does not exist it will be created):

C:\ProgramData\MySQL\MySQL Server 8.3\

Choose Developer settings as follows:



Type and Networking

Server Configuration Type

Choose the correct server configuration type for this MySQL Server installation. This setting will define how much system resources are assigned to the MySQL Server instance.

Config Type: Development Computer

Connectivity

Use the following controls to select how you would like to connect to this server.

[✓] TCP/IP Port: 3306 X Protocol Port: 33060

[✓] Open Windows Firewall ports for network access

[ ] Named Pipe Pipe Name: MYSQL

[ ] Shared Memory Memory Name: MYSQL

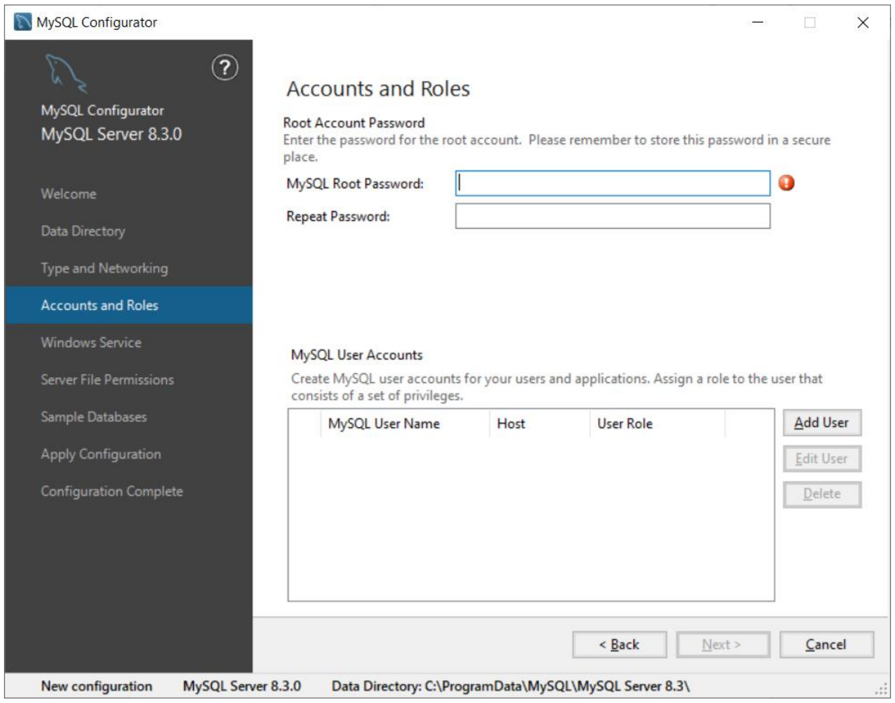
Advanced Configuration

Select the check box below to get additional configuration pages where you can set advanced and logging options for this server instance.

[ ] Show Advanced and Logging Options

New configuration MySQL Server 8.3.0 Data Directory: C:\ProgramData\MySQL\MySQL Server 8.3\

Choose the root password in the next window as follows (password: admin).



Accounts and Roles

Root Account Password

Enter the password for the root account. Please remember to store this password in a secure place.

MySQL Root Password:

Repeat Password:

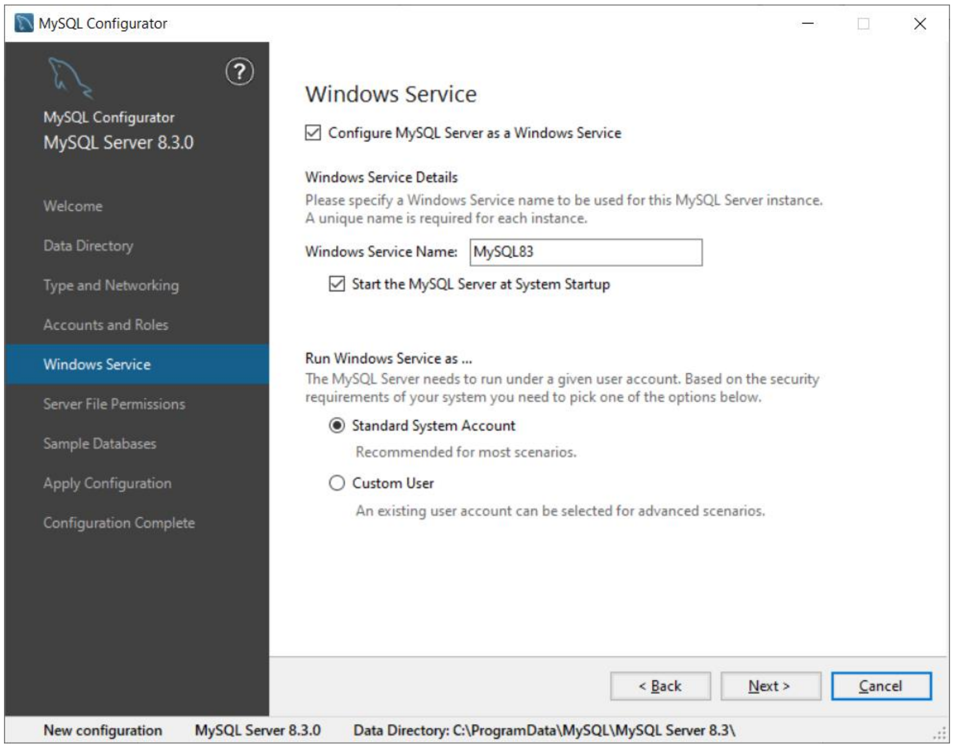
MySQL User Accounts

Create MySQL user accounts for your users and applications. Assign a role to the user that consists of a set of privileges.

MySQL User Name	Host	User Role

New configuration MySQL Server 8.3.0 Data Directory: C:\ProgramData\MySQL\MySQL Server 8.3\

Configure MySQL as a service as follows:


MySQL Configurator

Windows Service

[✓] Configure MySQL Server as a Windows Service

Windows Service Details

Please specify a Windows Service name to be used for this MySQL Server instance.
A unique name is required for each instance.

Windows Service Name: MySQL83

[✓] Start the MySQL Server at System Startup

Run Windows Service as ...

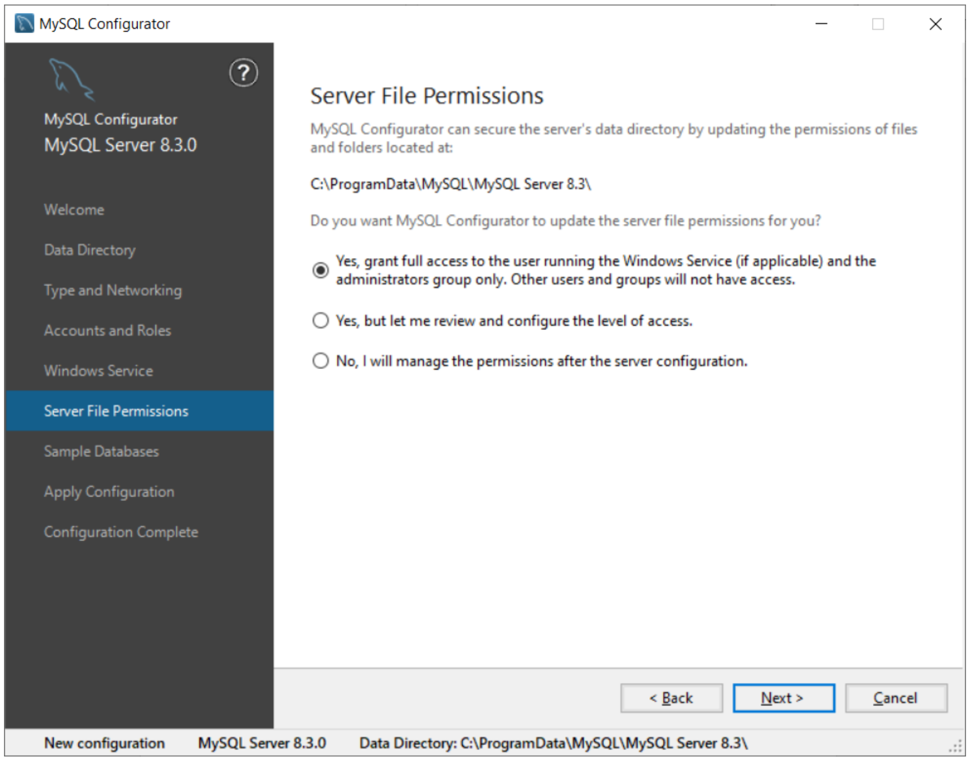
The MySQL Server needs to run under a given user account. Based on the security requirements of your system you need to pick one of the options below.

[✓ Standard System Account]
Recommended for most scenarios.

Custom User
An existing user account can be selected for advanced scenarios.

New configuration MySQL Server 8.3.0 Data Directory: C:\ProgramData\MySQL\MySQL Server 8.3\

Grant rights to the users as follows:


MySQL Configurator

Server File Permissions

MySQL Configurator can secure the server's data directory by updating the permissions of files and folders located at:

C:\ProgramData\MySQL\MySQL Server 8.3\

Do you want MySQL Configurator to update the server file permissions for you?

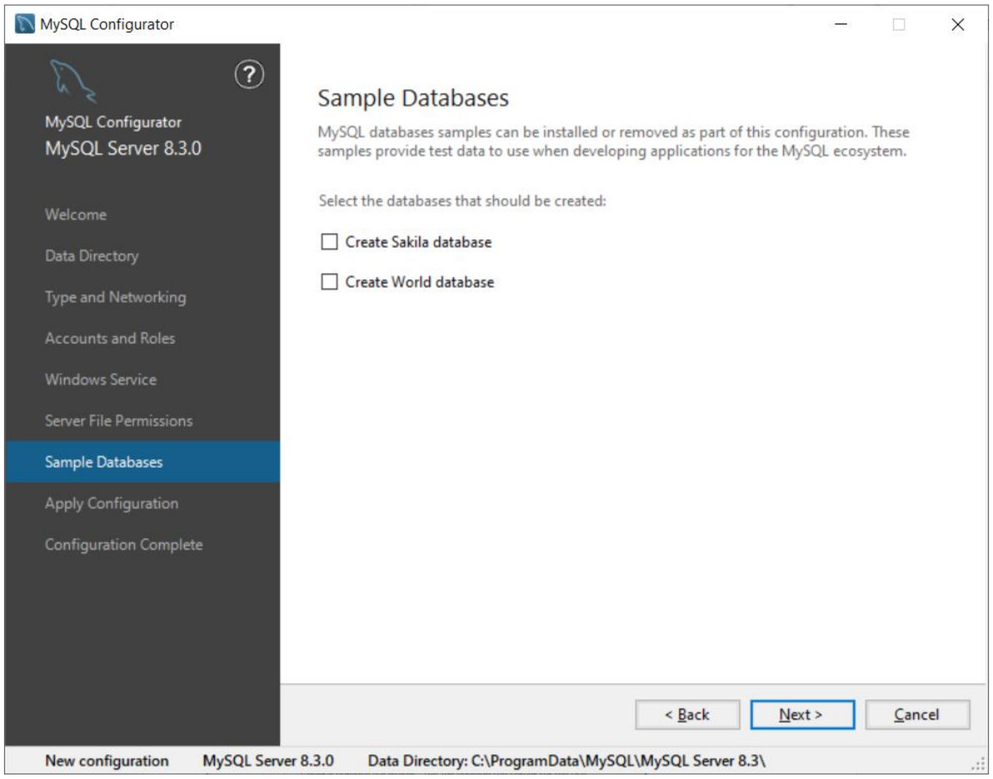
[✓ Yes, grant full access to the user running the Windows Service (if applicable) and the administrators group only. Other users and groups will not have access.]

Yes, but let me review and configure the level of access.

No, I will manage the permissions after the server configuration.

New configuration MySQL Server 8.3.0 Data Directory: C:\ProgramData\MySQL\MySQL Server 8.3\

You may choose to install sample databases or not:



Sample Databases

MySQL databases samples can be installed or removed as part of this configuration. These samples provide test data to use when developing applications for the MySQL ecosystem.

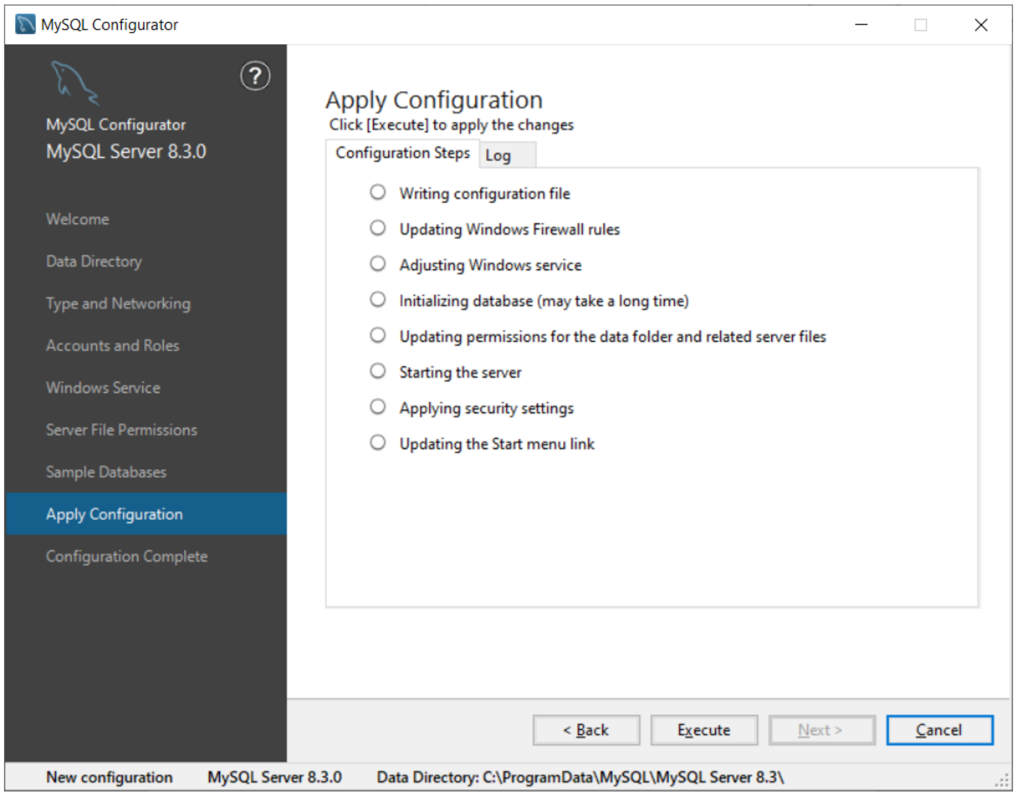
Select the databases that should be created:

☐ Create Sakila database

☐ Create World database

New configuration MySQL Server 8.3.0 Data Directory: C:\ProgramData\MySQL\MySQL Server 8.3\

In the final window wait for the setup program to complete all the points as follows:


MySQL Configurator

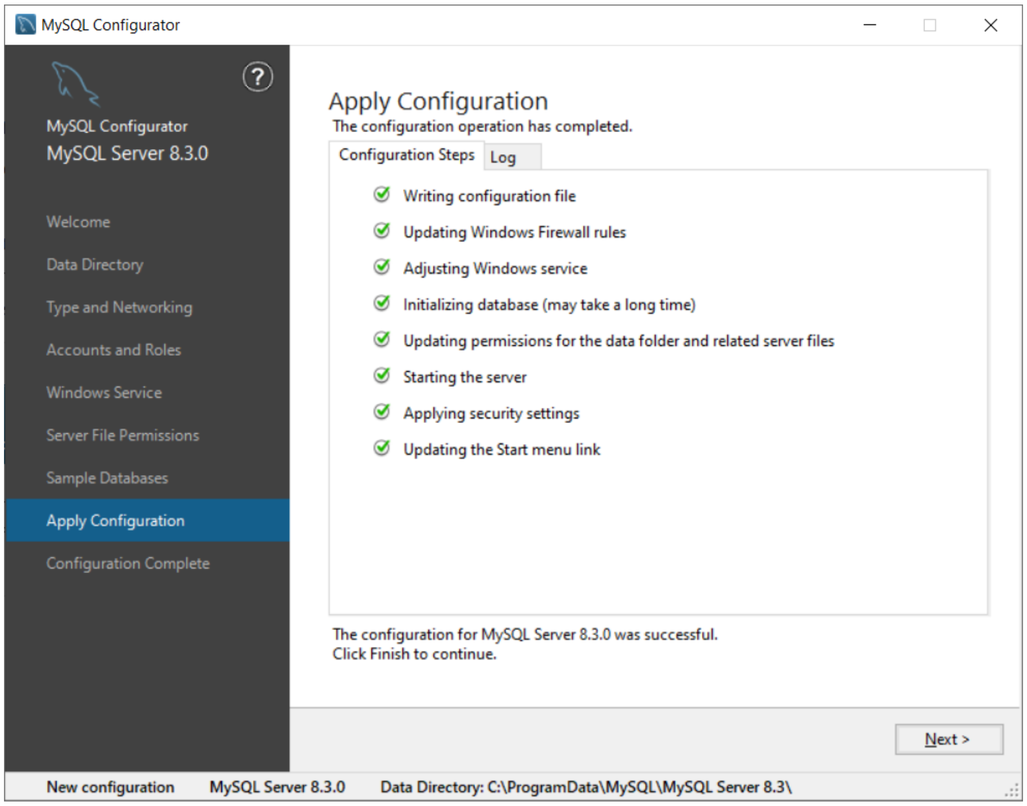
MySQL Configurator
MySQL Server 8.3.0

Apply Configuration
Click [Execute] to apply the changes

- Writing configuration file
- Updating Windows Firewall rules
- Adjusting Windows service
- Initializing database (may take a long time)
- Updating permissions for the data folder and related server files
- Starting the server
- Applying security settings
- Updating the Start menu link

New configuration MySQL Server 8.3.0 Data Directory: C:\ProgramData\MySQL\MySQL Server 8.3\

If all steps were executed correctly, the following window should appear:



Apply Configuration

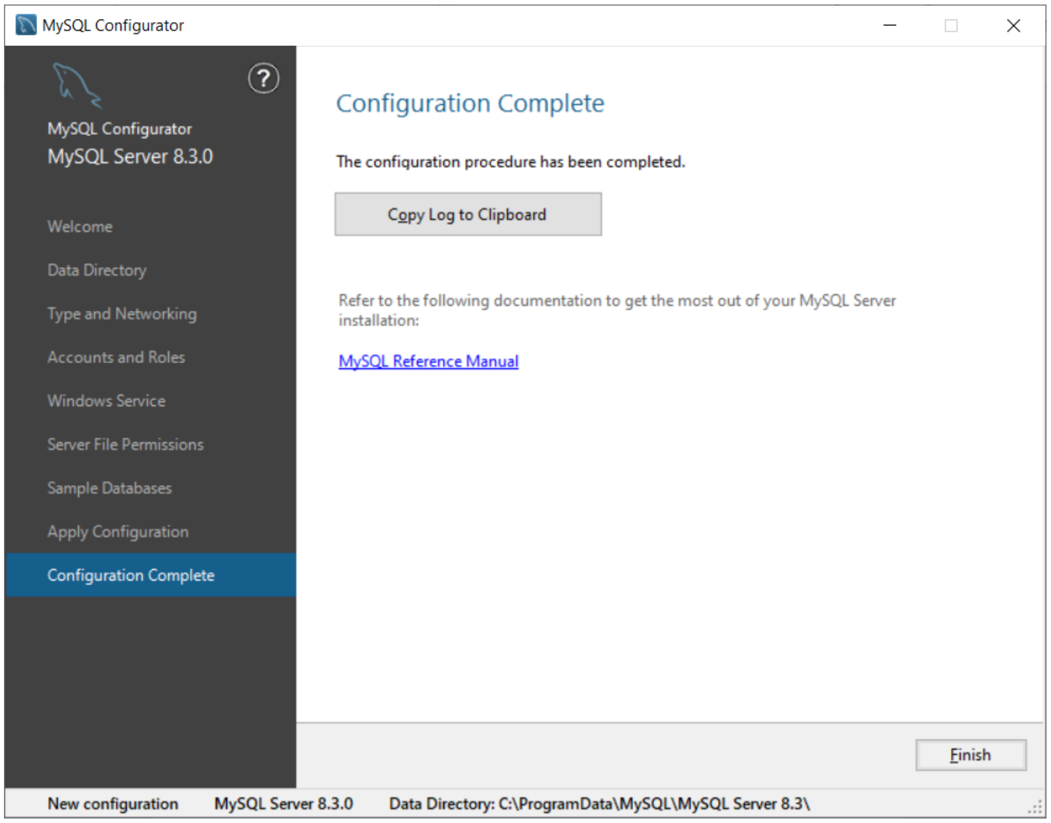
The configuration operation has completed.

- ✓ Writing configuration file
- ✓ Updating Windows Firewall rules
- ✓ Adjusting Windows service
- ✓ Initializing database (may take a long time)
- ✓ Updating permissions for the data folder and related server files
- ✓ Starting the server
- ✓ Applying security settings
- ✓ Updating the Start menu link

The configuration for MySQL Server 8.3.0 was successful.
Click Finish to continue.

Data Directory: C:\ProgramData\MySQL\MySQL Server 8.3\

You may want to copy the log of the setup procedure as follows:


MySQL Configurator

Configuration Complete

The configuration procedure has been completed.

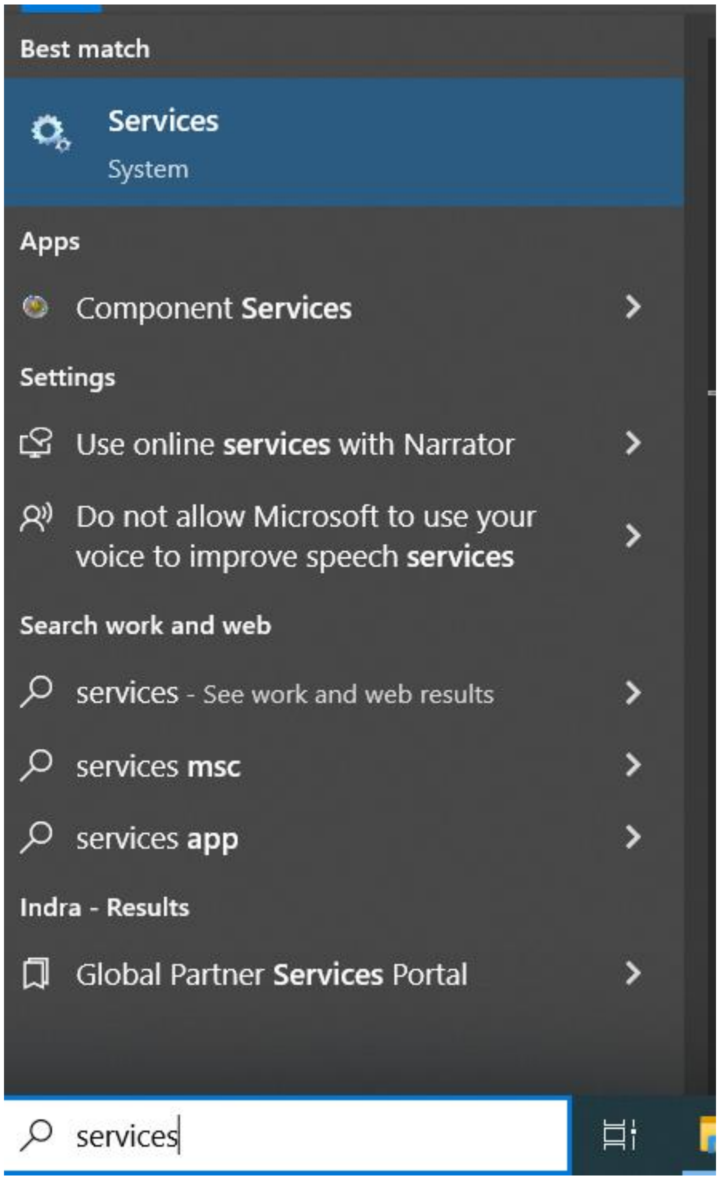
Refer to the following documentation to get the most out of your MySQL Server installation:

MySQL Reference Manual

New configuration MySQL Server 8.3.0 Data Directory: C:\ProgramData\MySQL\MySQL Server 8.3\

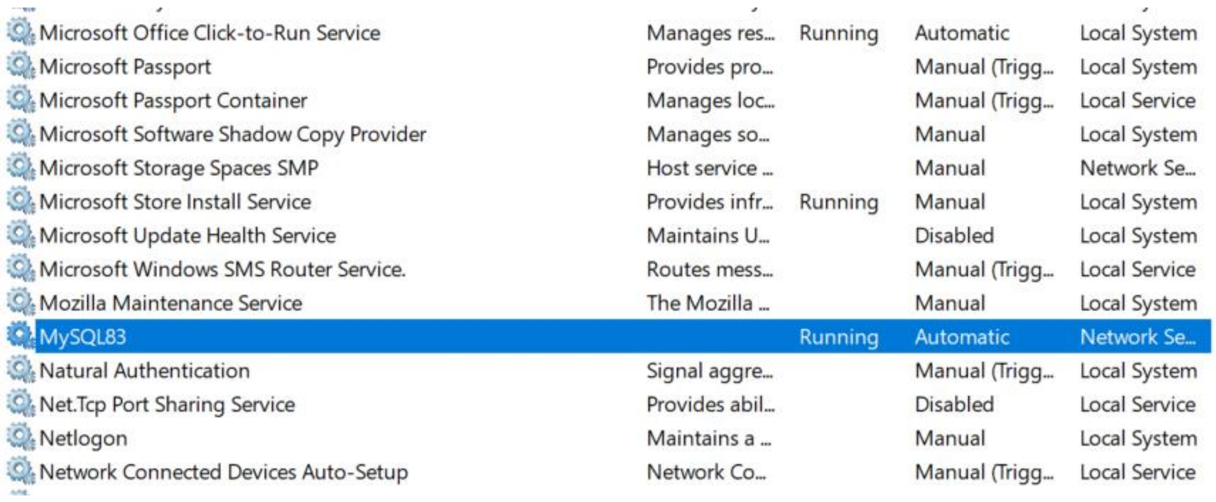
To verify that all has been setup correctly and the service is working fine go to Services and check the MySQL service as follows:

Type Services in the search bar of Windows:


Best match
Services
System
Apps
Component Services
Settings
Use online services with Narrator
Do not allow Microsoft to use your voice to improve speech services
Search work and web
services - See work and web results
services msc
services app
Indra - Results
Global Partner Services Portal
services

Go to Services:

Check that MySQL 8.3 is running as a service:



Microsoft Office Click-to-Run Service	Manages res...	Running	Automatic	Local System
Microsoft Passport	Provides pro...		Manual (Trigg...	Local System
Microsoft Passport Container	Manages loc...		Manual (Trigg...	Local Service
Microsoft Software Shadow Copy Provider	Manages so...		Manual	Local System
Microsoft Storage Spaces SMP	Host service ...		Manual	Network Se...
Microsoft Store Install Service	Provides infr...	Running	Manual	Local System
Microsoft Update Health Service	Maintains U...		Disabled	Local System
Microsoft Windows SMS Router Service.	Routes mess...		Manual (Trigg...	Local Service
Mozilla Maintenance Service	The Mozilla ...		Manual	Local System
MySQL83		Running	Automatic	Network Se...
Natural Authentication	Signal aggre...		Manual (Trigg...	Local System
Net.Tcp Port Sharing Service	Provides abil...		Disabled	Local Service
Netlogon	Maintains a ...		Manual	Local System
Network Connected Devices Auto-Setup	Network Co...		Manual (Trigg...	Local Service

Install the MySQL Workbench.

You can create the database in two ways:

1. By commands in the MySQL console
2. By graphical user interface in MySQL Workbench

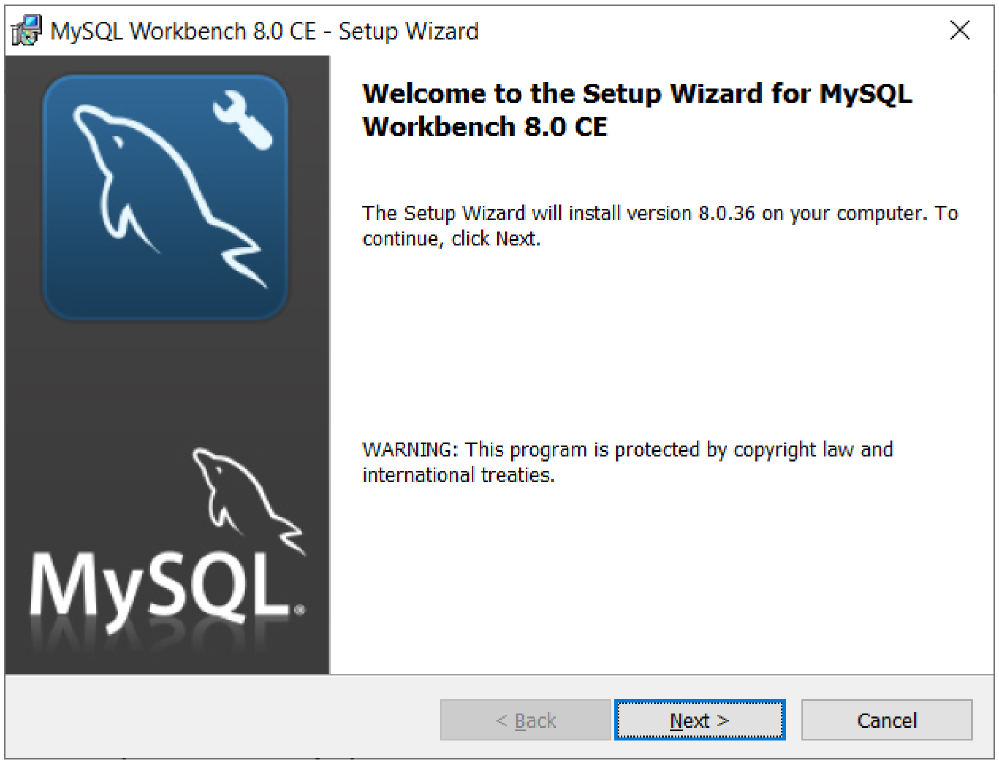
Download MySQL Workbench

MySQL Workbench 8.0.36

https://dev.mysql.com/downloads/workbench/

Run the setup file

Continue with next on the following window:

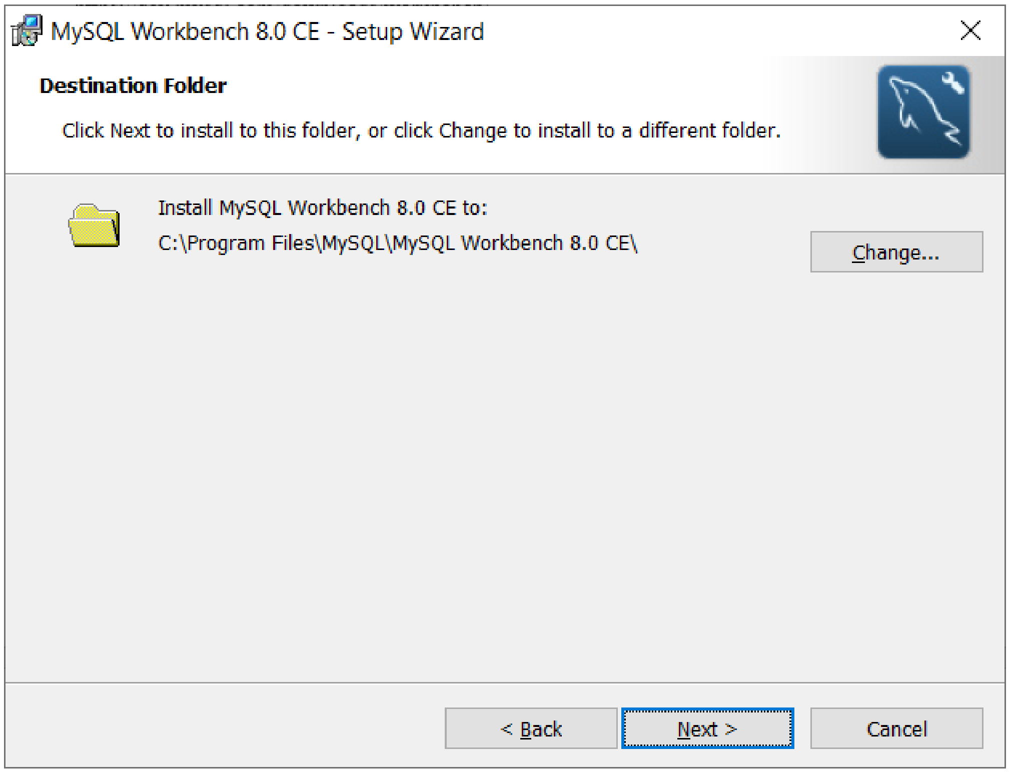

MySQL Workbench 8.0 CE - Setup Wizard

Welcome to the Setup Wizard for MySQL Workbench 8.0 CE

The Setup Wizard will install version 8.0.36 on your computer. To continue, click Next.

WARNING: This program is protected by copyright law and international treaties.

Choose a setup folder:

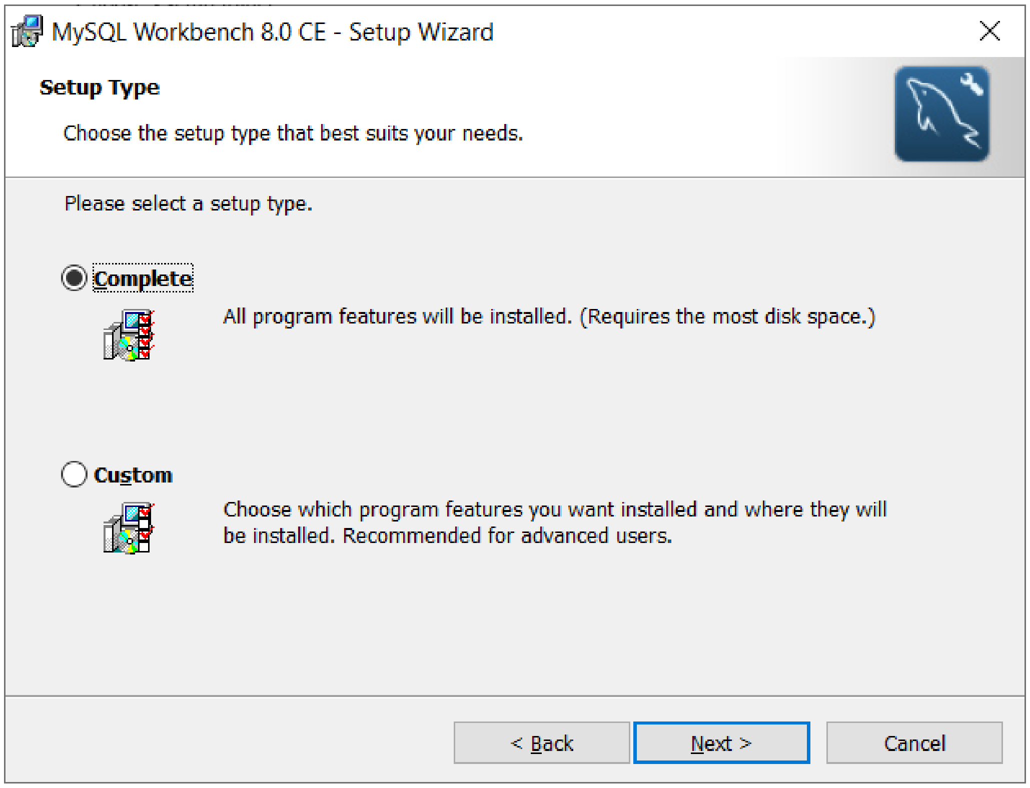

MySQL Workbench 8.0 CE - Setup Wizard

Destination Folder

Click Next to install to this folder, or click Change to install to a different folder.

Install MySQL Workbench 8.0 CE to:
C:\Program Files\MySQL\MySQL Workbench 8.0 CE\

Choose Custom in the following window:


MySQL Workbench 8.0 CE - Setup Wizard

Setup Type

Choose the setup type that best suits your needs.

Please select a setup type.

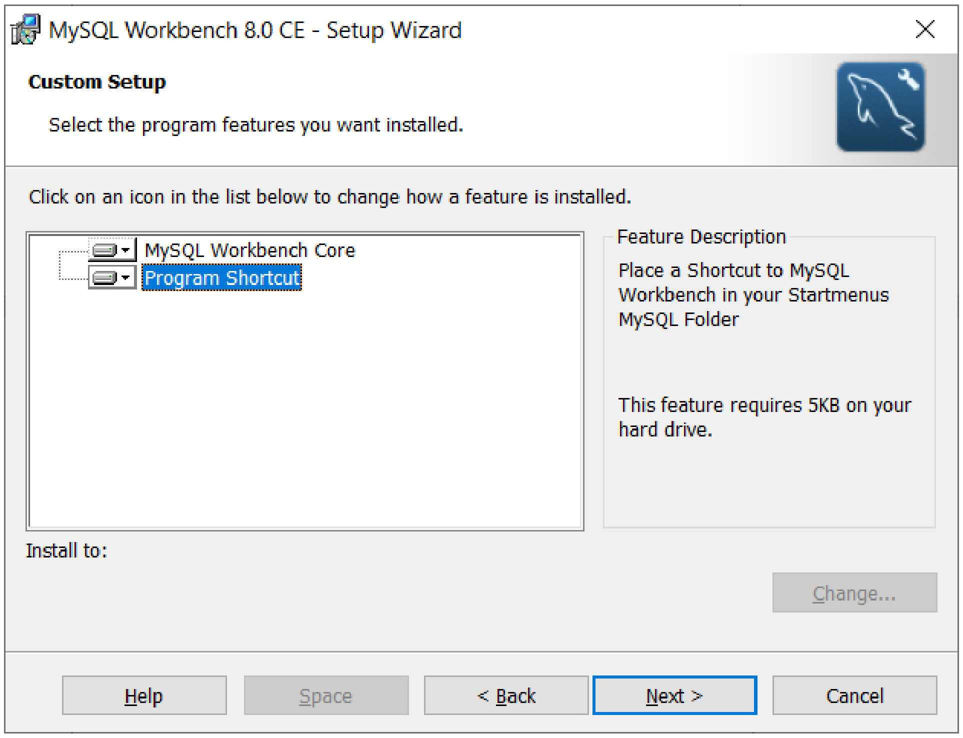
● Complete

All program features will be installed. (Requires the most disk space.)

○ Custom

Choose which program features you want installed and where they will be installed. Recommended for advanced users.

Choose to install all from local hard drive in the following window:


MySQL Workbench 8.0 CE - Setup Wizard

Custom Setup

Select the program features you want installed.

Click on an icon in the list below to change how a feature is installed.

MySQL Workbench Core
Program Shortcut

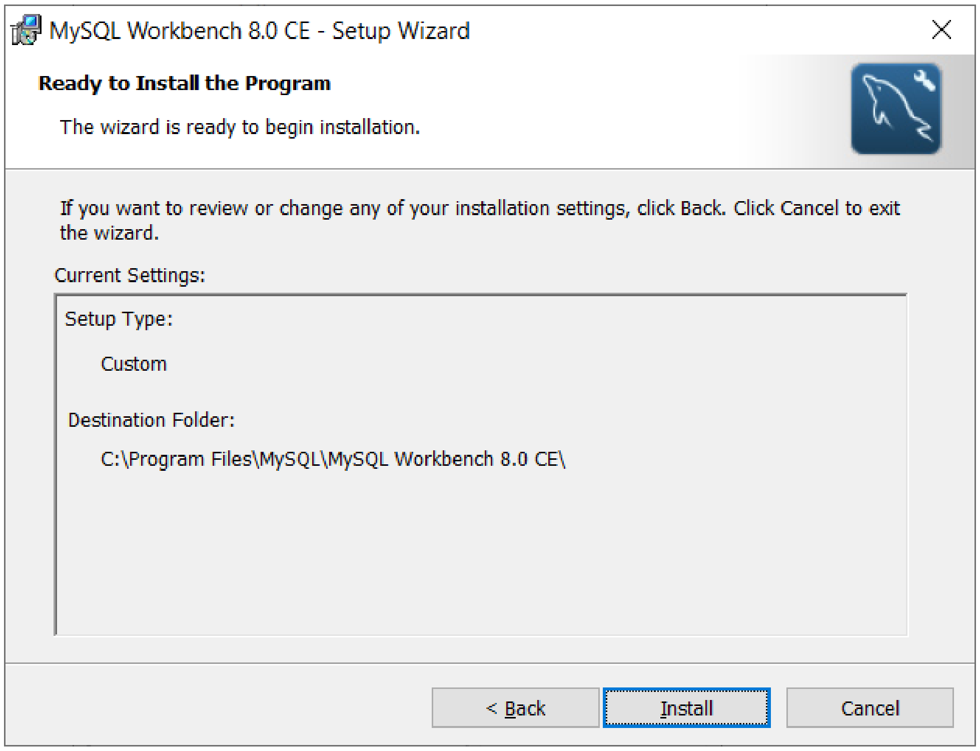
Feature Description

Place a Shortcut to MySQL Workbench in your Startmenus MySQL Folder

This feature requires 5KB on your hard drive.

Install to:

Continue with Install in the next window:


MySQL Workbench 8.0 CE - Setup Wizard

Ready to Install the Program

The wizard is ready to begin installation.

If you want to review or change any of your installation settings, click Back. Click Cancel to exit the wizard.

Current Settings:

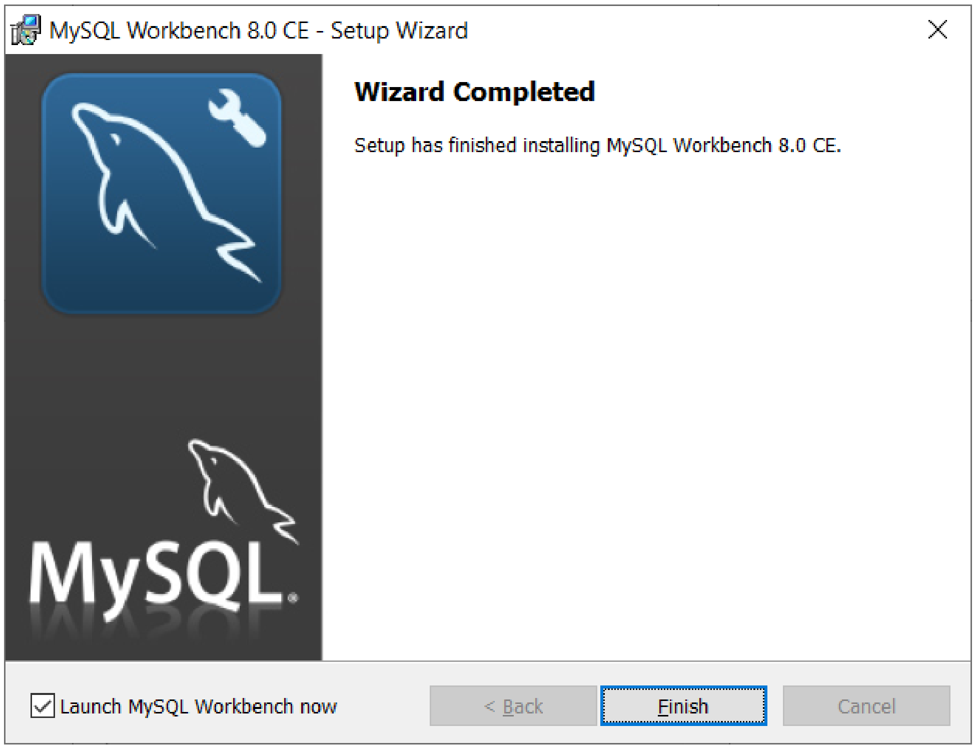
Setup Type:

Custom

Destination Folder:

C:\Program Files\MySQL\MySQL Workbench 8.0 CE\

The setup program will complete the installation procedure and you should see the following window:

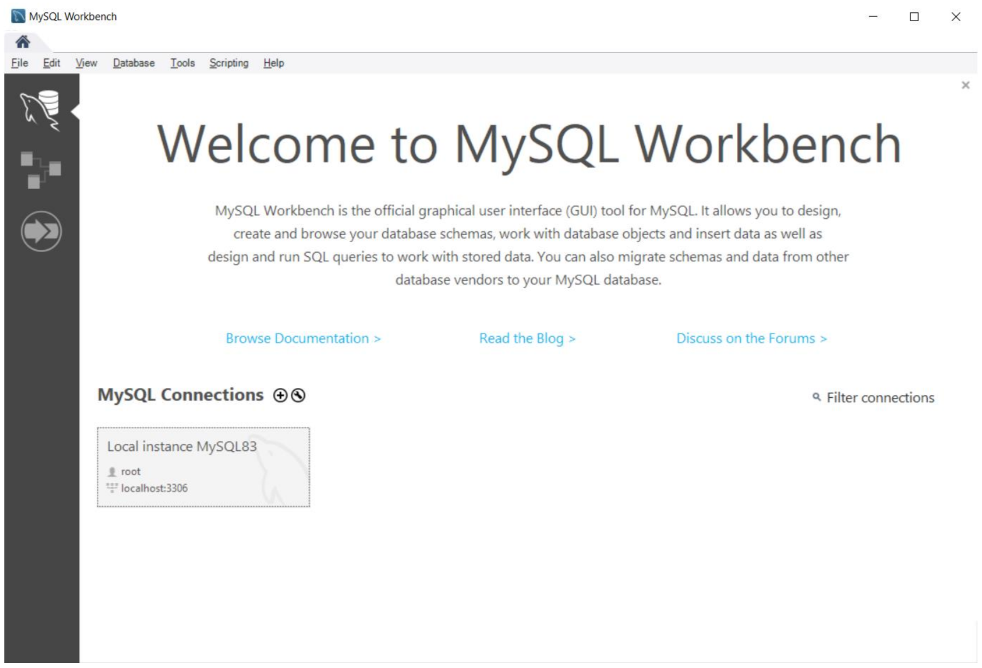

MySQL Workbench 8.0 CE - Setup Wizard

Wizard Completed

Setup has finished installing MySQL Workbench 8.0 CE.

☑ Launch MySQL Workbench now

Click Finish and the program will start as follows:



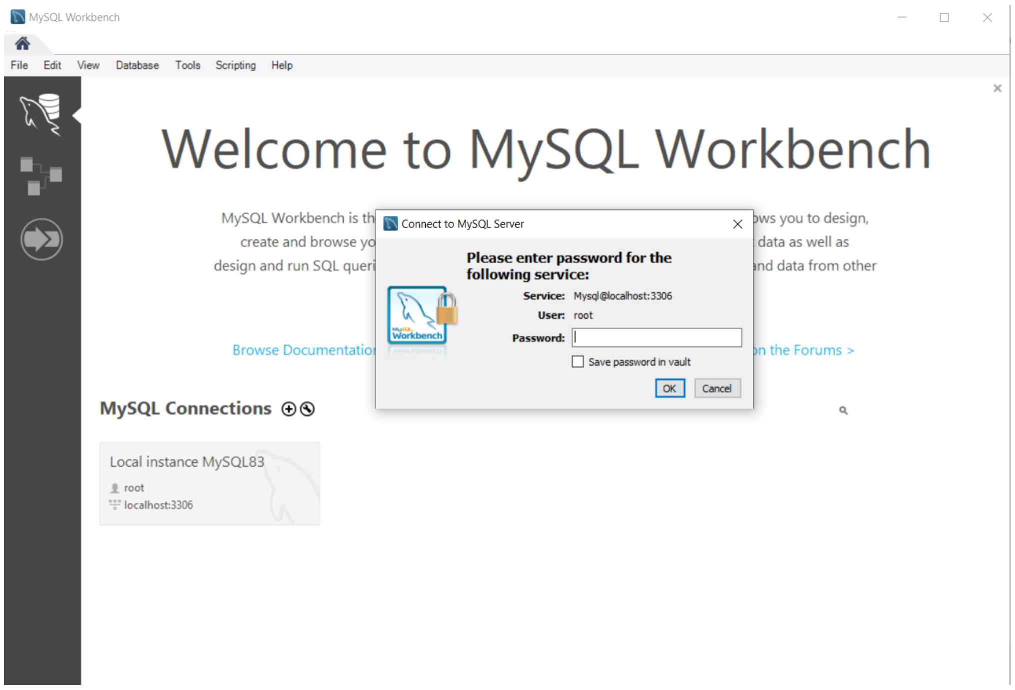
Welcome to MySQL Workbench

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.

MySQL Connections

Local instance MySQL83
root
localhost:3306

If you click on Local instance you will see the following:



Welcome to MySQL Workbench

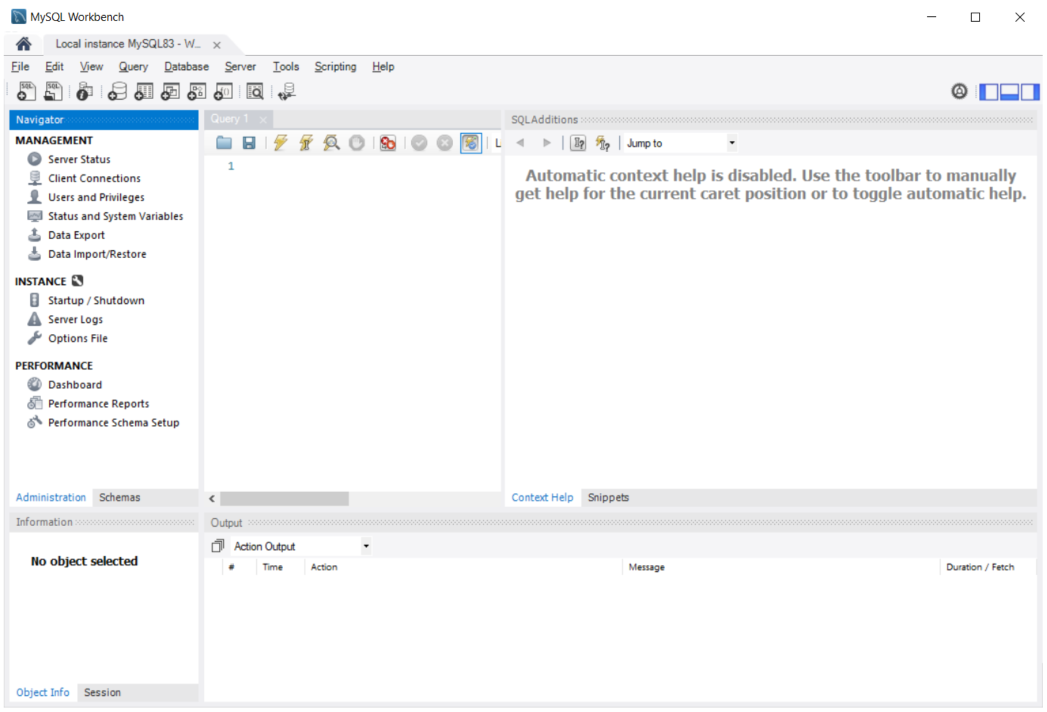
Please enter password for the following service:

Service: Mysql@localhost:3306
User: root
Password:

MySQL Connections

Local instance MySQL83
root
localhost:3306

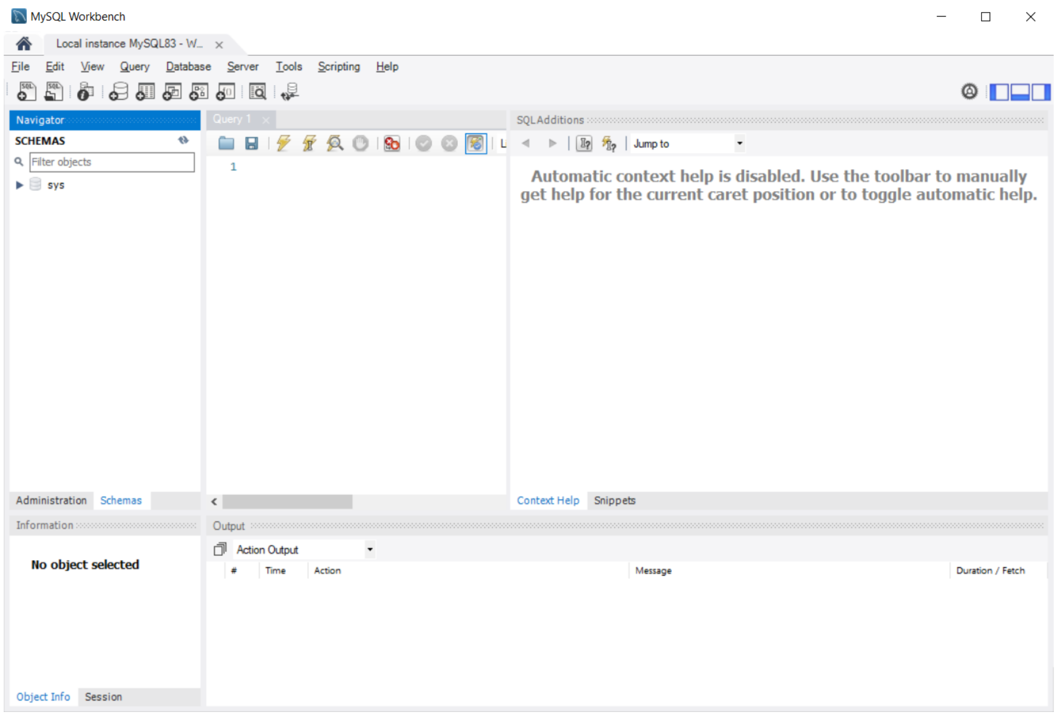
Insert the root password we used during installation of MySQL. You should see the following window:



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

No object selected

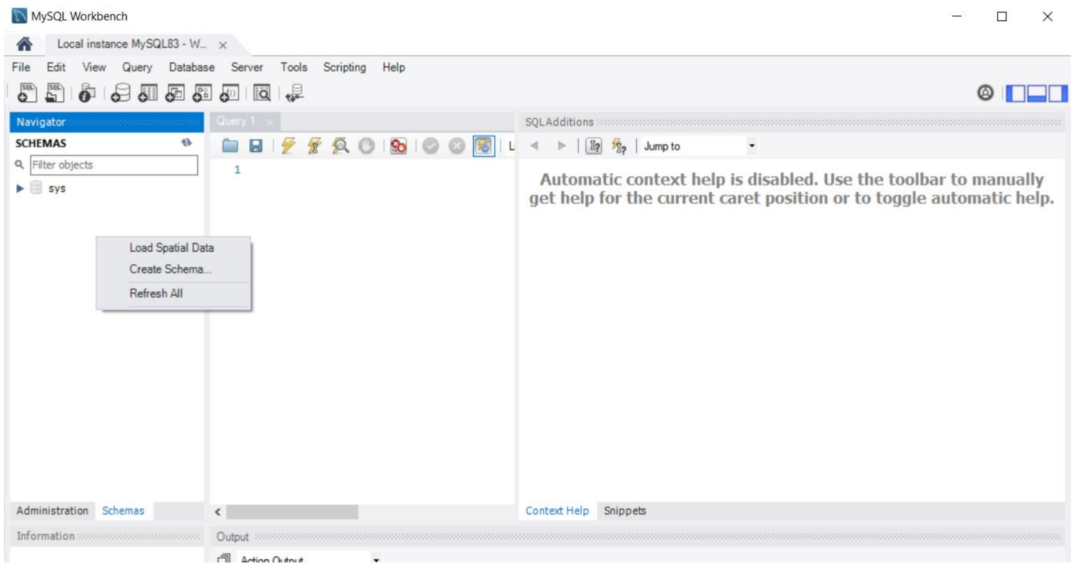
On the left panel, click on Schemas:



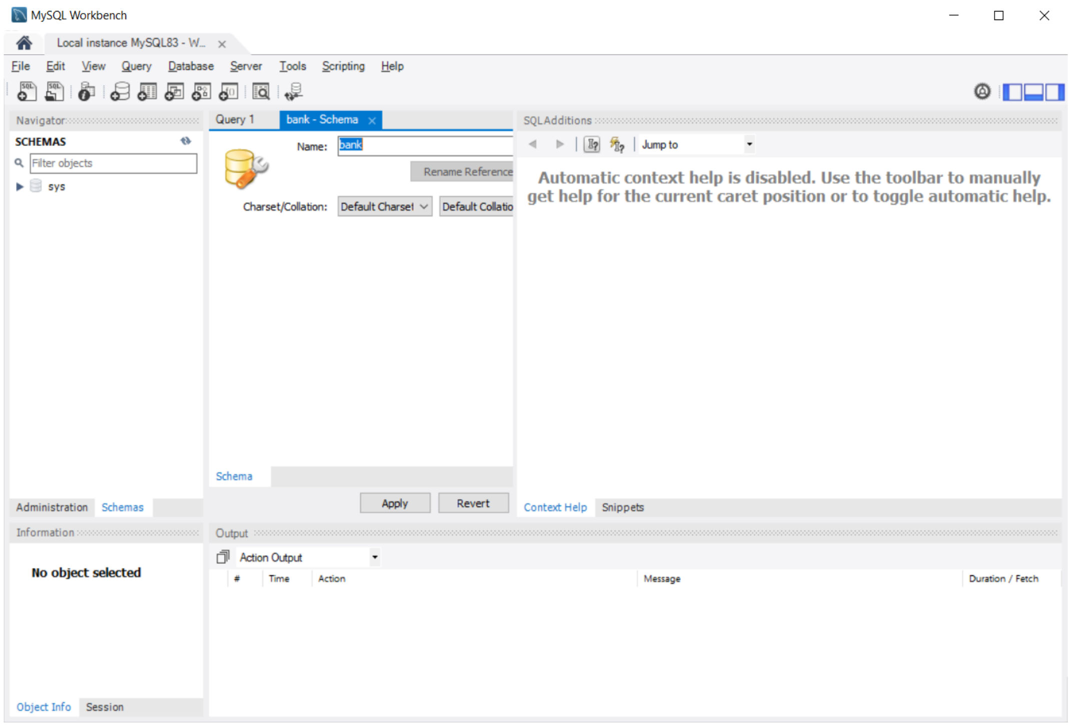
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

No object selected

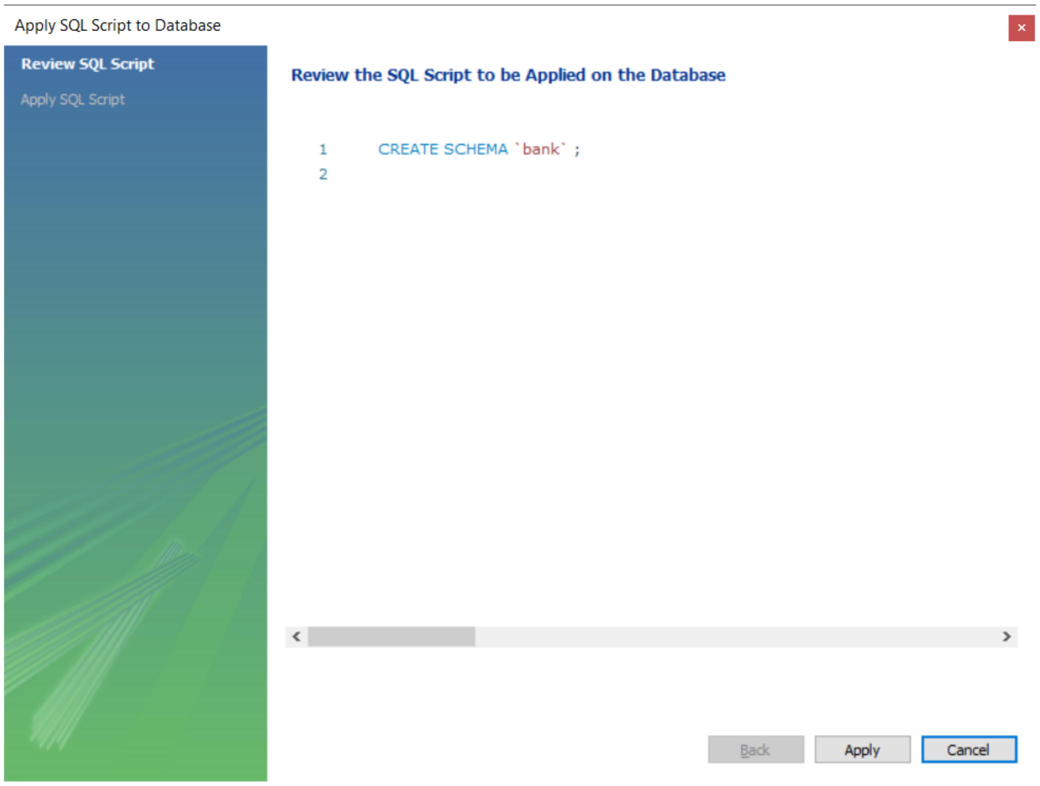
Click with the right button of the mouse on the left panel and select create schema:


MySQL Workbench
Local instance MySQL83 - W...
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
sys
Query 1
Load Spatial Data
Create Schema...
Refresh All
SQLAdditions
Jump to
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.
Administration
Schemas
Context Help
Snippets
Information
Output
Action Output

Name the database as Bank as follows:

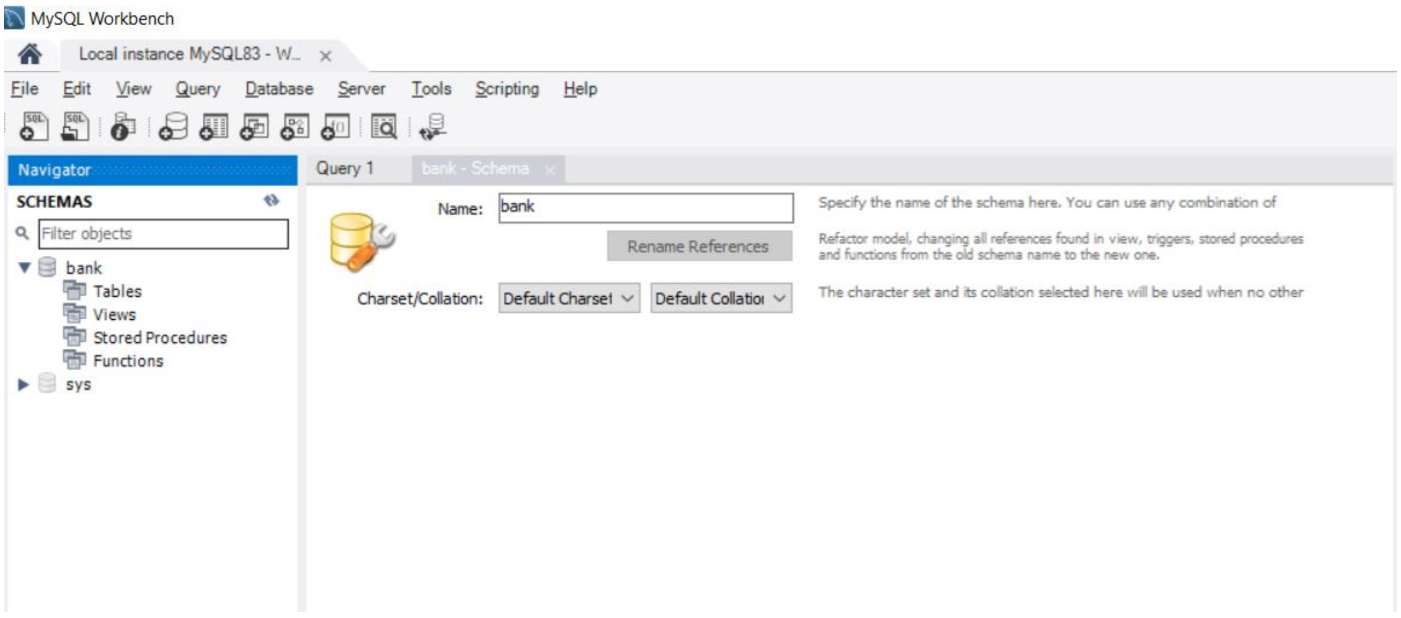

MySQL Workbench
Local instance MySQL83 - W...
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
sys
Query 1
bank - Schema
Name: bank
Rename References
Charset/Collation: Default Charset Default Collation
Schema
Apply
Revert
SQLAdditions
Jump to
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.
Administration
Schemas
Context Help
Snippets
Information
No object selected
Output
Action Output
# Time Action Message Duration / Fetch
Object Info
Session

Click Apply in the following window:


Apply SQL Script to Database
Review SQL Script
Apply SQL Script
Review the SQL Script to be Applied on the Database
Back
Apply
Cancel

```
CREATE SCHEMA `bank` ;
```

In the next window, click Finish and you will see the following:



Name: bank

Charset/Collation: Default Charset Default Collation

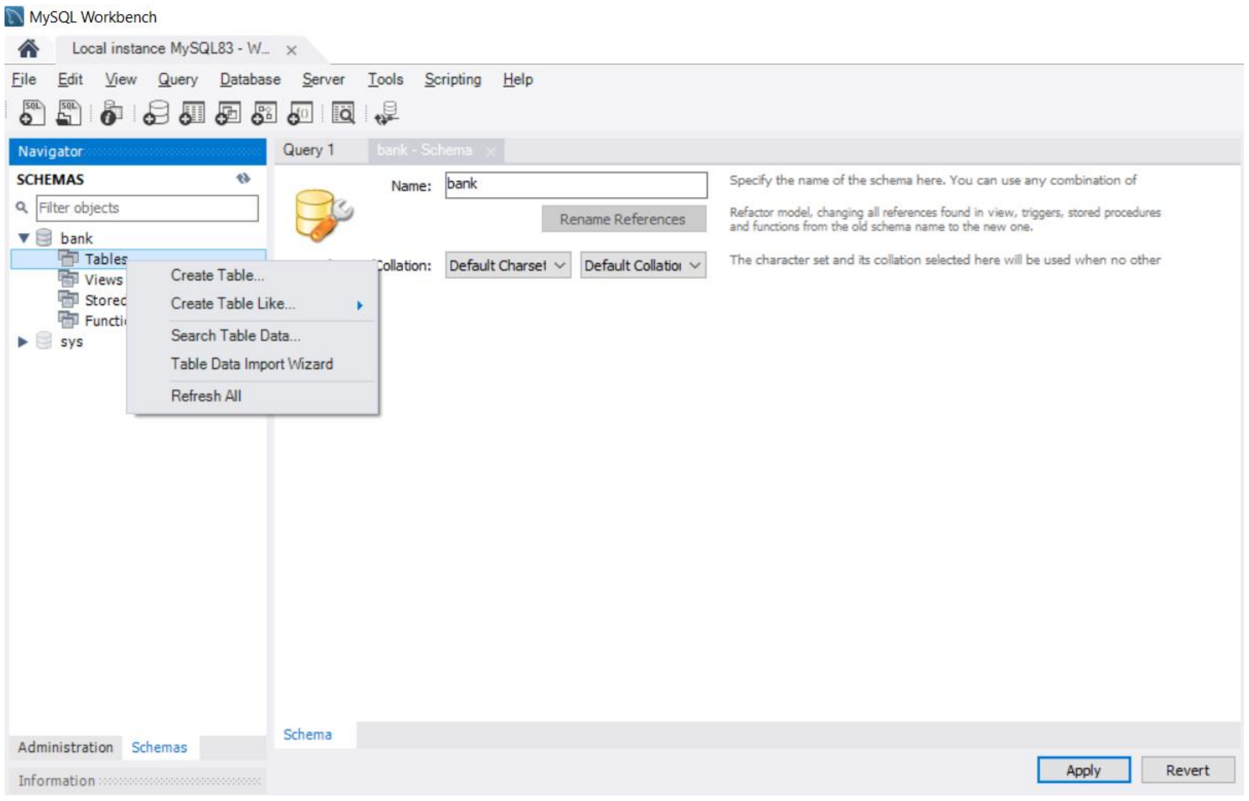
Specify the name of the schema here. You can use any combination of

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

Go the tables and create three tables with the following data:

Table Account
Fields: IdAccount (int), Balance (float)



Name: bank

Collation: Default Charset | Default Collation

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

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

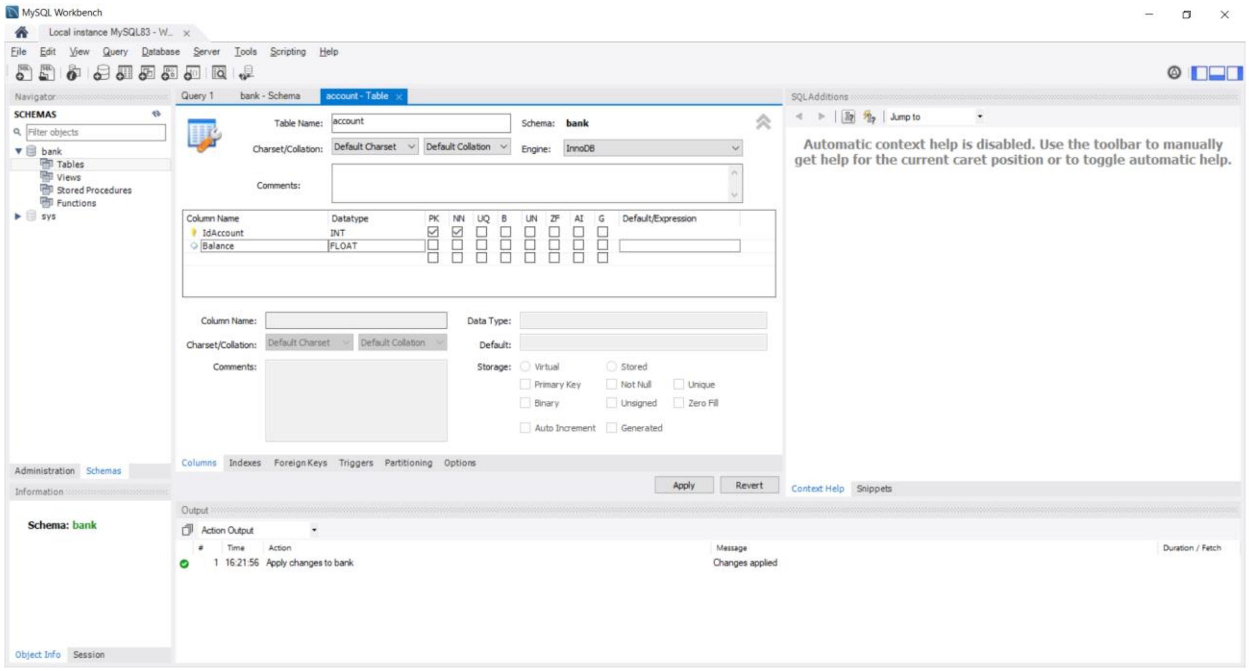


Table Name: account Schema: **bank**

Charset/Collation: Default Charset | Default Collation Engine: InnoDB

Comments:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
IdAccount	INT	☑	☑	☐	☐	☐	☐	☐	☐	
Balance	FLOAT	☐	☐	☐	☐	☐	☐	☐	☐	

Column Name: Data Type:

Charset/Collation: Default Charset | Default Collation Default:

Comments: Storage: ○ Virtual ○ Stored

☐ Primary Key ☐ Not Null ☐ Unique

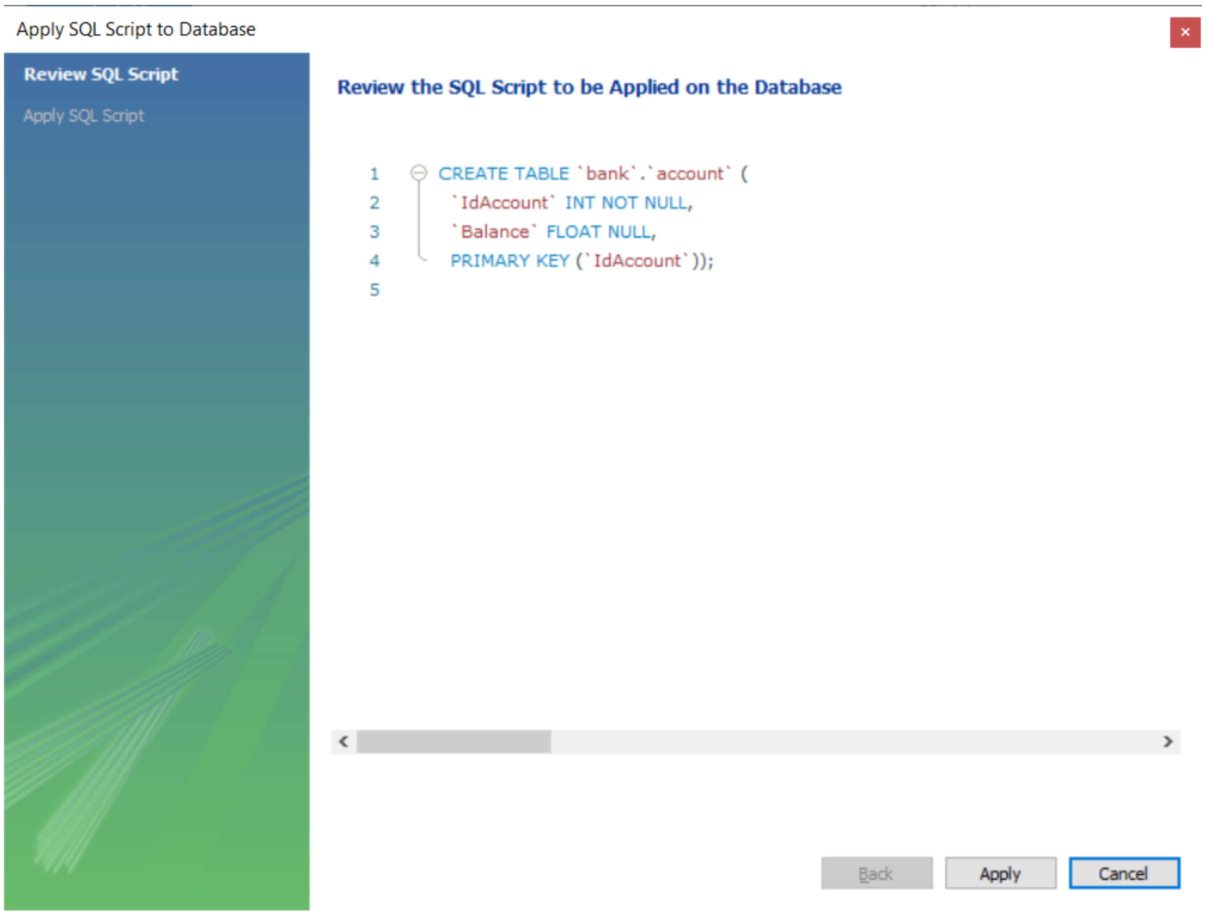
☐ Binary ☐ Unsigned ☐ Zero Fill

☐ Auto Increment ☐ Generated

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

Schema: bank

```
#  Time      Action                  Message          Duration / Fetch
1  16:21:56  Apply changes to bank   Changes applied
```



Review the SQL Script to be Applied on the Database

```sql
CREATE TABLE `bank`.`account` (
  `IdAccount` INT NOT NULL,
  `Balance` FLOAT NULL,
  PRIMARY KEY (`IdAccount`));
```

Table Customer
Fields: IdCustomer(int), Name (Varchar), Surname (Varchar)

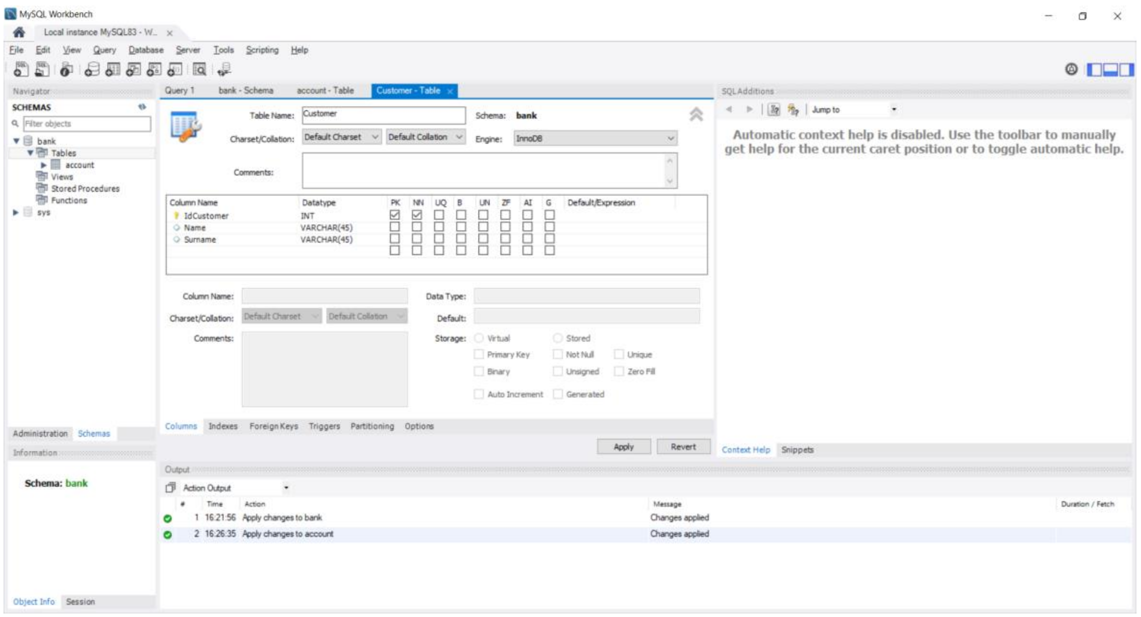


Table Name: Customer
Schema: bank
Charset/Collation: Default Charset, Default Collation
Engine: InnoDB
Comments:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
IdCustomer	INT	✓	✓							
Name	VARCHAR(45)									
Surname	VARCHAR(45)									

Column Name:
Data Type:
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 to manually get help for the current caret position or to toggle automatic help.

Schema: bank

Output
Action Output

#	Time	Action	Message	Duration / Fetch
1	16:21:56	Apply changes to bank	Changes applied	
2	16:26:35	Apply changes to account	Changes applied	

Table AccountCustomer
Fields: IdAccount, IdCustomer

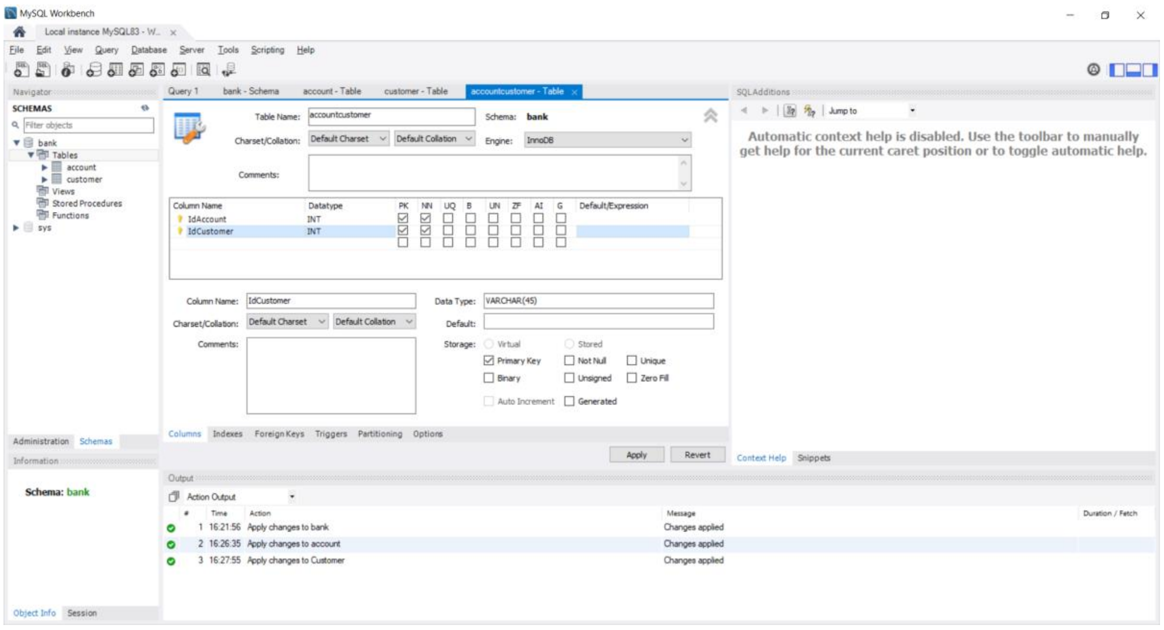


Table Name: accountcustomer
Schema: bank
Charset/Collation: Default Charset, Default Collation
Engine: InnoDB

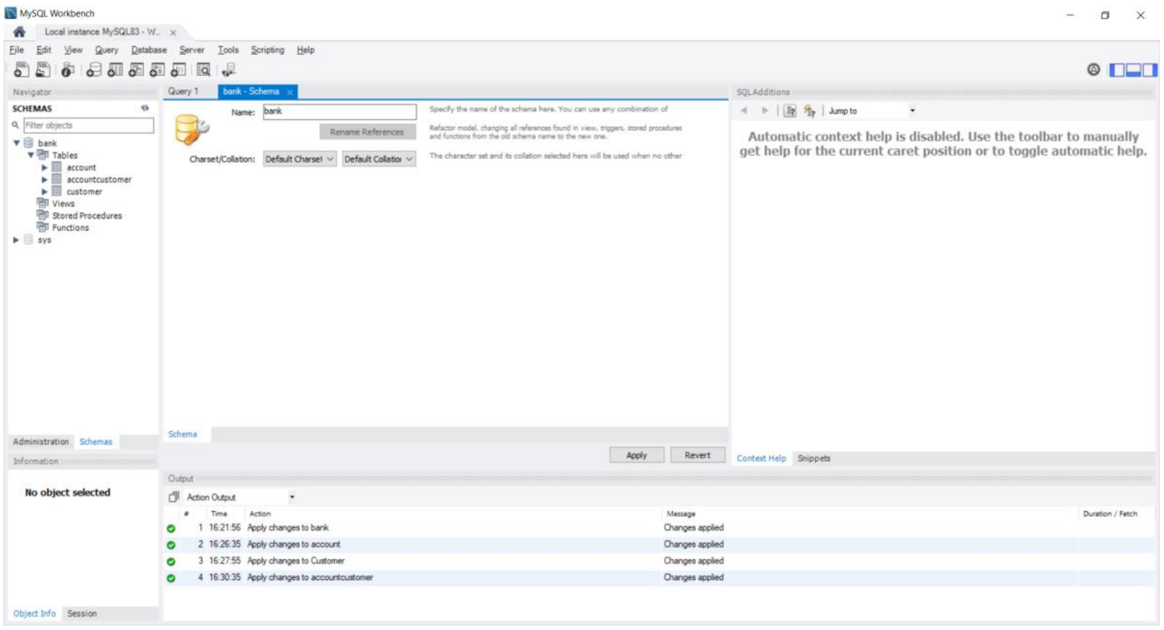
Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
IdAccount	INT	✓	✓							
IdCustomer	INT	✓	✓							

Column Name: IdCustomer
Data Type: VARCHAR(45)
Storage: [✓ Primary Key]

```
#  Time      Action                      Message
1  16:21:56  Apply changes to bank       Changes applied
2  16:26:35  Apply changes to account    Changes applied
3  16:27:55  Apply changes to Customer   Changes applied
```

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

Your database should now have three tables as follows:



SCHEMAS
- bank
  - Tables
    - account
    - accountcustomer
    - customer
  - Views
  - Stored Procedures
  - Functions
- sys

Name: bank
Charset/Collation: Default Charset, Default Collation

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

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

```
#  Time      Action                            Message
1  16:21:56  Apply changes to bank             Changes applied
2  16:26:35  Apply changes to account          Changes applied
3  16:27:55  Apply changes to Customer         Changes applied
4  16:30:35  Apply changes to accountcustomer  Changes applied
```

Download Apache Netbeans IDE 21 and setup the full program with all features on the local drive.

2. Developing the RMI Client and Server

When you finish this exercise, you will have run your first RMI sytem.

It consists of three major parts:

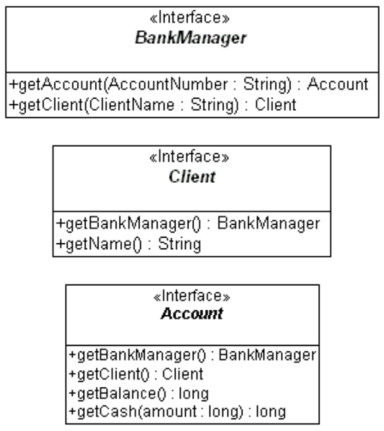
- The **RMI Registry** that hold references to the remote services.
- The **RMI host server** program that creates the remote services, registers them with the registry and waits for client requests.
- The **RMI client program.** A program that obtains references to remote service object from the RMI registry and then uses those services.

2.1 Fundamentals of RMI

This exercise will introduce you to the definition of RMI remote services using Java interfaces.

Educational goals:

- Introduce the UML Description of a banking system
- Complete the Java source code for the system interfaces


«Interface»
BankManager
+getAccount(AccountNumber : String) : Account
+getClient(ClientName : String) : Client
«Interface»
Client
+getBankManager() : BankManager
+getName() : String
«Interface»
Account
+getBankManager() : BankManager
+getClient() : Client
+getBalance() : long
+getCash(amount : long) : long

These are the interfaces to develop in the project.

File Account.java

```
import java.rmi.Remote;
import java.rmi.RemoteException;

public interface Account extends Remote
   {
   // Add method to return master BankManager
   // Add method to return Client of this account

   // Add method to return balance of this account

   // Add method to withdraw cash from this account
   }
```

File BankManager.java

```
import java.rmi.Remote;
import java.rmi.RemoteException;



public interface BankManager extends Remote
   {
   // Add method to return an Account service

   // Add method to return a Client service
   }
```

File Client.java

```
import java.rmi.Remote;
import java.rmi.RemoteException;

public interface Client extends Remote
   {
   // Add method to return master BankManager

   // Add method to return the name of this client
   }
```

2.2 Development of a Simple Banking System

In this part you will run your first RMI system. It is based on the Banking System that you started in the previous lab session.

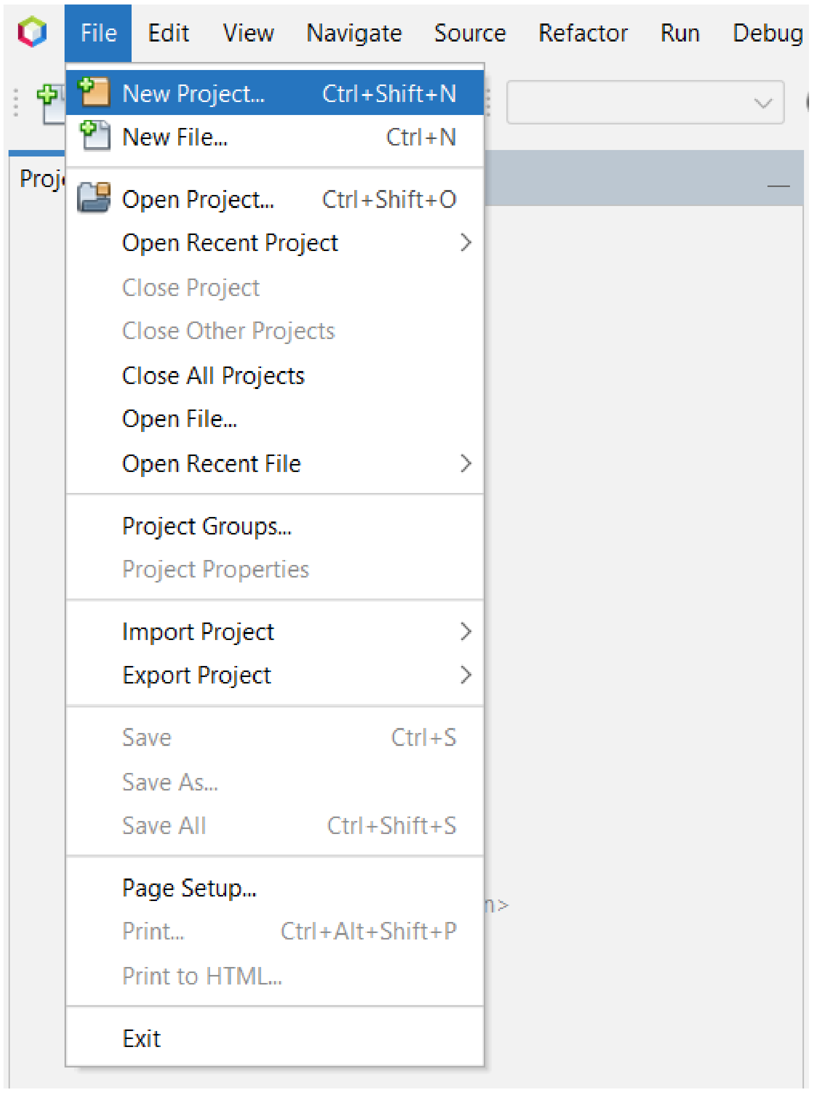
Educational goals:

- Run a server that starts the RMI Registry and supports remote RMI objects
- Implement an RMI client that uses remote services.

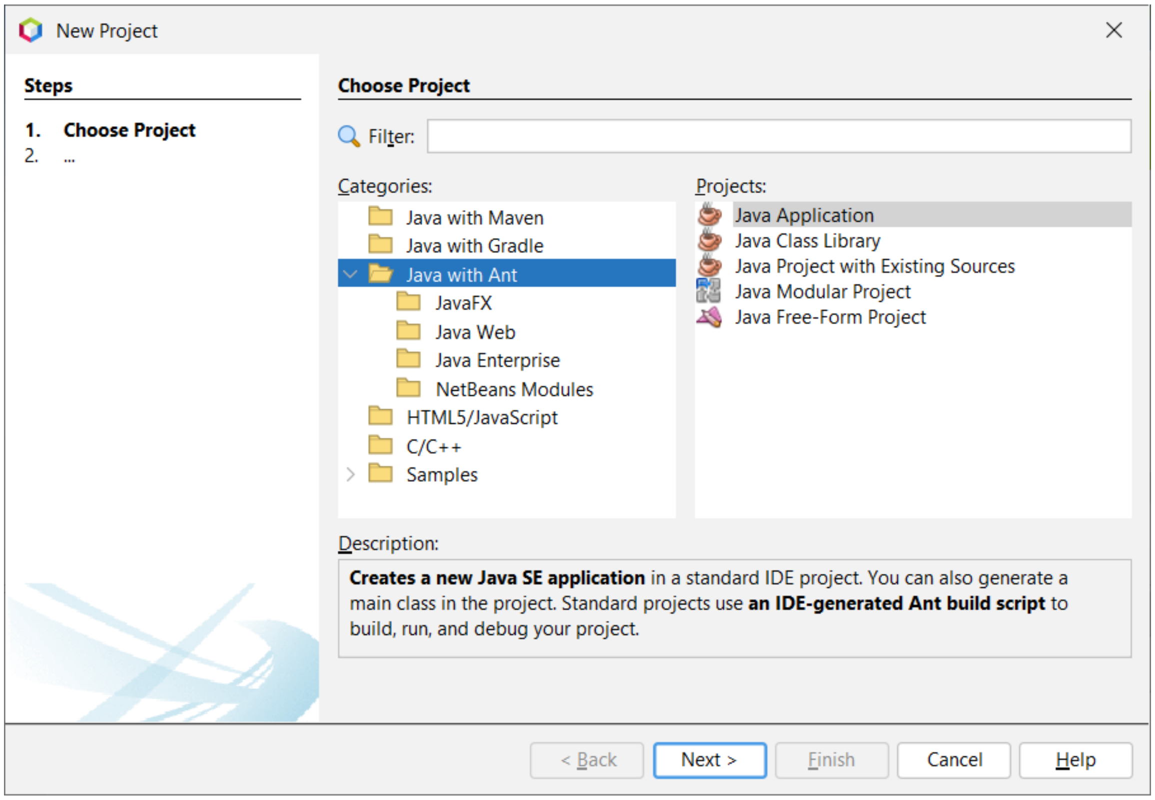
The **RMI Registry** manages the publication of the RMI remote services. You have to run a server program that creates the actual remote services, and finally, finish coding the program BankUser, which will use the RMI remote services.

Step 1: Code Development

Create a new project in NetBeans as follows, in File, select New Project.

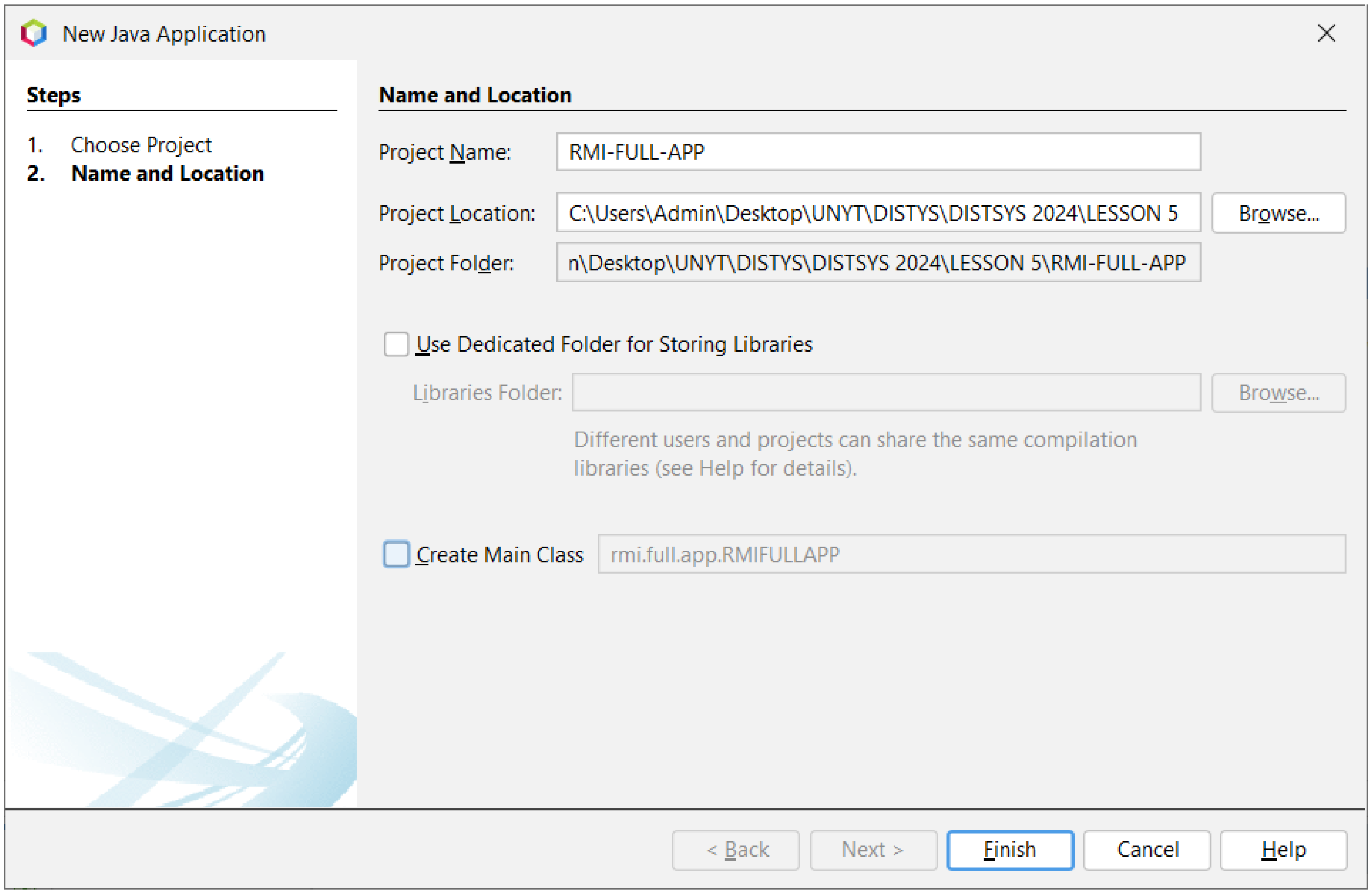


In the following window select Java with Ant and Java Application.



Creates a new Java SE application in a standard IDE project. You can also generate a main class in the project. Standard projects use **an IDE-generated Ant build script** to build, run, and debug your project.

Then enter the name of the project and the respective project folder as follows:


New Java Application

Steps

1. Choose Project
2. **Name and Location**

Name and Location

Project Name: RMI-FULL-APP

Project Location: C:\Users\Admin\Desktop\UNYT\DISTYS\DISTSYS 2024\LESSON 5

Project Folder: n\Desktop\UNYT\DISTYS\DISTSYS 2024\LESSON 5\RMI-FULL-APP

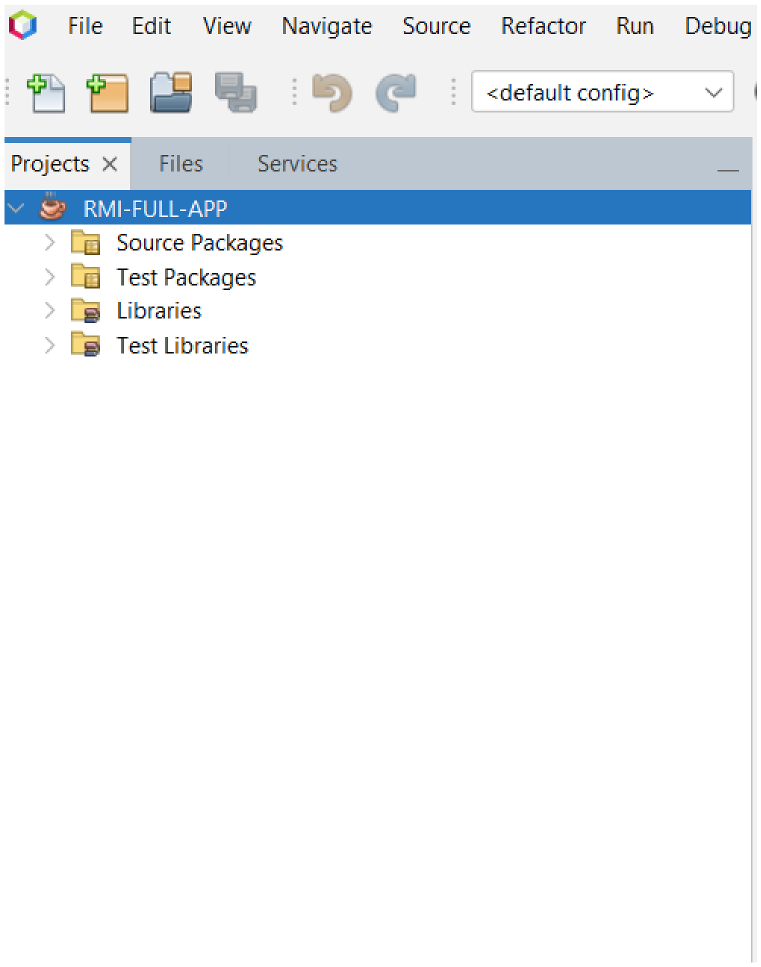
[ ] Use Dedicated Folder for Storing Libraries

Libraries Folder:

Different users and projects can share the same compilation libraries (see Help for details).

[ ] Create Main Class rmi.full.app.RMIFULLAPP

The project is now ready for code development as shown in the right panel as follows:



Develop the following code in Netbeans:

DbAccess.java

```
import java.sql.*;

public class DbAccess {

        private Connection conn;
        private Statement s;

        public boolean initializeConnection(String SERVER, String DATABASE, String USER_ID,
                        String PASSWORD) throws ClassNotFoundException, SQLException {
                try {
                        Class.forName("com.mysql.jdbc.Driver").newInstance();
                        String path = ("jdbc:mysql://"+ SERVER + "/" + DATABASE + "?user="
                                        + USER_ID + "&password=" + PASSWORD);
                        conn = DriverManager.getConnection(path);
                        s = conn.createStatement();
                        return true;
                }
                catch (SQLException e) {
```

```
                                        return false;
                                }
                                catch (Exception e) {
                                        e.printStackTrace();
                                        return false;
                                }
                }

                public void CreateConnection(){
                                 if(conn == null)
                                 try{
                                 initializeConnection("localhost","bank","root","unyt");
                                 }catch(Exception e){e.printStackTrace();}
                  }

                public Connection getConnection() {
                                return conn;
                }

                public void closeConnection() {
        try {
            conn.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
                }

                public void closeStatement(){
                                try{
                                        s.close();
                                }catch(Exception e){
                                        e.printStackTrace();
                                }

                }

}
```

Database.java

```
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.Statement;
import java.util.ArrayList;

public class Database {

                 private static DbAccess dba = new DbAccess();
                 private static Connection conn;


                 public static void CreateConnection(){
                                 dba.CreateConnection();
```

```
                        conn = dba.getConnection();
                }

                public static void main(String args[])
                {
                        CreateConnection();
                        System.out.println(getCustomerId(2));
                }

        public static ArrayList<Integer> getCustomerId(int idAccount){
            ArrayList<Integer> ids = new  ArrayList<Integer>();
                                try {
                                        Statement s = dba.getConnection().createStatement();
                                        String sql = "Select idCustomer from accountCustomer where idAccount ='" +
idAccount + "'";
                                        ResultSet r = s.executeQuery(sql);
                                        while(r.next()){
                                                ids.add(r.getInt("IdCustomer"));
                                        }
                                } catch (Exception ex) {
                                        ex.printStackTrace();
                                }
                                return ids;
        }

            public static int getCustomerId2(int idAccount){
            ArrayList<Integer> ids = new  ArrayList<Integer>();
                                try {
                                        Statement s = dba.getConnection().createStatement();
                                        String sql = "Select idCustomer from accountCustomer where idAccount ='" +
idAccount + "'";
                                        ResultSet r = s.executeQuery(sql);
                                        while(r.next()){
                                                ids.add(r.getInt("idCustomer"));
                                        }
                                } catch (Exception ex) {
                                        ex.printStackTrace();
                                }
                                return 1;
        }

}
```

Account.java

```
import java.rmi.Remote;
import java.rmi.RemoteException;

public interface Account extends Remote {
  // Add method to return master BankManager
  public BankManager getBankManager()
      throws RemoteException;

  // Add method to return Client of this account
  public Client getClient()
```

```
      throws RemoteException;

  // Add method to return balance of this account
  public float getBalance()
      throws RemoteException;

    // Add method to return balance of this account
  public void setBalance(float bal)
      throws RemoteException;

  // Add method to withdraw cash from this account
  public long getCash (long amount)
      throws NoCashAvailableException, RemoteException;

  public void deposit(float amount)
    throws RemoteException;

    public void withdraw(float amount)
    throws RemoteException;
}
```

Client.java

```
import java.rmi.Remote;
import java.rmi.RemoteException;

public interface Client extends Remote {

  // Add method to return master BankManager
  public BankManager getBankManager()
      throws RemoteException;

  // Add method to return the name of this client
  public String getName()
      throws RemoteException;
}
```

BankManager.java

```
import java.rmi.Remote;
import java.rmi.RemoteException;
import java.util.ArrayList;

public interface BankManager extends Remote {

  // Add method to return an Account service
  public Account getAccount(String accountNumber)
      throws RemoteException;

  // Add method to return a Client service
  public Client getClient(String clientName)
      throws RemoteException;

  //public int getCustomerId(int accountId)
  //   throws RemoteException;
  public void deposit(String idAccount, float Amount)
        throws RemoteException;

  public void withdraw(String idAccount, float Amount)
        throws RemoteException;
}
```

AccountImpl.java

```
import java.io.Serializable;
import java.rmi.RemoteException;

public class AccountImpl implements Account, Serializable {

  private BankManager bankManager;
  private Client      client;
  private float       balance;
  private String      accountNumber;

  // public constructor
  public AccountImpl (
      BankManager bankManager,
      Client client,
      String accountNumber, float bal) {
    this.bankManager = bankManager;
    this.client      = client;
    this.balance     = bal;
    this.accountNumber = accountNumber;
  }

  @Override
  public void deposit(float amount) {
    balance += amount;
  }

  @Override
  public void withdraw(float amount) {
    balance -= amount;
  }

  @Override
  public BankManager getBankManager()
      throws RemoteException {
    return bankManager;
  }

  @Override
  public Client getClient()
      throws RemoteException {
    return client;
  }

  @Override
  public float getBalance()
      throws RemoteException {
```

```
    return balance;
  }

  @Override
    public void setBalance(float bal)
       throws RemoteException{
        balance = bal;
    }

  public long getCash(long amount)
      throws NoCashAvailableException, RemoteException {
    if (amount > balance) {
      throw new NoCashAvailableException();
    }
    balance = balance - amount;
    return amount;
  }
}
```

ClientImpl.java

```
import java.io.Serializable;
import java.rmi.RemoteException;

public class ClientImpl implements Client, Serializable {

  private BankManager bankManager;
  private String      clientName;

  // public constructor
  public ClientImpl(BankManager bm, String name) {
    this.bankManager = bm;
    this.clientName  = name;
  }

  @Override
  public BankManager getBankManager()
      throws RemoteException {
    return bankManager;
  }

  @Override
  public String getName()
      throws RemoteException {
    return clientName;
  }
}
```

BankManagerImpl.java

```
import java.io.Serializable;
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;

public class BankManagerImpl extends UnicastRemoteObject implements BankManager, Serializable {

  // public No-argument constructor
  public BankManagerImpl()
      throws java.rmi.RemoteException {
    super();
    testConn();
  }

  @Override
  public Account getAccount(String accountNumber)
      throws RemoteException {
    return null; // to be implemented
  }

  @Override
  public Client getClient(String clientName)
      throws RemoteException {
    return null; // to be implemented
  }

  @Override
  public int testConn(){
      Database.CreateConnection();
      return Database.getCustomerId(1).get(0);

  }
}
```

NoCashAvailableException.java

```
public class NoCashAvailableException extends Exception {
}
```

BankSystemServer.java

```
import java.awt.BorderLayout;
import java.net.MalformedURLException;
import java.awt.Dimension;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import java.rmi.Naming;
import java.rmi.NotBoundException;
import java.rmi.RemoteException;
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;

public class BankSystemServer extends JFrame{

                private JPanel contentPane;
                private JButton buttonConnect;
                private JButton buttonDisconnect;
                private JLabel labelState;
                private boolean connected;
                private BankManagerImpl BM;

                //private JButton buttonExit = null;

  // public No-argument constructor
  public BankSystemServer() {

    super();
    this.setSize(302, 149);
    this.setMinimumSize(new Dimension(265, 125));
    this.setPreferredSize(new Dimension(265, 125));
    this.setResizable(false);
    contentPane = new JPanel();
    this.setContentPane(contentPane);
    this.setTitle("Bank Manager Server");
    this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    labelState = new JLabel();
    labelState.setText("Pending to Start RMI Server...");
    buttonConnect = new JButton();
    buttonConnect.setText("Connect");
    buttonConnect.setPreferredSize(new Dimension(110, 25));
    buttonConnect.setMaximumSize(new Dimension(100, 25));
    buttonConnect.setMinimumSize(new Dimension(100, 25));
    buttonConnect.setSize(new Dimension(80, 25));
    buttonConnect.addMouseListener(new MouseAdapter() {
           public void mouseClicked(MouseEvent e) {
                  buttonConnect_mouseClicked(e);
           }
    });
    buttonConnect.setEnabled(true);
    buttonDisconnect = new JButton();
    buttonDisconnect.setText("Disconnect");
    buttonDisconnect.setPreferredSize(new Dimension(110, 25));
    buttonDisconnect.setMaximumSize(new Dimension(100, 25));
    buttonDisconnect.setMinimumSize(new Dimension(100, 25));
    buttonDisconnect.addMouseListener(new MouseAdapter() {
           public void mouseClicked(MouseEvent e) {
                  buttonDisconnect_mouseClicked(e);
           }
    });
                                buttonDisconnect.setEnabled(false);
                                contentPane.setLayout(new BorderLayout());
                                contentPane.setSize(new Dimension(260, 100));
                                contentPane.setPreferredSize(new Dimension(260, 100));
                                contentPane.setMinimumSize(new Dimension(260, 100));
                                contentPane.setMaximumSize(new Dimension(260, 100));
		JPanel panelbutton1 = new JPanel();
		panelbutton1.add(buttonConnect);

		JPanel panelbutton2 = new JPanel();
		panelbutton1.add(buttonDisconnect);

		contentPane.add(labelState,"North");
		contentPane.add(panelbutton1,"Center");
				//contentPane.add(panelbutton2,"Center");

		this.setVisible(true);

  }

		public void buttonDisconnect_mouseClicked(MouseEvent e) {
				if (connected) {
						disconnectServer();
				}
		}

		public void buttonConnect_mouseClicked(MouseEvent e) {
				if (!connected) {
						connectServer();
				}
		}

   private void connectServer() {
try {
        System.setProperty("java.security.policy","file:./security.policy");
        System.setProperty("java.rmi.server.hostname","192.168.14.202");
        Registry registry = LocateRegistry.createRegistry(1099);
        registry.rebind("BankManagerImpl", new BankManagerImpl());
       connected = true;
} catch (RemoteException err) {
       System.out.println(err.getMessage());
       err.printStackTrace();
}
if(connected) {
       buttonDisconnect.setEnabled(true);
       buttonConnect.setEnabled(false);
       labelState.setText("RMI Server started");
       System.out.println("Server Connected.");
} else {
       buttonDisconnect.setEnabled(false);
       buttonConnect.setEnabled(true);
       labelState.setText("Pending to start RMi Server...");
       System.out.println("Server Disconnected.");
}
		}

  private void disconnectServer() {
				BM = null;
				try {
						Naming.unbind("BankSystem");
						connected = false;
				} catch (RemoteException err) {
```

```
                        System.out.println(err.getMessage());
                        err.printStackTrace();
                } catch (NotBoundException err) {
                        connected = false;
                        System.out.println(err.getMessage());
                        err.printStackTrace();
                } catch (MalformedURLException err) {
                        System.out.println(err.getMessage());
                        err.printStackTrace();
                }
                if(connected) {
                        buttonDisconnect.setEnabled(true);
                        buttonConnect.setEnabled(false);
                        labelState.setText("RMI Server started");
                        System.out.println("Server Connected.");
                } else {
                        buttonDisconnect.setEnabled(false);
                        buttonConnect.setEnabled(true);
                        labelState.setText("Pending to start RMi Server...");
                        System.out.println("Server Disconnected.");
                }
        }


  public static void main(String args[]) {
    new BankSystemServer();

//    BankManager bm = null;
//
//    try {
//      // Create a BankManager object
//      bm = new BankManagerImpl();
//
//      // Export it to RMI
//      UnicastRemoteObject.exportObject( bm );
//    } catch (RemoteException remoteException) {
//      System.err.println(
//        "Failure during object export to RMI: " +
//         remoteException);
//    }
//
//    // Register an external name for the service
//    try {
//      Naming.rebind("//localhost/BankSystem", bm);
//    } catch (RemoteException remoteException) {
//      System.err.println(
//        "Failure during Name registration: " +
//        remoteException);
//    } catch (MalformedURLException malformedException) {
//      System.err.println(
//        "Failure during Name registration: " +
//        malformedException);
//    }
//
//    System.out.println("Server started.");
//    System.out.println("Enter <CR> to end.");
//    try {
//       int i = System.in.read();
//    } catch (IOException ioException) {
//    }
//    System.exit(0);
  }
}
```

BankUser.java

```
import java.rmi.*;
import javax.swing.*;
import java.awt.Container;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.awt.GridLayout;
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;

public class BankUser extends JFrame{

  // Interface reference to BankManager
   private BankManager bm;
   private JTextArea printArea;
   private JTextField customerField;
   private JTextField accountField;
   private JButton queryButton;
   private JTextField depositAccountField;
   private JTextField depositAmount;
   private JTextField withdrawAccountField;
   private JTextField withdrawAmount;

  // No-argument constructor
  public BankUser() {

    super();

    final int DEFAULT_FRAME_WIDTH = 430;
    final int DEFAULT_FRAME_HEIGHT = 500;
    setSize(DEFAULT_FRAME_WIDTH, DEFAULT_FRAME_HEIGHT);
    addWindowListener(new WindowCloser());
    setTitle("UNYT 2024 - MSc in Computer Science: Bank User Window");
      printArea = new JTextArea(10,10);

      customerField = new JTextField(10);
      accountField = new JTextField(10);

      queryButton = new JButton("Query Database");
      queryButton.addActionListener(new ButtonListener());

      // add components to content pane

      Container contentPane = getContentPane();
```

```
contentPane.setLayout(new GridLayout(4,1));
contentPane.add(printArea);

// arrange the label and text field in a panel

JPanel query1Panel = new JPanel();
query1Panel.add(new JLabel("Customer ID: "));
query1Panel.add(customerField);

JPanel query2Panel = new JPanel();
query2Panel.add(new JLabel("Account ID: "));
query2Panel.add(accountField);

// place the button in a panel

JPanel buttonPanel = new JPanel();
buttonPanel.add(queryButton);

// stack up these two panels in another panel

JPanel centerPanel = new JPanel();
centerPanel.setLayout(new GridLayout(3, 1));
centerPanel.add(query1Panel);
centerPanel.add(query2Panel);
centerPanel.add(buttonPanel);

contentPane.add(centerPanel);


//// Deposit part

depositAccountField = new JTextField(10);
depositAmount = new JTextField(10);

JPanel query3Panel = new JPanel();
query3Panel.add(new JLabel("Account ID to deposit: "));
query3Panel.add(depositAccountField);

JPanel query4Panel = new JPanel();
query4Panel.add(new JLabel("Amount to deposit: "));
query4Panel.add(depositAmount);

JPanel depositPanel = new JPanel();
depositPanel.setLayout(new GridLayout(3, 1));
depositPanel.add(query3Panel);
depositPanel.add(query4Panel);

JButton buttonPayment = new JButton("Deposit");
buttonPayment.addActionListener(new ButtonPaymentListener());
JPanel query5Panel = new JPanel();
query5Panel.add(buttonPayment);

depositPanel.add(query5Panel);
contentPane.add(depositPanel);

//// Withdraw part
```

```
  withdrawAccountField = new JTextField(10);
  withdrawAmount = new JTextField(10);

  JPanel query6Panel = new JPanel();
  query6Panel.add(new JLabel("Account ID to withdraw: "));
  query6Panel.add(withdrawAccountField);

  JPanel query7Panel = new JPanel();
  query7Panel.add(new JLabel("Amount to withdraw: "));
  query7Panel.add(withdrawAmount);

  JPanel withdrawPanel = new JPanel();
  withdrawPanel.setLayout(new GridLayout(3, 1));
  withdrawPanel.add(query6Panel);
  withdrawPanel.add(query7Panel);

  JButton buttonWithdraw = new JButton("Withdraw");
  buttonWithdraw.addActionListener(new ButtonWithdrawListener());
  JPanel query8Panel = new JPanel();
  query8Panel.add(buttonWithdraw);

  withdrawPanel.add(query8Panel);
  contentPane.add(withdrawPanel);



  setVisible(true);

try {
Registry registry = LocateRegistry.getRegistry("localhost", 1099);
bm = (BankManager) registry.lookup("BankManagerImpl");
}catch (NotBoundException notBoundException) {
  System.err.println("Not Bound: " + notBoundException);
} catch (RemoteException remoteException) {
  System.err.println("Remote Exception: " + remoteException);
}

try {
  // Lookup account 4461
  Account account = bm.getAccount("1");



  // Get client for account
  Client client = account.getClient();
  System.out.println("Currently in the database there is this customer with ID 1:" + client.getName());

  // Get name for client
  String name = client.getName();

  // Get balance for account
  //long cash = account.getBalance();

  // Format and display output
 // NumberFormat currencyFormat =
     //   NumberFormat.getCurrencyInstance(Locale.US);
     // String balanceString = currencyFormat.format(cash);
      //System.out.println(name + "'s account has " + balanceString);

    } catch (RemoteException remoteException) {
      System.err.println(remoteException);
    }
  }



     private class WindowCloser extends WindowAdapter
   {  public void windowClosing(WindowEvent event)
      {  System.exit(0);
      }
   }

     private class ButtonPaymentListener implements ActionListener
   {  public void actionPerformed(ActionEvent event)
      {  // get user input from text area
      try{
         if(!depositAccountField.getText().equals("") && !depositAmount.getText().equals("")){
             bm.deposit(depositAccountField.getText(), Float.parseFloat(depositAmount.getText()));
             Account account = bm.getAccount(depositAccountField.getText());
             printArea.setText("The account was credited \n as follows:" + account.getClient().getName() + "\n Balance: " +
account.getBalance());
             }
             }catch (RemoteException remoteException) {
      System.err.println(remoteException);
    }
      }
     }

        private class ButtonWithdrawListener implements ActionListener
   {  public void actionPerformed(ActionEvent event)
      {  // get user input from text area
      try{
         if(!withdrawAccountField.getText().equals("") && !withdrawAmount.getText().equals("")){
             bm.withdraw(withdrawAccountField.getText(), Float.parseFloat(withdrawAmount.getText()));
             Account account = bm.getAccount(withdrawAccountField.getText());
             printArea.setText("The account was credited \n as follows:" + account.getClient().getName() + "\n Balance: " +
account.getBalance());
             }
             }catch (RemoteException remoteException) {
      System.err.println(remoteException);
    }
      }
     }

    private class ButtonListener implements ActionListener
   {  public void actionPerformed(ActionEvent event)
      {  // get user input from text area
      try{
         if(!customerField.getText().equals("")){
        System.out.println("opps" + customerField.getText() + "000");
        Client client = bm.getClient(customerField.getText());
        customerField.setText("");
        System.out.println("Currently in the database "
                + "there \n is this customer with the "
                + "requested ID:" + client.getName());
        printArea.setText("Currently in the database "
                + "there \n is this customer with "
                + "the requested ID:" + client.getName());
 }else
          if(!accountField.getText().equals("")){
                Account account = bm.getAccount(accountField.getText());
                accountField.setText("");
                System.out.println("Currently in the database there is \n this account with the requested ID:" +
account.getClient().getName());
                printArea.setText("Currently in the database there is \n this account with the requested ID:" +
account.getClient().getName() + "\n Balance: " + account.getBalance());

          }
          }catch (RemoteException remoteException) {
      System.err.println(remoteException);
    }
      }
     }

  public static void main(String[] args) {
    new BankUser();
  }
}
```

Step 2: Compile the code following these steps:

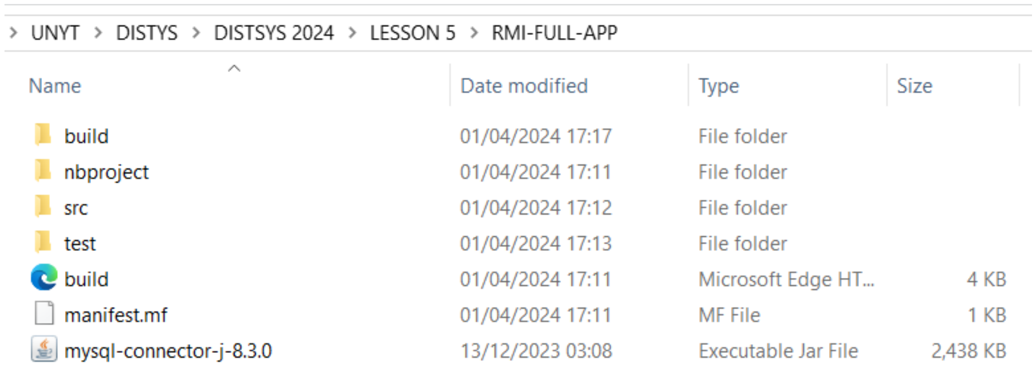
In order to connect Java with MySQL we need the following connector:

mysql-connector-j-8.3.0.jar

Download such connector from the following site:

https://dev.mysql.com/downloads/connector/j/

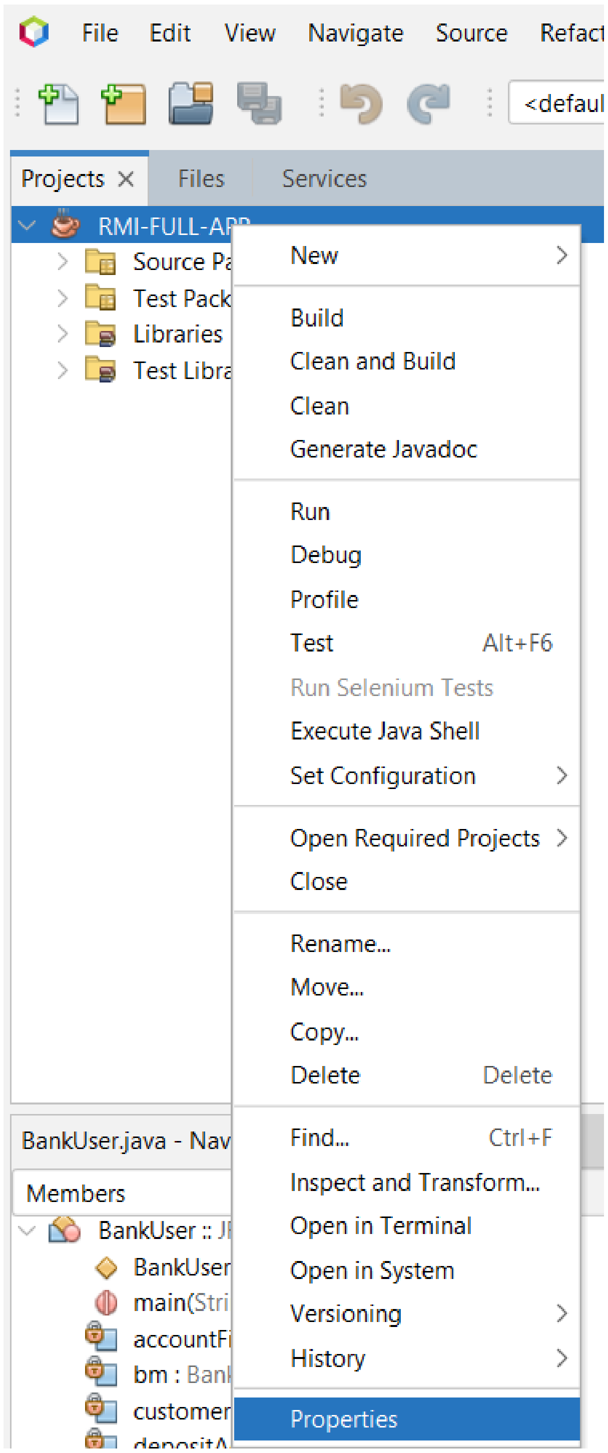
as platform independent file which is a zip file inside which you will find the file mysql-connector-j-8.3.0.jar. Copy such file in the main folder of the NetBeans project folder as follows:



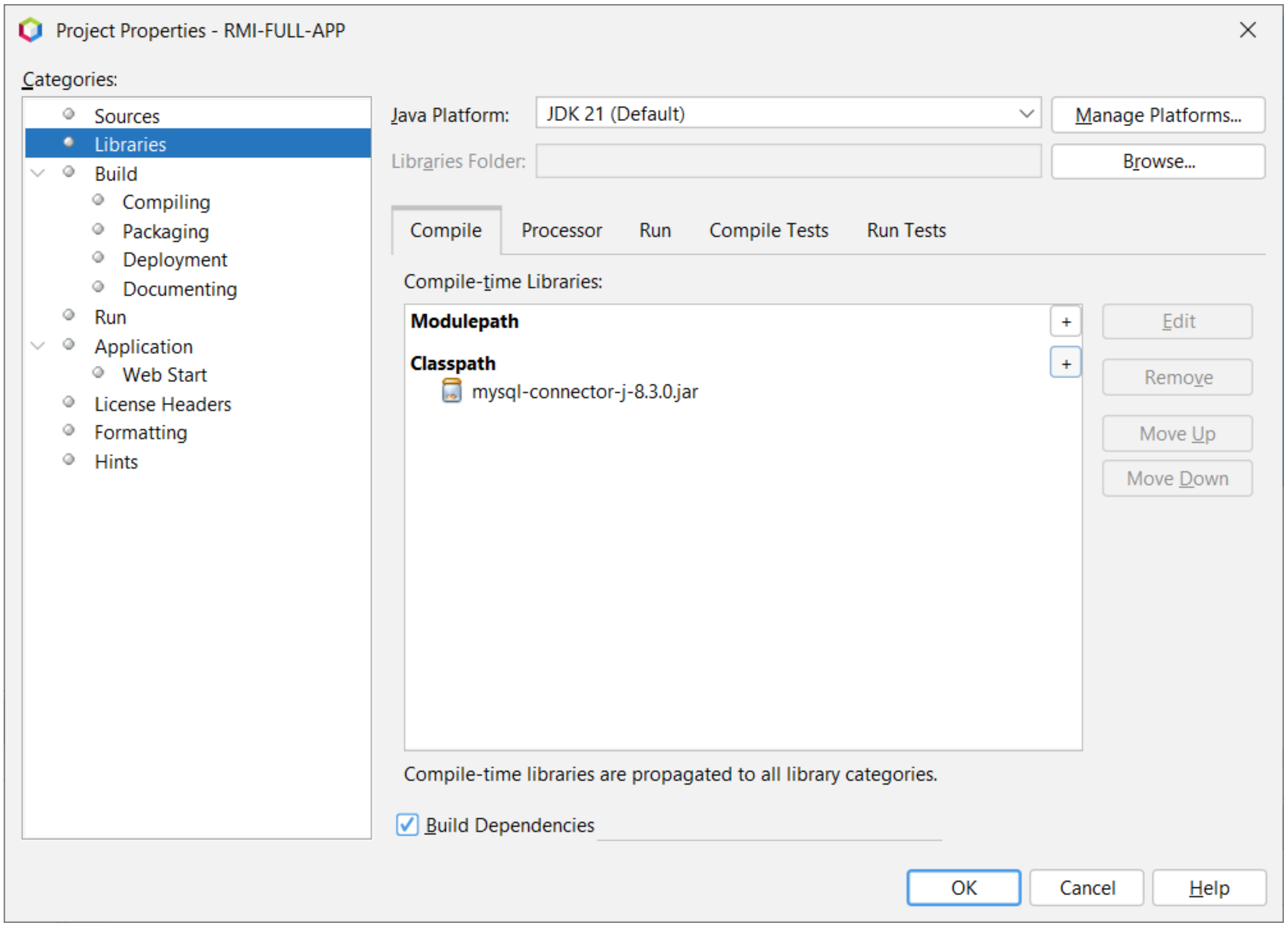
> UNYT > DISTYS > DISTSYS 2024 > LESSON 5 > RMI-FULL-APP

Name	Date modified	Type	Size
build	01/04/2024 17:17	File folder	
nbproject	01/04/2024 17:11	File folder	
src	01/04/2024 17:12	File folder	
test	01/04/2024 17:13	File folder	
build	01/04/2024 17:11	Microsoft Edge HT...	4 KB
manifest.mf	01/04/2024 17:11	MF File	1 KB
mysql-connector-j-8.3.0	13/12/2023 03:08	Executable Jar File	2,438 KB

Add the connector to the libraries of the project in Netbeans as follows:



Click with right of the mouse on the Project. Select Properties and in the next window select Libraries:



Project Properties - RMI-FULL-APP

Categories:

- Sources
- Libraries
- Build
  - Compiling
  - Packaging
  - Deployment
  - Documenting
- Run
- Application
  - Web Start
- License Headers
- Formatting
- Hints

Java Platform: JDK 21 (Default)

Libraries Folder:

Compile-time Libraries:

Modulepath

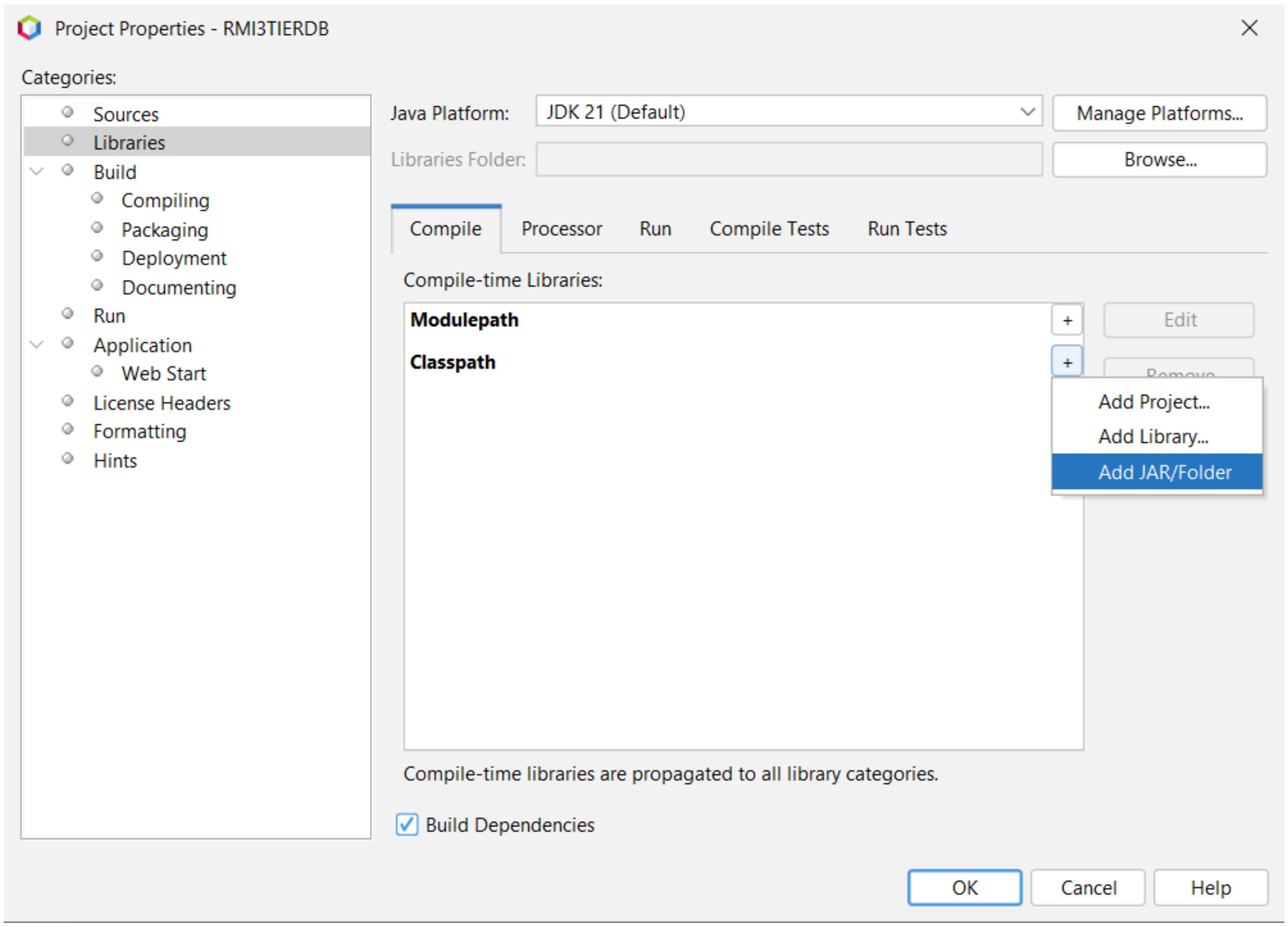
Classpath

mysql-connector-j-8.3.0.jar

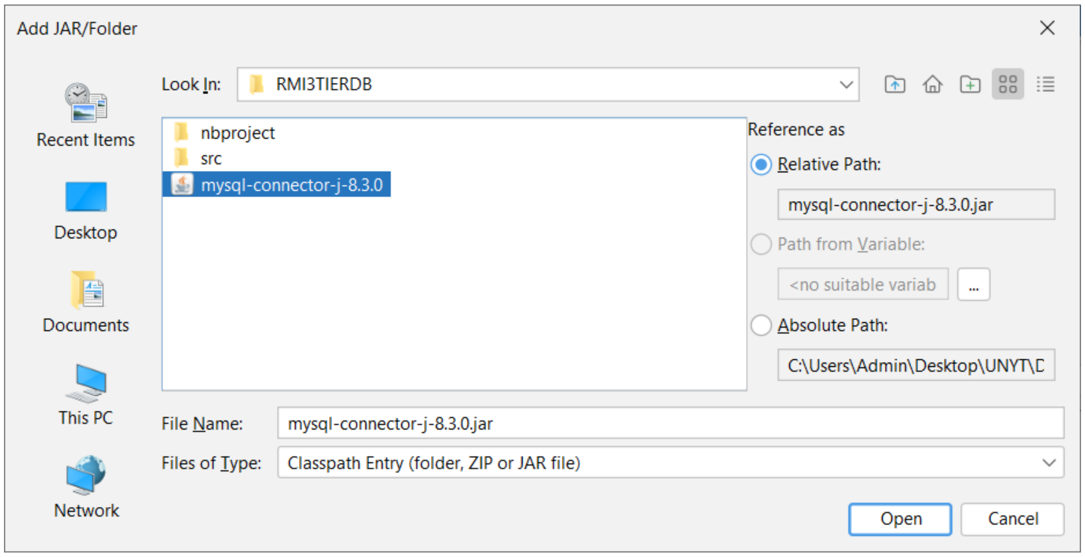
Compile-time libraries are propagated to all library categories.

☑ Build Dependencies

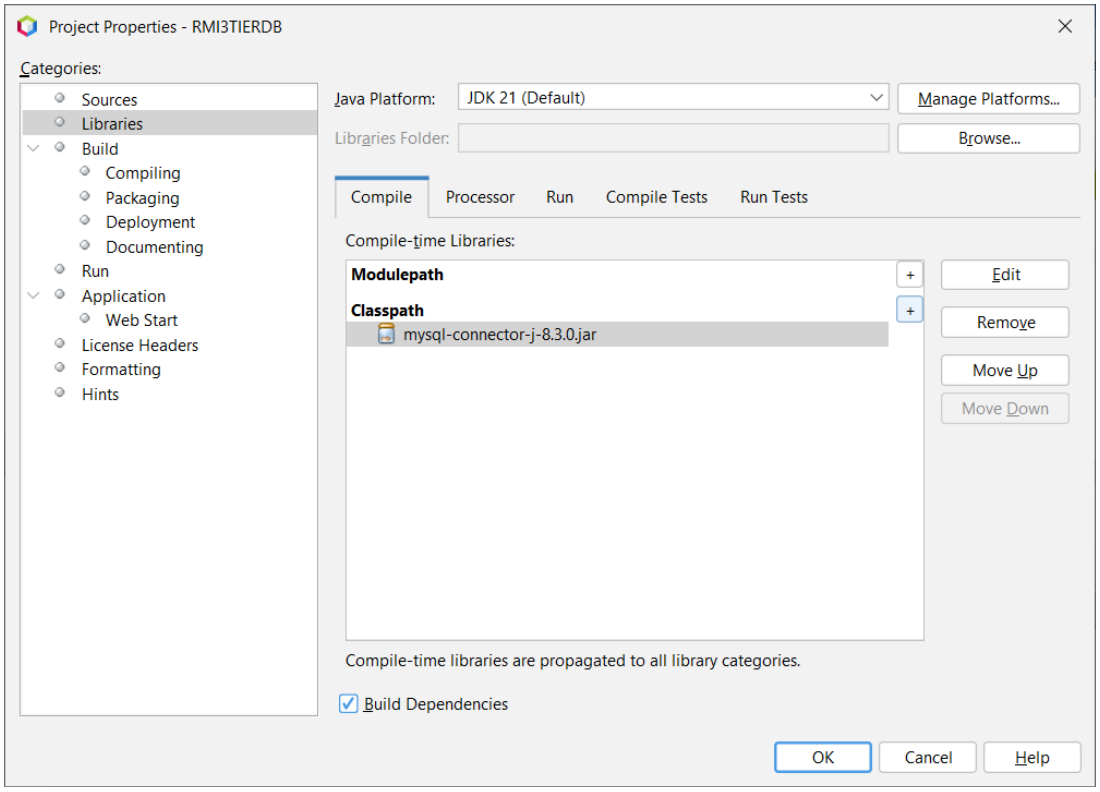
Click on the right + symbol of the Classpath item as follows, and select Add Jar/Folder:



Select relative path:


Add JAR/Folder
Look In: RMI3TIERDB
Recent Items
Desktop
Documents
This PC
Network
nbproject
src
mysql-connector-j-8.3.0
Reference as
Relative Path: mysql-connector-j-8.3.0.jar
Path from Variable: <no suitable variab
Absolute Path: C:\Users\Admin\Desktop\UNYT\C
File Name: mysql-connector-j-8.3.0.jar
Files of Type: Classpath Entry (folder, ZIP or JAR file)
Open
Cancel

You should see the following window:



Java Platform: JDK 21 (Default)

Libraries Folder:

Compile-time Libraries:

Modulepath

Classpath

mysql-connector-j-8.3.0.jar

Compile-time libraries are propagated to all library categories.

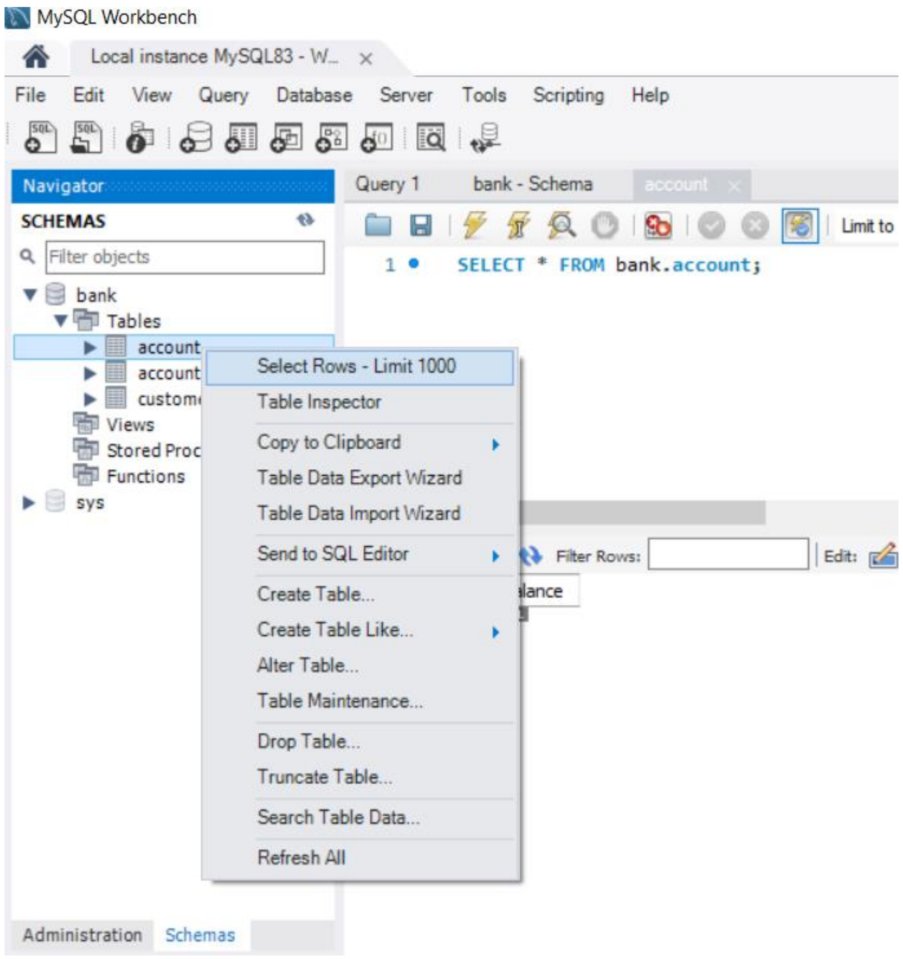
[✓] Build Dependencies

Test the connection with MySQL as follows by using the following classes:

DbAccess.java and Database.java

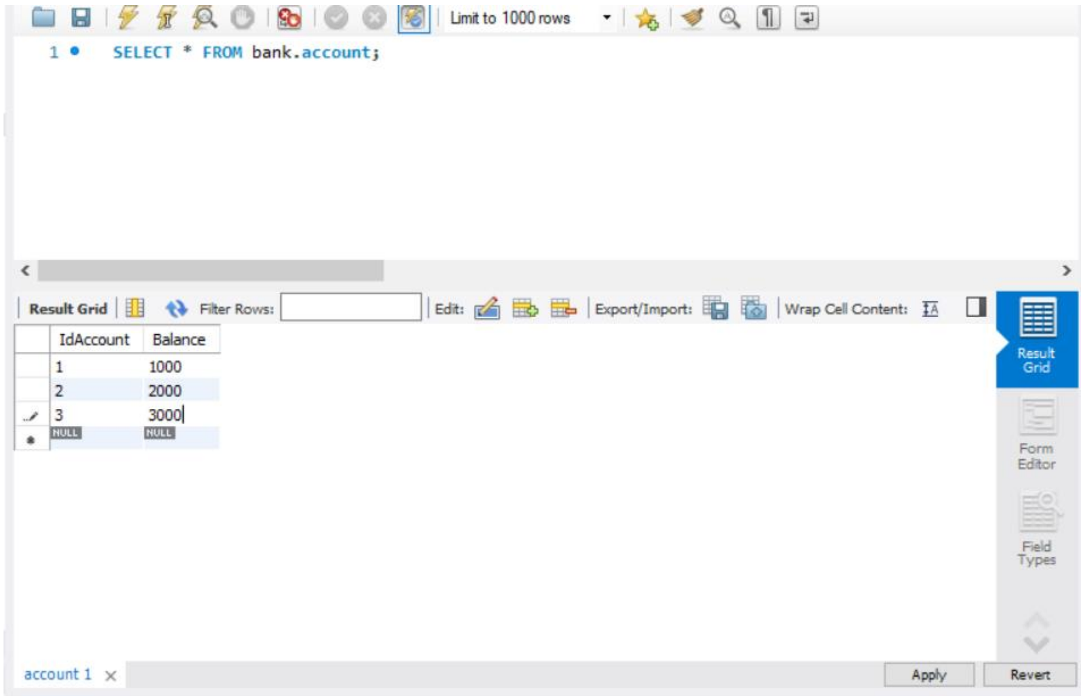
To test the connection insert some data in the database as follows:

Click with the right button of the mous on the table account and then click on Select Rows as follows:



```sql
SELECT * FROM bank.account;
```

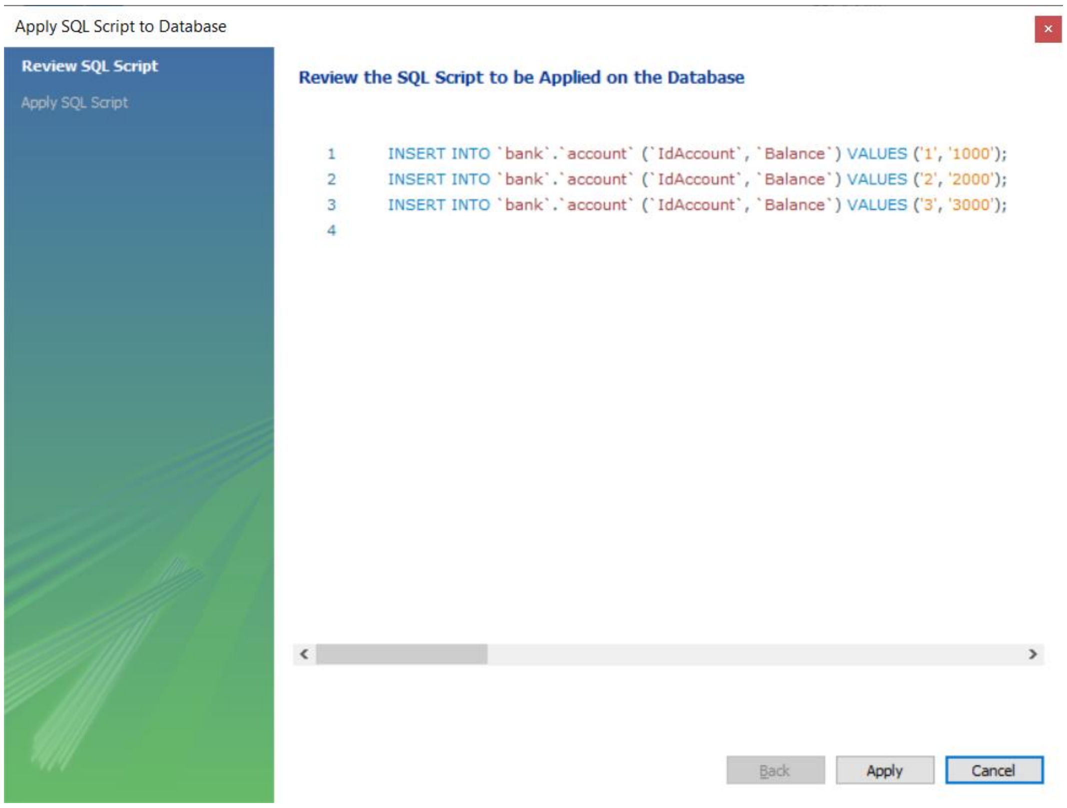
Double click on the fields of the table directly and fill in the data as follows:



```
SELECT * FROM bank.account;
```

IdAccount	Balance
1	1000
2	2000
3	3000
NULL	NULL

In the above window and the following one, click on Apply and the data will be saved into the table:

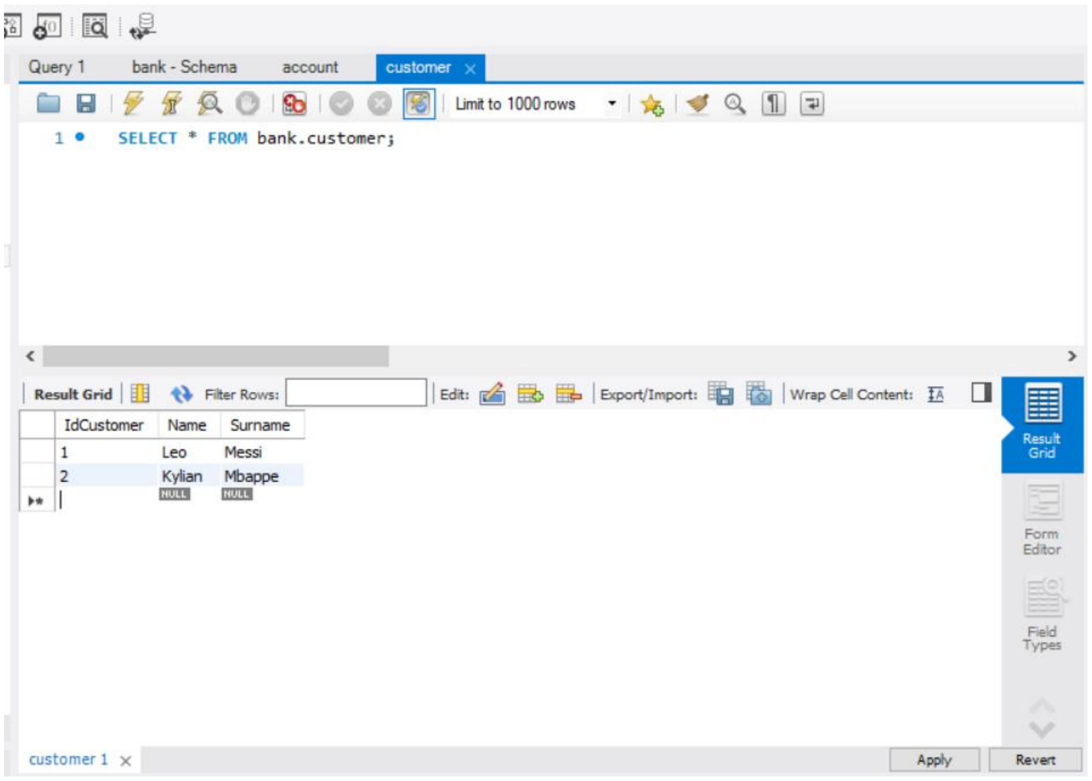


Apply SQL Script to Database
Review SQL Script
Apply SQL Script
Review the SQL Script to be Applied on the Database
Back
Apply
Cancel

```
INSERT INTO `bank`.`account` (`IdAccount`, `Balance`) VALUES ('1', '1000');
INSERT INTO `bank`.`account` (`IdAccount`, `Balance`) VALUES ('2', '2000');
INSERT INTO `bank`.`account` (`IdAccount`, `Balance`) VALUES ('3', '3000');
```

Repeat the same operations for the other two tables as follows:

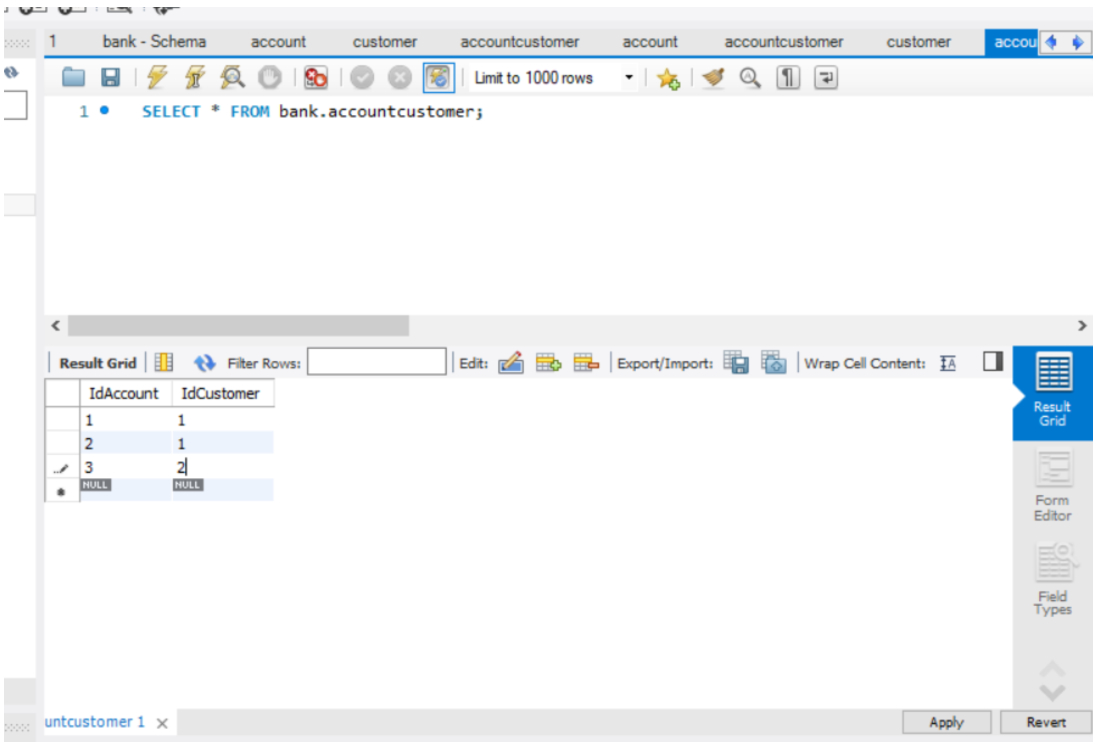
Table customer



```
SELECT * FROM bank.customer;
```

IdCustomer	Name	Surname
1	Leo	Messi
2	Kylian	Mbappe
	NULL	NULL

Table accountcustomer:

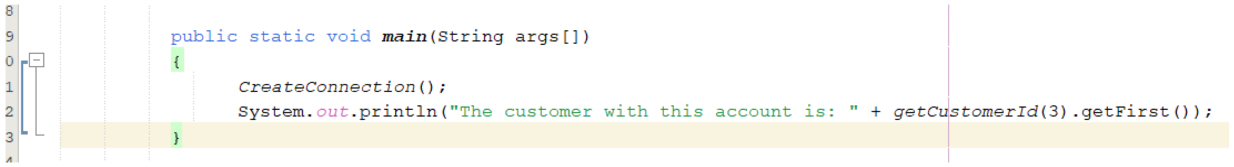


```
SELECT * FROM bank.accountcustomer;
```

IdAccount	IdCustomer
1	1
2	1
3	2
NULL	NULL

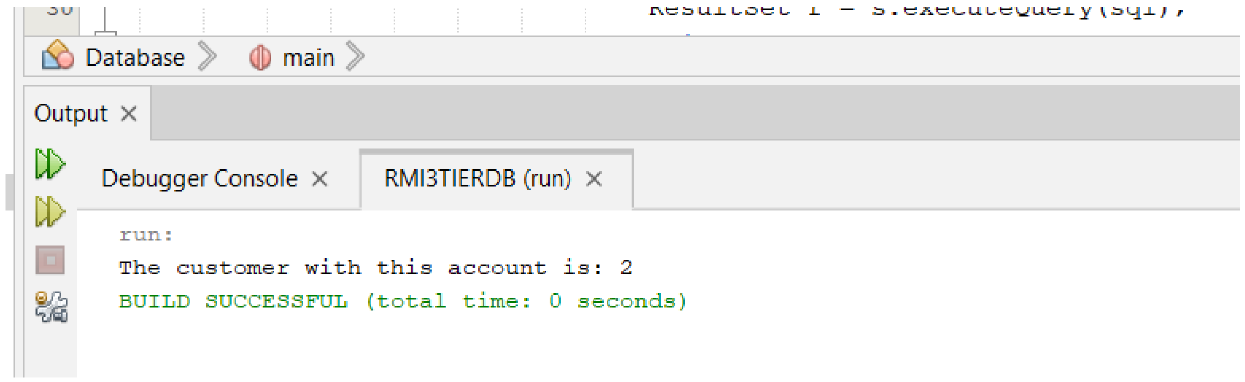
Now test the connection of the Java programs with the query to find out the customer to which account 3 belongs to.

In the Database class, you will find these statements:



```
public static void main(String args[])
{
    CreateConnection();
    System.out.println("The customer with this account is: " + getCustomerId(3).getFirst());
}
```

If you run the class, you will see the following result as per the data in the database:



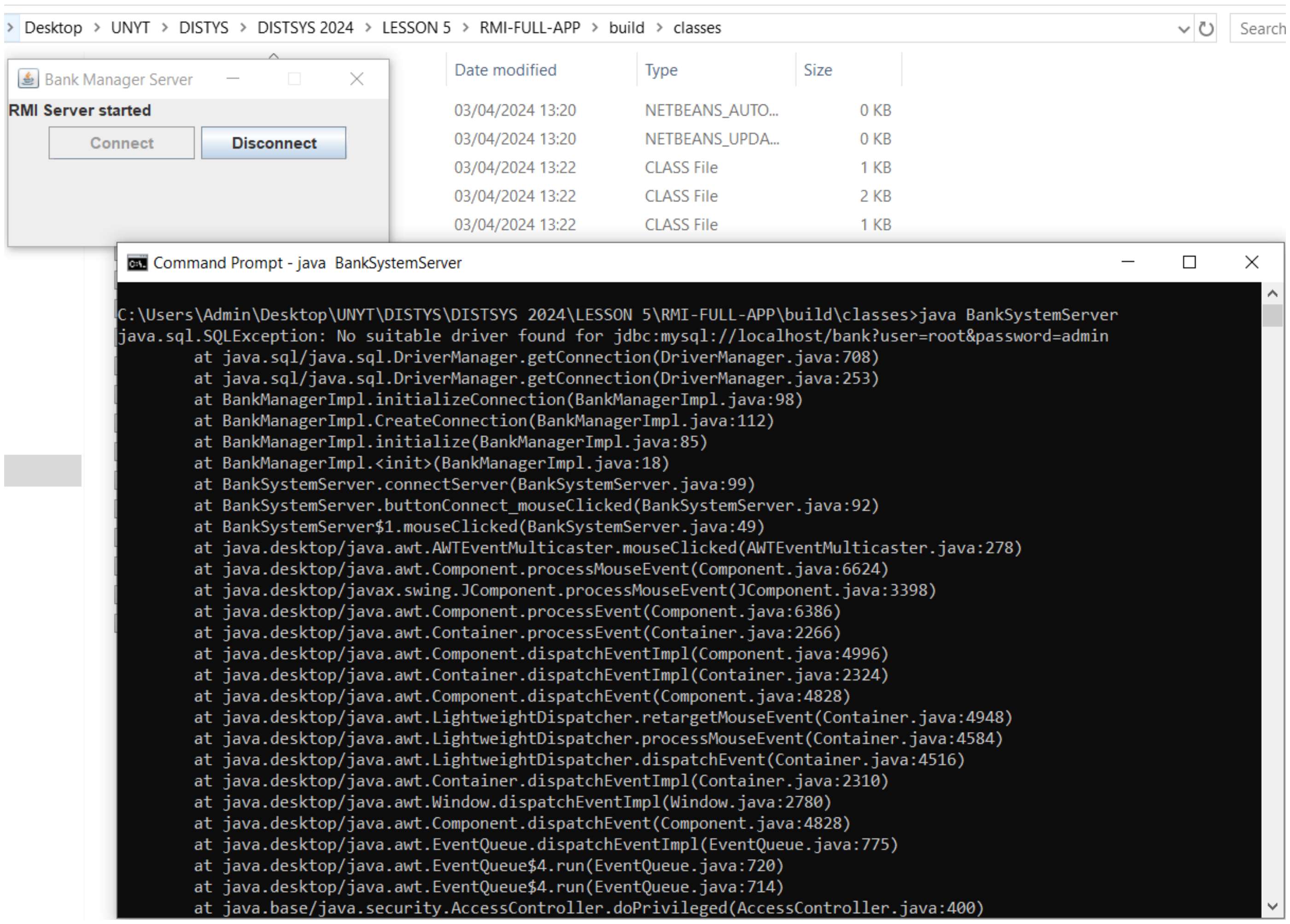
```
run:
The customer with this account is: 2
BUILD SUCCESSFUL (total time: 0 seconds)
```

3. Run the server and the client

3.1 Run the server RMI Registry program.

Open a command line and move to the directory that will contain your code from this exercise.

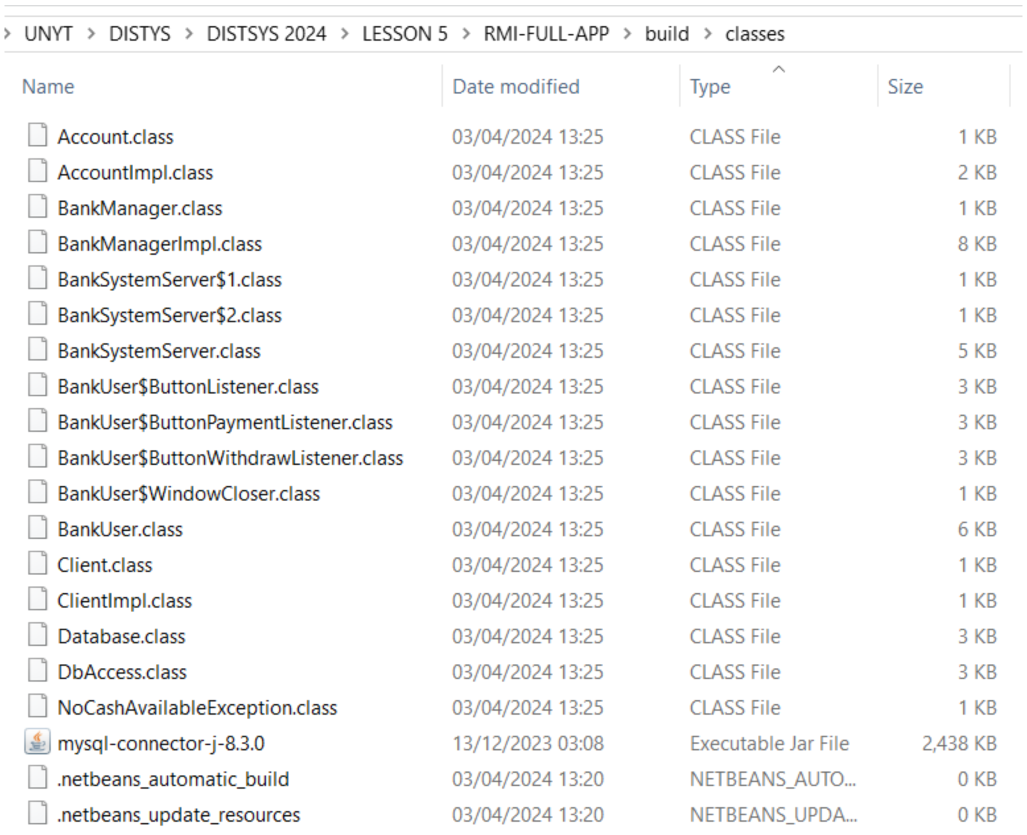
From that location, run the server with the following command. Click on the button Connect. You will get an error due to the absence of the JDBC connector missing as follows:


Desktop > UNYT > DISTYS > DISTSYS 2024 > LESSON 5 > RMI-FULL-APP > build > classes
Search
Bank Manager Server
RMI Server started
Connect
Disconnect
Command Prompt - java BankSystemServer

Date modified	Type	Size
03/04/2024 13:20	NETBEANS_AUTO...	0 KB
03/04/2024 13:20	NETBEANS_UPDA...	0 KB
03/04/2024 13:22	CLASS File	1 KB
03/04/2024 13:22	CLASS File	2 KB
03/04/2024 13:22	CLASS File	1 KB

```
C:\Users\Admin\Desktop\UNYT\DISTYS\DISTSYS 2024\LESSON 5\RMI-FULL-APP\build\classes>java BankSystemServer
java.sql.SQLException: No suitable driver found for jdbc:mysql://localhost/bank?user=root&password=admin
        at java.sql/java.sql.DriverManager.getConnection(DriverManager.java:708)
        at java.sql/java.sql.DriverManager.getConnection(DriverManager.java:253)
        at BankManagerImpl.initializeConnection(BankManagerImpl.java:98)
        at BankManagerImpl.CreateConnection(BankManagerImpl.java:112)
        at BankManagerImpl.initialize(BankManagerImpl.java:85)
        at BankManagerImpl.<init>(BankManagerImpl.java:18)
        at BankSystemServer.connectServer(BankSystemServer.java:99)
        at BankSystemServer.buttonConnect_mouseClicked(BankSystemServer.java:92)
        at BankSystemServer$1.mouseClicked(BankSystemServer.java:49)
        at java.desktop/java.awt.AWTEventMulticaster.mouseClicked(AWTEventMulticaster.java:278)
        at java.desktop/java.awt.Component.processMouseEvent(Component.java:6624)
        at java.desktop/javax.swing.JComponent.processMouseEvent(JComponent.java:3398)
        at java.desktop/java.awt.Component.processEvent(Component.java:6386)
        at java.desktop/java.awt.Container.processEvent(Container.java:2266)
        at java.desktop/java.awt.Component.dispatchEventImpl(Component.java:4996)
        at java.desktop/java.awt.Container.dispatchEventImpl(Container.java:2324)
        at java.desktop/java.awt.Component.dispatchEvent(Component.java:4828)
        at java.desktop/java.awt.LightweightDispatcher.retargetMouseEvent(Container.java:4948)
        at java.desktop/java.awt.LightweightDispatcher.processMouseEvent(Container.java:4584)
        at java.desktop/java.awt.LightweightDispatcher.dispatchEvent(Container.java:4516)
        at java.desktop/java.awt.Container.dispatchEventImpl(Container.java:2310)
        at java.desktop/java.awt.Window.dispatchEventImpl(Window.java:2780)
        at java.desktop/java.awt.Component.dispatchEvent(Component.java:4828)
        at java.desktop/java.awt.EventQueue.dispatchEventImpl(EventQueue.java:775)
        at java.desktop/java.awt.EventQueue$4.run(EventQueue.java:720)
        at java.desktop/java.awt.EventQueue$4.run(EventQueue.java:714)
        at java.base/java.security.AccessController.doPrivileged(AccessController.java:400)
```

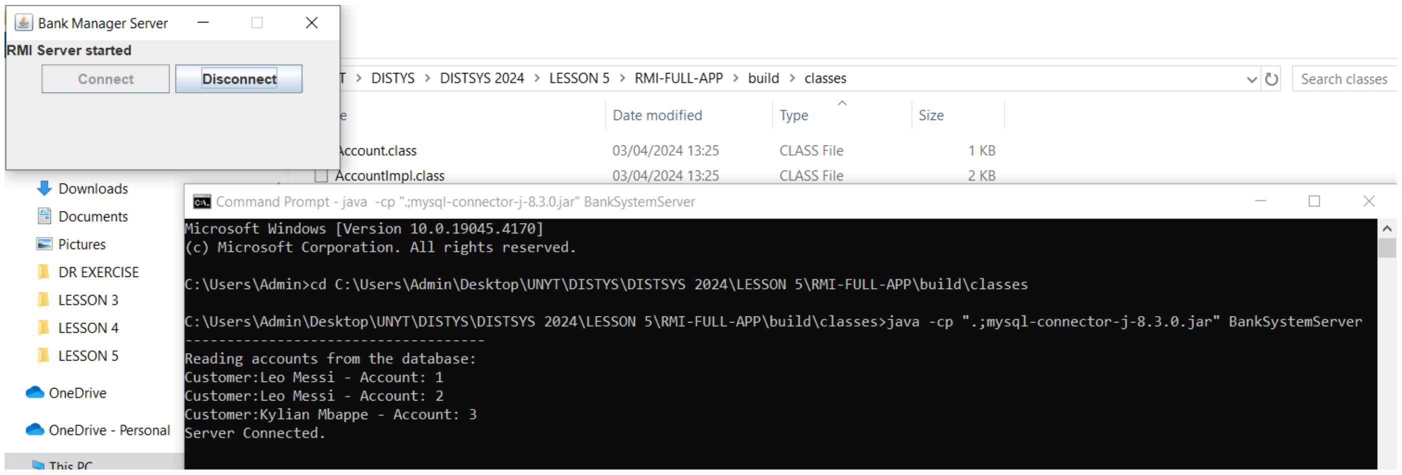
Copy the JDBC connector under the directory of the java classes for easiness of invocation in the command line as follows:



› UNYT › DISTYS › DISTSYS 2024 › LESSON 5 › RMI-FULL-APP › build › classes

Name	Date modified	Type	Size
Account.class	03/04/2024 13:25	CLASS File	1 KB
AccountImpl.class	03/04/2024 13:25	CLASS File	2 KB
BankManager.class	03/04/2024 13:25	CLASS File	1 KB
BankManagerImpl.class	03/04/2024 13:25	CLASS File	8 KB
BankSystemServer$1.class	03/04/2024 13:25	CLASS File	1 KB
BankSystemServer$2.class	03/04/2024 13:25	CLASS File	1 KB
BankSystemServer.class	03/04/2024 13:25	CLASS File	5 KB
BankUser$ButtonListener.class	03/04/2024 13:25	CLASS File	3 KB
BankUser$ButtonPaymentListener.class	03/04/2024 13:25	CLASS File	3 KB
BankUser$ButtonWithdrawListener.class	03/04/2024 13:25	CLASS File	3 KB
BankUser$WindowCloser.class	03/04/2024 13:25	CLASS File	1 KB
BankUser.class	03/04/2024 13:25	CLASS File	6 KB
Client.class	03/04/2024 13:25	CLASS File	1 KB
ClientImpl.class	03/04/2024 13:25	CLASS File	1 KB
Database.class	03/04/2024 13:25	CLASS File	3 KB
DbAccess.class	03/04/2024 13:25	CLASS File	3 KB
NoCashAvailableException.class	03/04/2024 13:25	CLASS File	1 KB
mysql-connector-j-8.3.0	13/12/2023 03:08	Executable Jar File	2,438 KB
.netbeans_automatic_build	03/04/2024 13:20	NETBEANS_AUTO...	0 KB
.netbeans_update_resources	03/04/2024 13:20	NETBEANS_UPDA...	0 KB

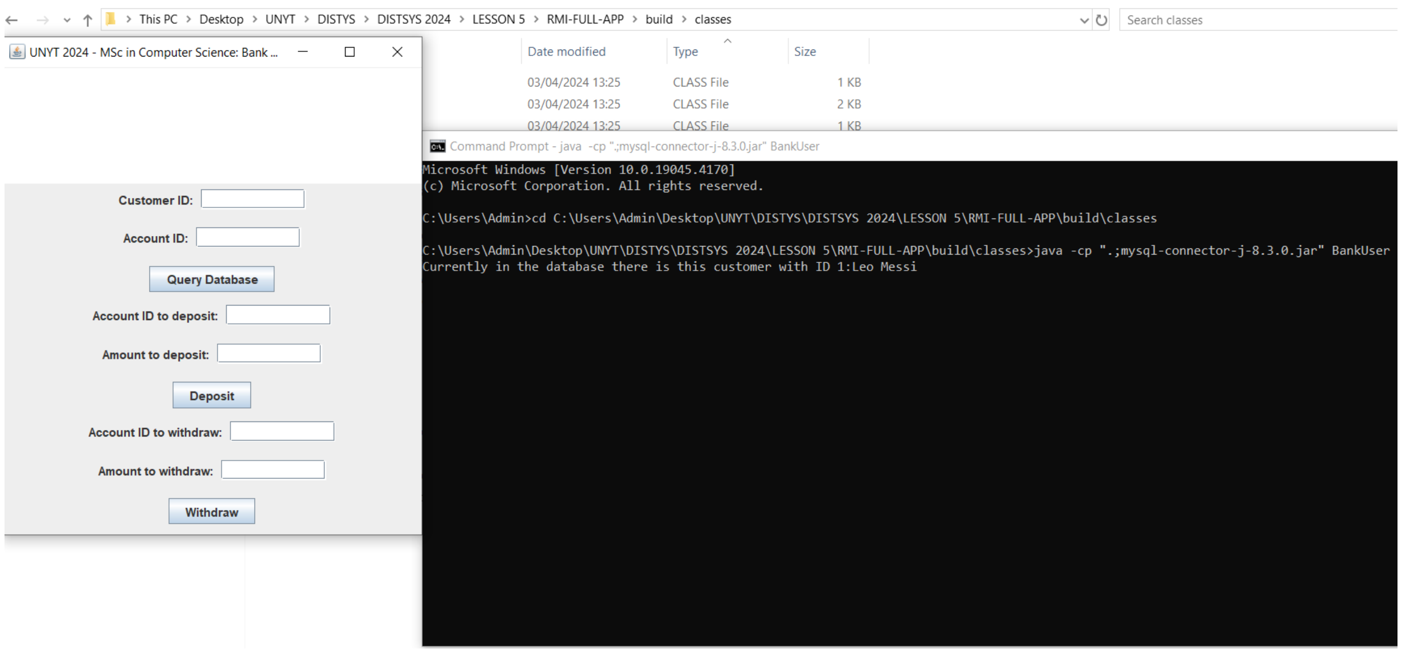
Now you can launch again the program as follows using the classpath option of Java as follows:



```
Command Prompt - java -cp ".;mysql-connector-j-8.3.0.jar" BankSystemServer
Microsoft Windows [Version 10.0.19045.4170]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Admin>cd C:\Users\Admin\Desktop\UNYT\DISTYS\DISTSYS 2024\LESSON 5\RMI-FULL-APP\build\classes

C:\Users\Admin\Desktop\UNYT\DISTYS\DISTSYS 2024\LESSON 5\RMI-FULL-APP\build\classes>java -cp ".;mysql-connector-j-8.3.0.jar" BankSystemServer
------------------------------------
Reading accounts from the database:
Customer:Leo Messi - Account: 1
Customer:Leo Messi - Account: 2
Customer:Kylian Mbappe - Account: 3
Server Connected.
```

Do not close this window!

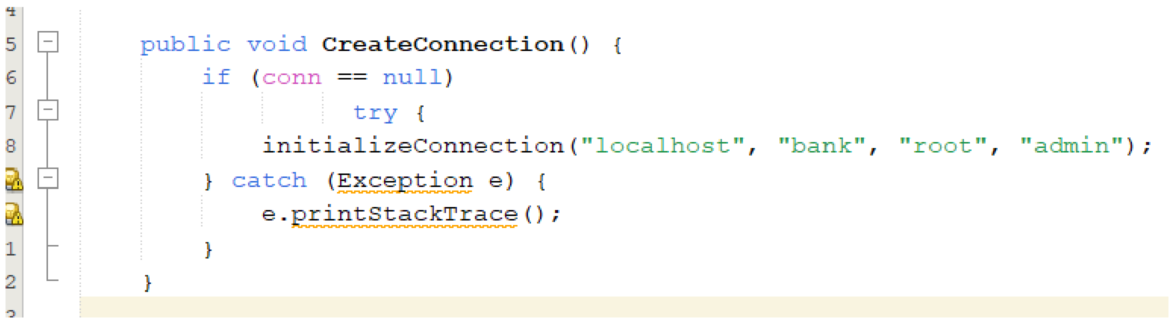
Open another cmd window and invoke now the client as follows:


This PC > Desktop > UNYT > DISTYS > DISTSYS 2024 > LESSON 5 > RMI-FULL-APP > build > classes
Search classes
Date modified
Type
Size
03/04/2024 13:25 CLASS File 1 KB
03/04/2024 13:25 CLASS File 2 KB
03/04/2024 13:25 CLASS File 1 KB
UNYT 2024 - MSc in Computer Science: Bank ...
Customer ID:
Account ID:
Query Database
Account ID to deposit:
Amount to deposit:
Deposit
Account ID to withdraw:
Amount to withdraw:
Withdraw
Command Prompt - java -cp ".;mysql-connector-j-8.3.0.jar" BankUser

```
Microsoft Windows [Version 10.0.19045.4170]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Admin>cd C:\Users\Admin\Desktop\UNYT\DISTYS\DISTSYS 2024\LESSON 5\RMI-FULL-APP\build\classes

C:\Users\Admin\Desktop\UNYT\DISTYS\DISTSYS 2024\LESSON 5\RMI-FULL-APP\build\classes>java -cp ".;mysql-connector-j-8.3.0.jar" BankUser
Currently in the database there is this customer with ID 1:Leo Messi
```

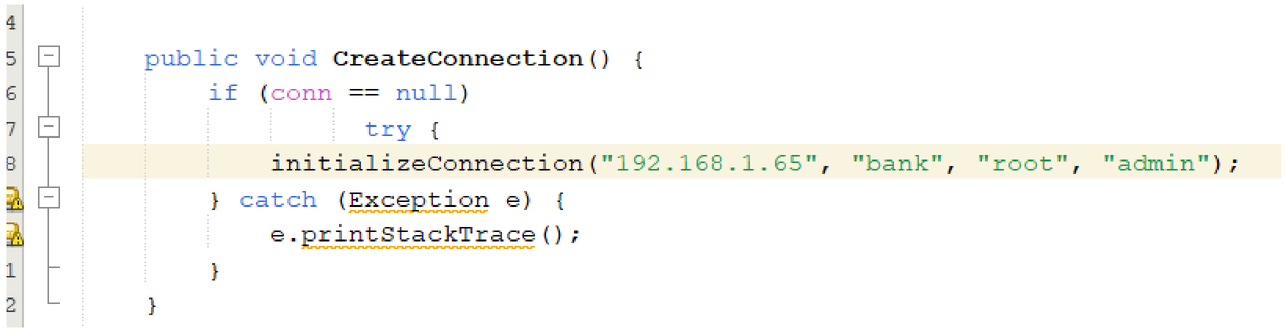
4. Run the database server, the RMI server and the client in three different computers.

First open the database class DbAccess and change the IP of the database server as follows:



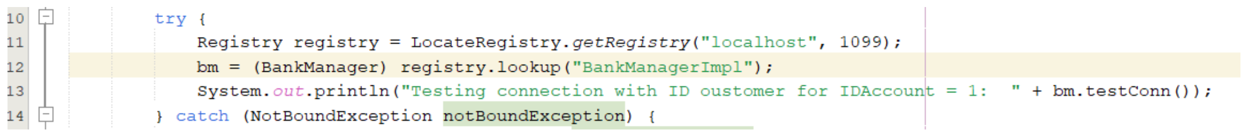
```java
    public void CreateConnection() {
        if (conn == null)
                try {
            initializeConnection("localhost", "bank", "root", "admin");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
```

Instead of localhost you should put the IP of the database server for example 192.168.1.65 if you are working in LAN:



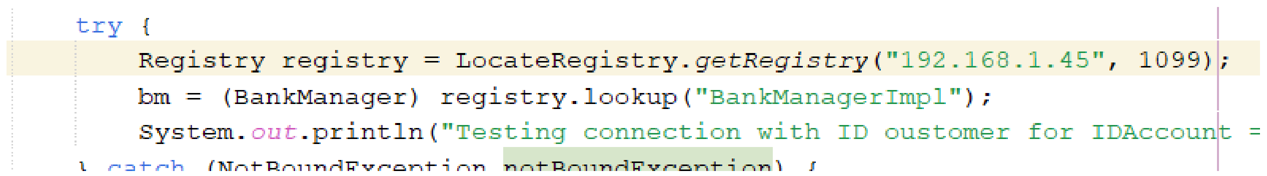
```
    public void CreateConnection() {
        if (conn == null)
            try {
            initializeConnection("192.168.1.65", "bank", "root", "admin");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
```

Then you need to run BankUser from a different IP of where the BankSystemServer is running. In BankUser you should replace localhost with the IP of the machine where you are running BankSystemServer.



```
        try {
            Registry registry = LocateRegistry.getRegistry("localhost", 1099);
            bm = (BankManager) registry.lookup("BankManagerImpl");
            System.out.println("Testing connection with ID oustomer for IDAccount = 1:  " + bm.testConn());
        } catch (NotBoundException notBoundException) {
```

Change the IP as follows if for example the BankSystemServer is running on 192.168.1.45.



```
    try {
        Registry registry = LocateRegistry.getRegistry("192.168.1.45", 1099);
        bm = (BankManager) registry.lookup("BankManagerImpl");
        System.out.println("Testing connection with ID oustomer for IDAccount =
    } catch (NotBoundException notBoundException) {
```

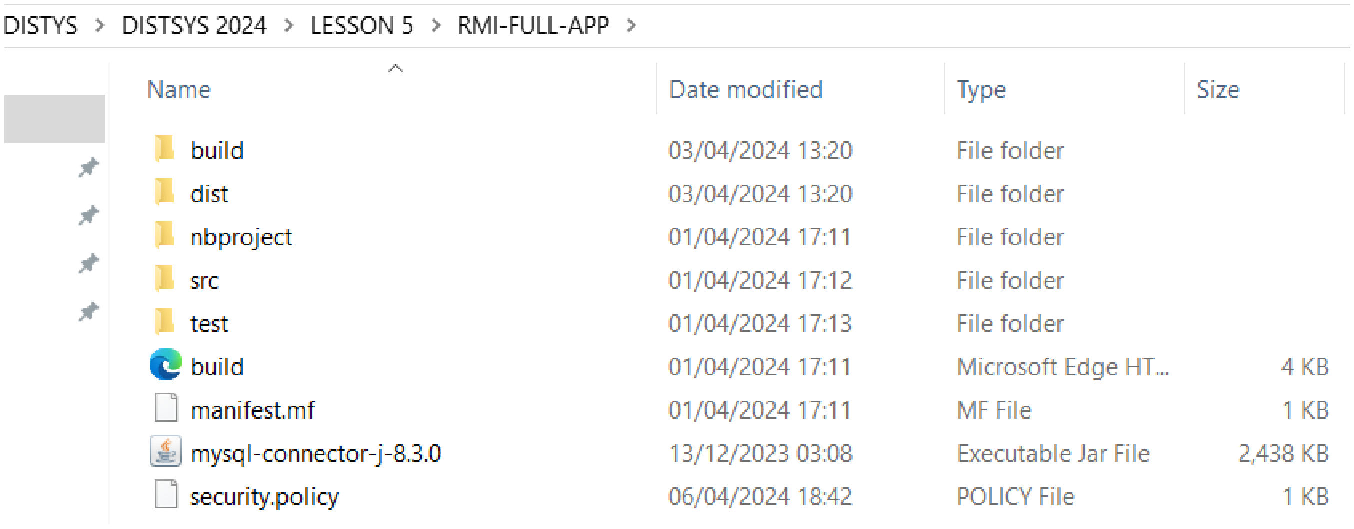
In addition create the following file in the rood directory of the project:

Filename: security.policy

Content of the file as follows:

```
grant {
    // Allow everything for now
    permission java.security.AllPermission;
};
```

As follows:


DISTYS › DISTSYS 2024 › LESSON 5 › RMI-FULL-APP ›

Name	Date modified	Type	Size
build	03/04/2024 13:20	File folder	
dist	03/04/2024 13:20	File folder	
nbproject	01/04/2024 17:11	File folder	
src	01/04/2024 17:12	File folder	
test	01/04/2024 17:13	File folder	
build	01/04/2024 17:11	Microsoft Edge HT...	4 KB
manifest.mf	01/04/2024 17:11	MF File	1 KB
mysql-connector-j-8.3.0	13/12/2023 03:08	Executable Jar File	2,438 KB
security.policy	06/04/2024 18:42	POLICY File	1 KB

You should then run the two programs normally and you will see execution of the distributed application over the three computers is achieved successfully.