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 II - Remote Method Invocation Three Tier Application with a Database Server

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

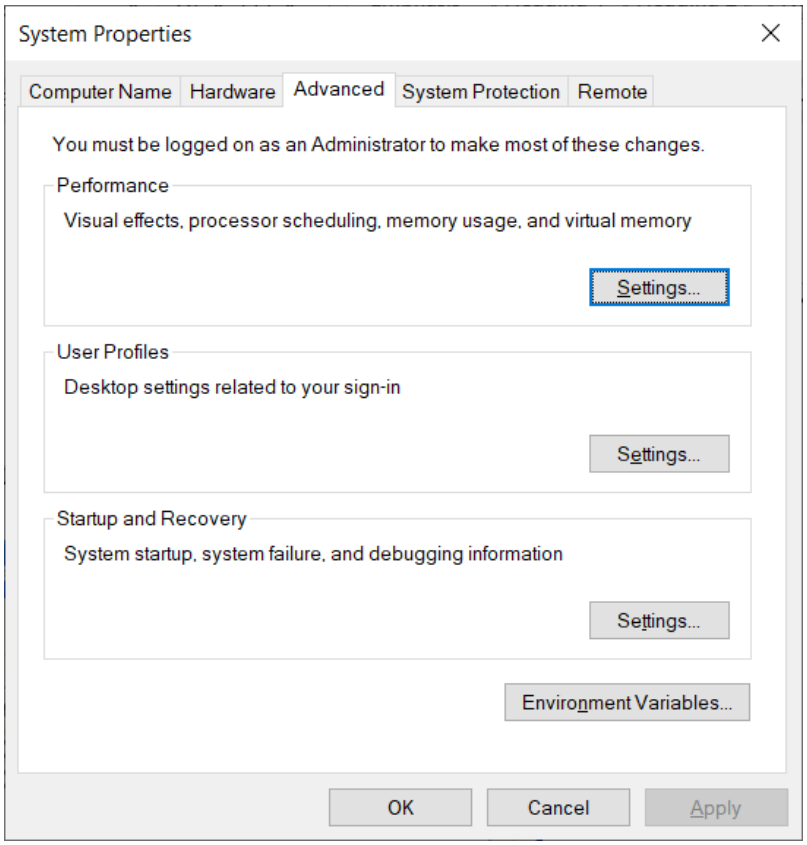
1. Document Purpose

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

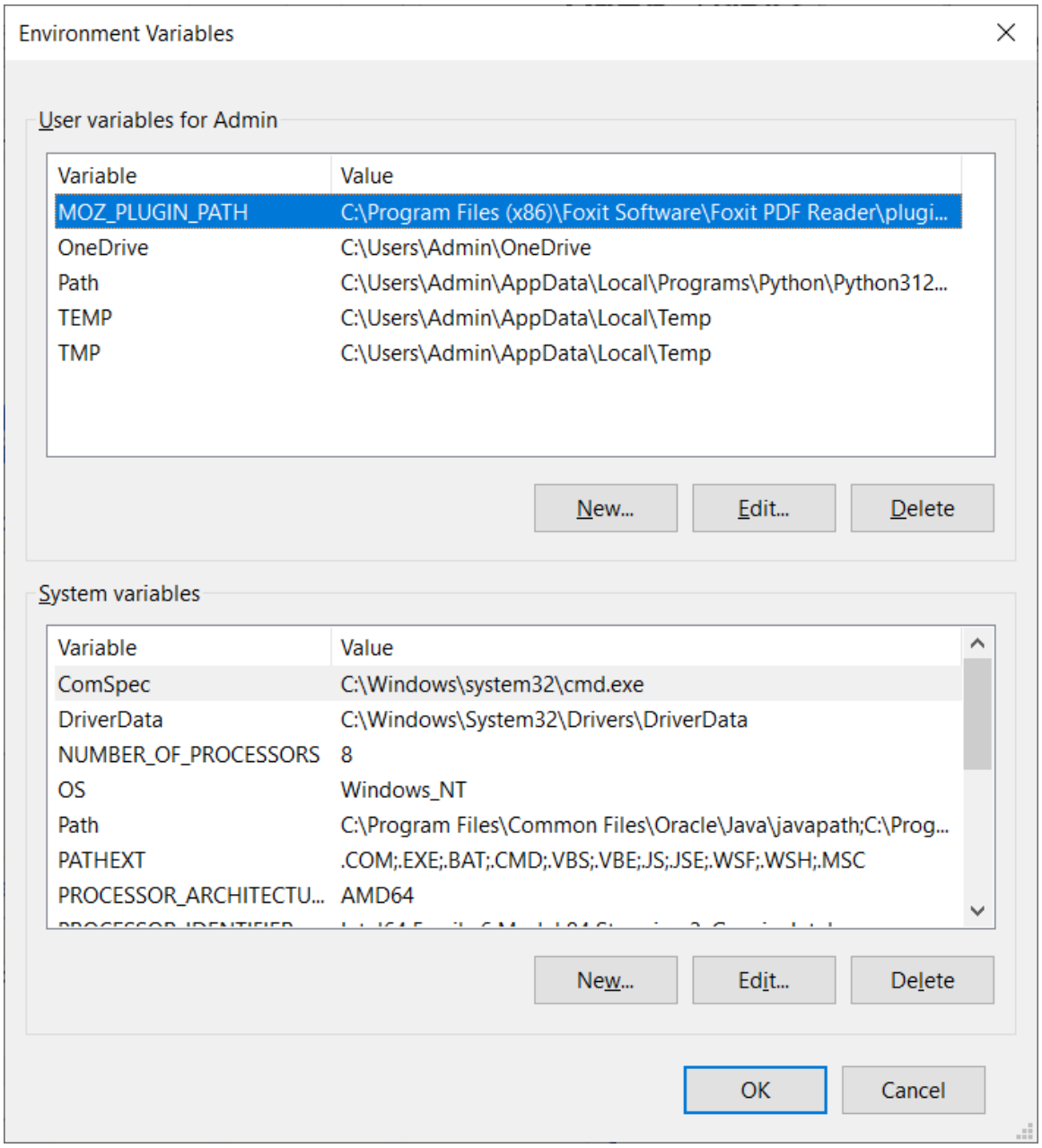
For running the programs, a correct configuration of the running environment is necessary (path and classpath variables).

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

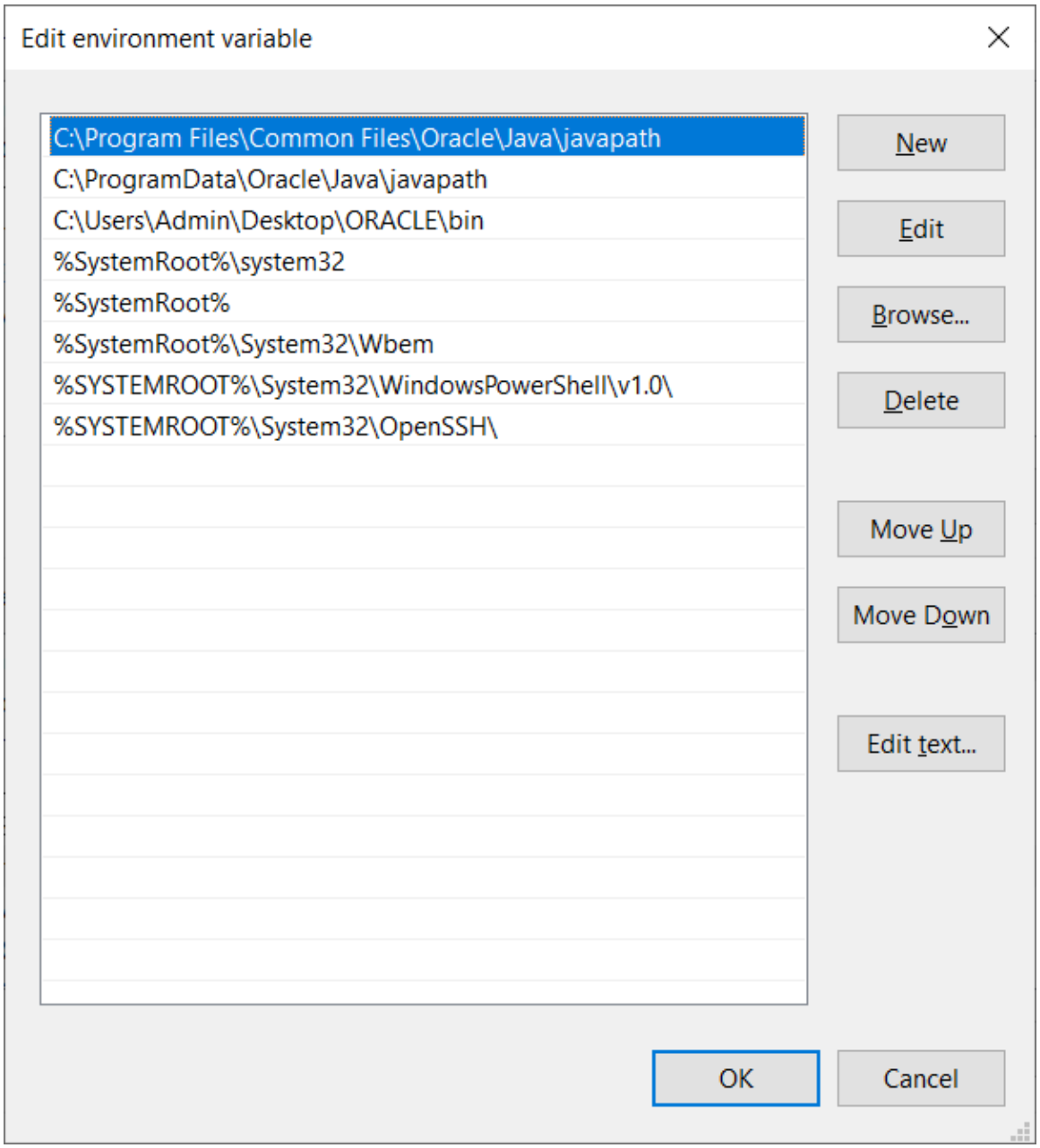
For running the programs, a correct configuration of the running environment is necessary (path variable).



Click on Environment Variables.



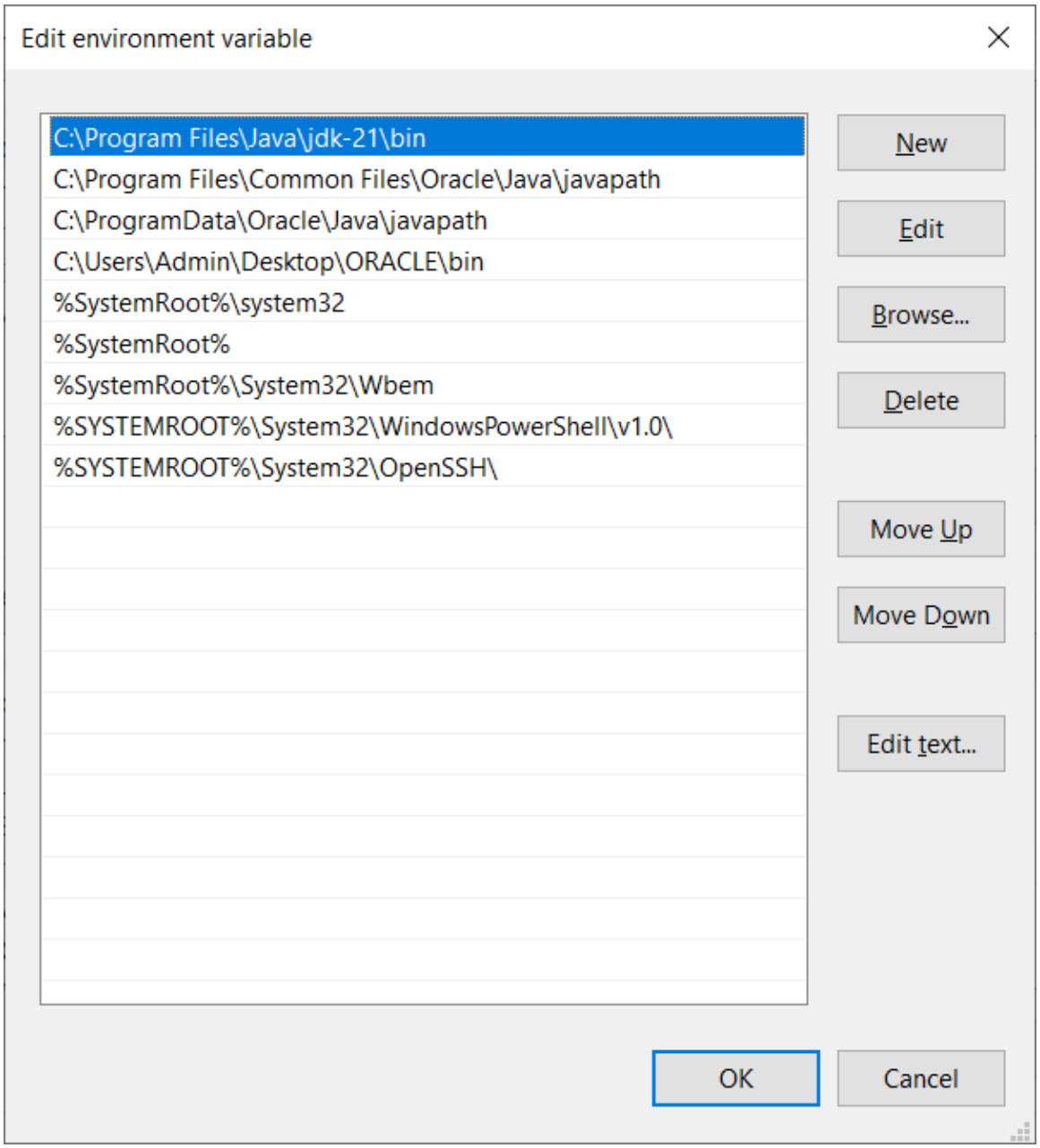
Find the Path system variable and click Edit.



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:

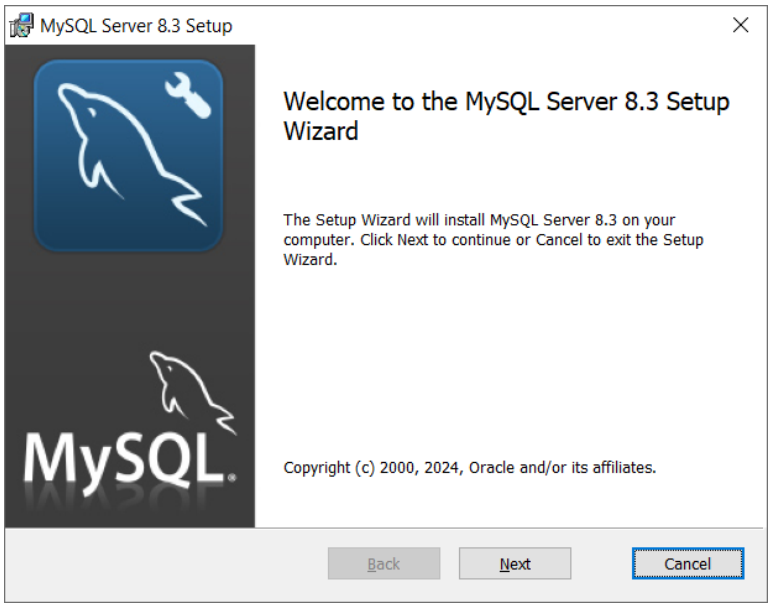


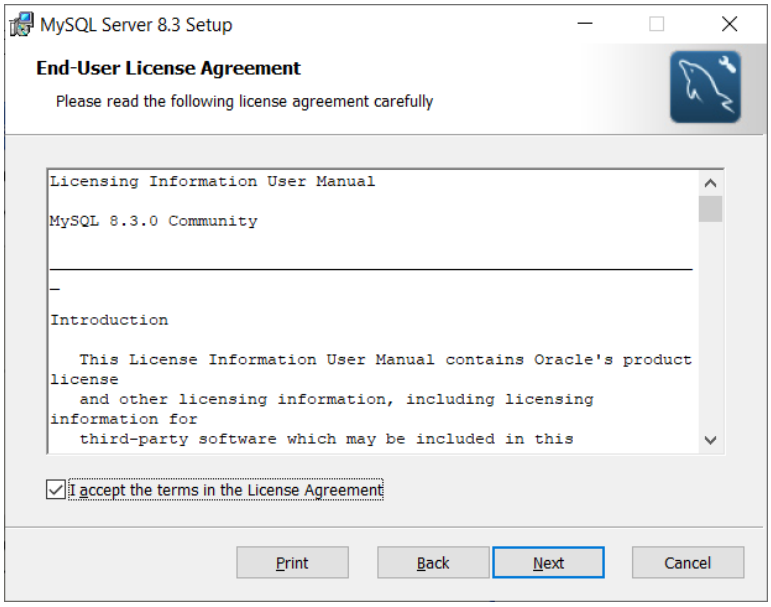
Download and Install MySQL Server

MySQL Community Server 8.3.0

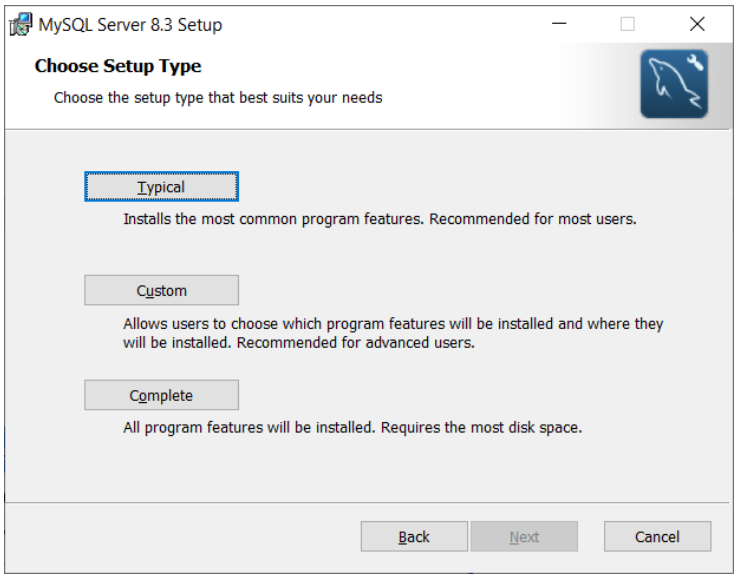
https://dev.mysql.com/downloads/installer/

After you download use MySQL Server Instance Config Wizard

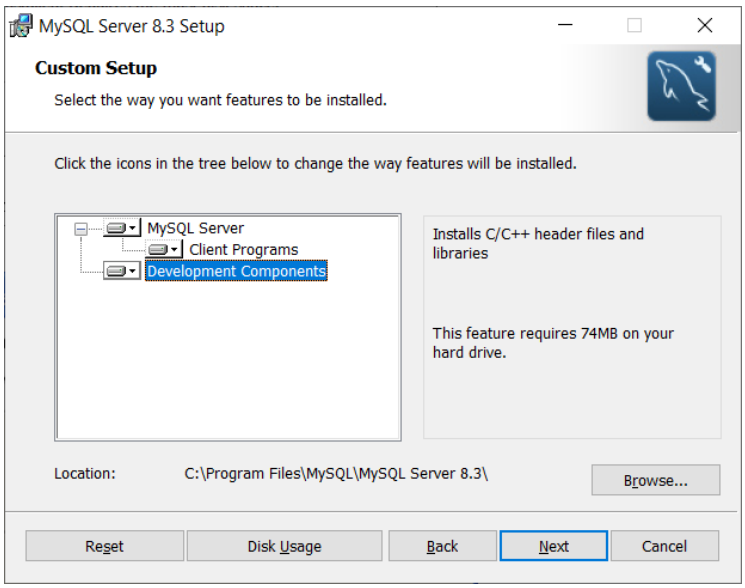




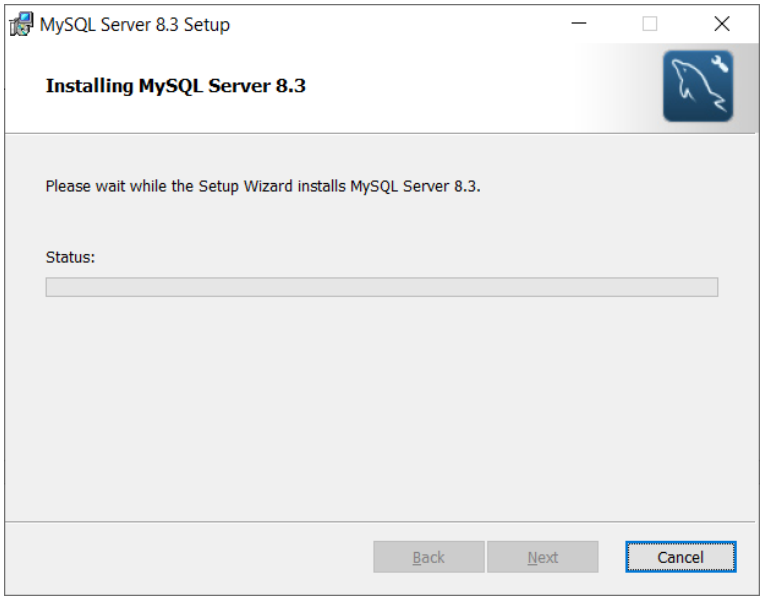
Choose the Custom configuration as follows:



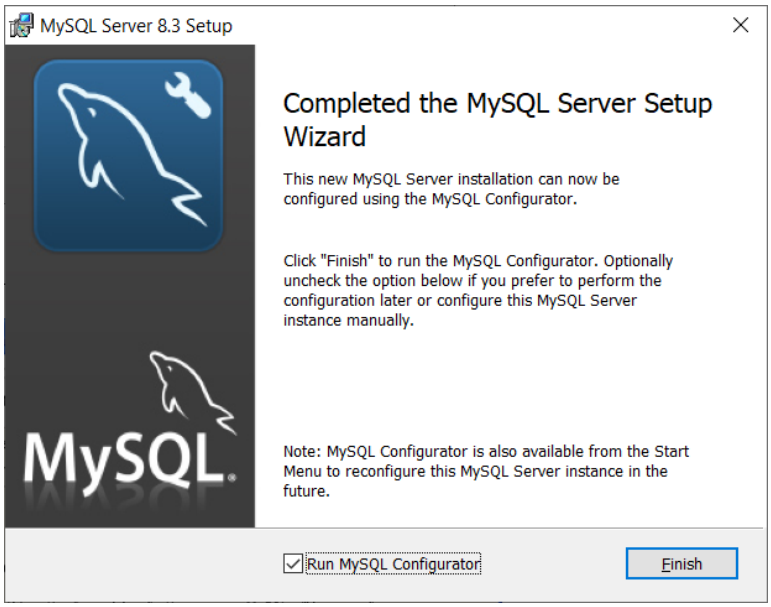
Install all on local hard drive in the following window:



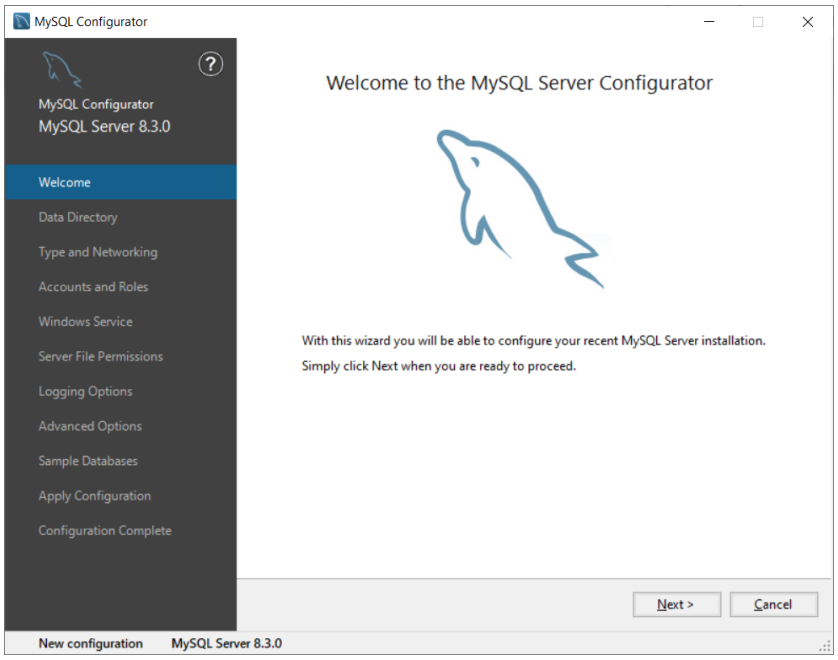
Proceed with next step as follows:



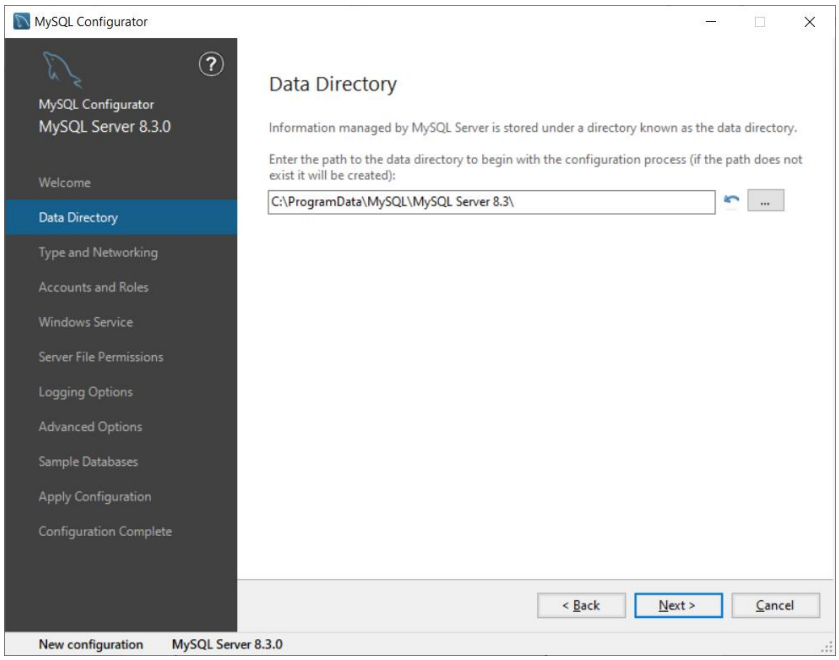
When you have finished copying the files, proceed with the configuration as follows:



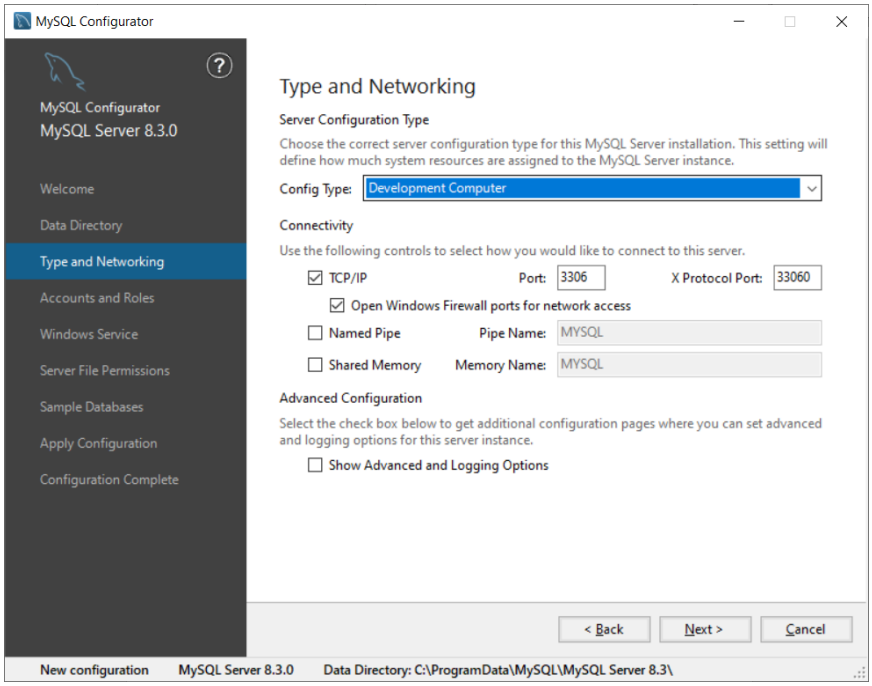
Click finish and continue with the configuration instance as follows:



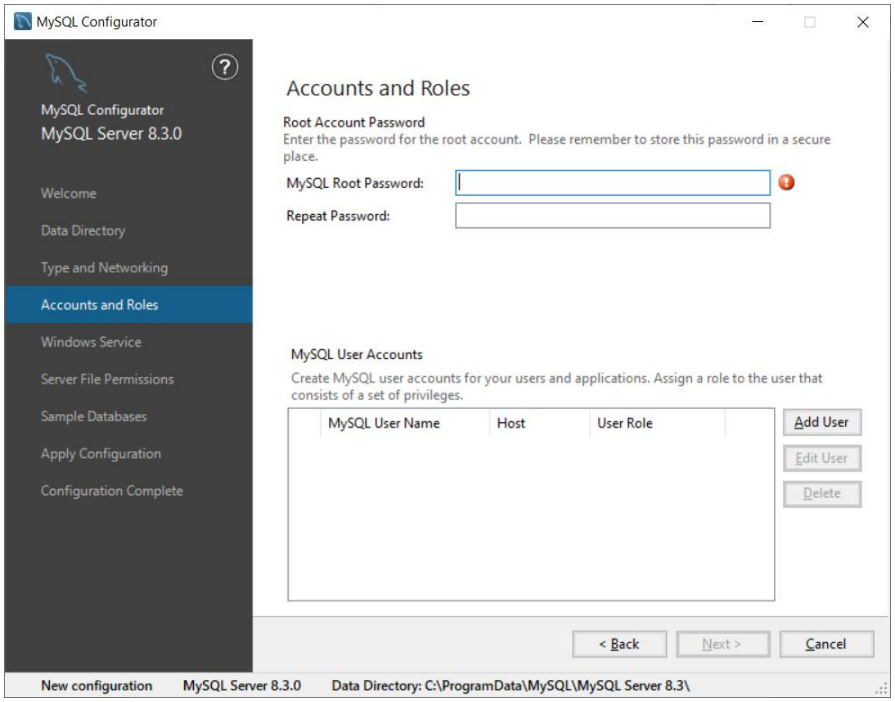
Choose the path of data storage:



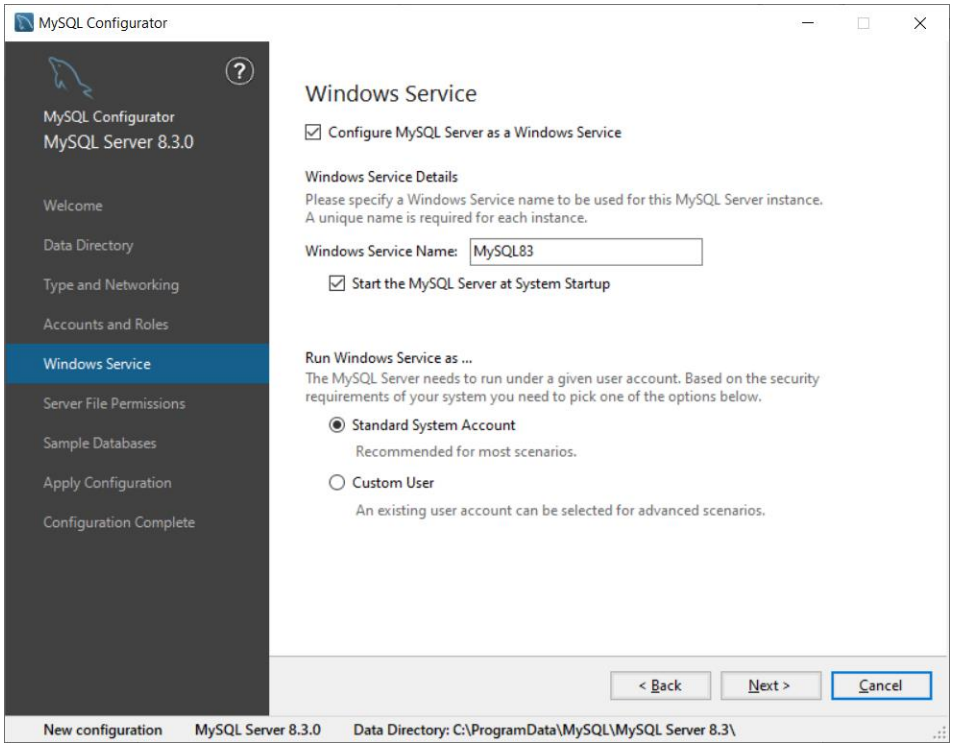
Choose Developer settings as follows:



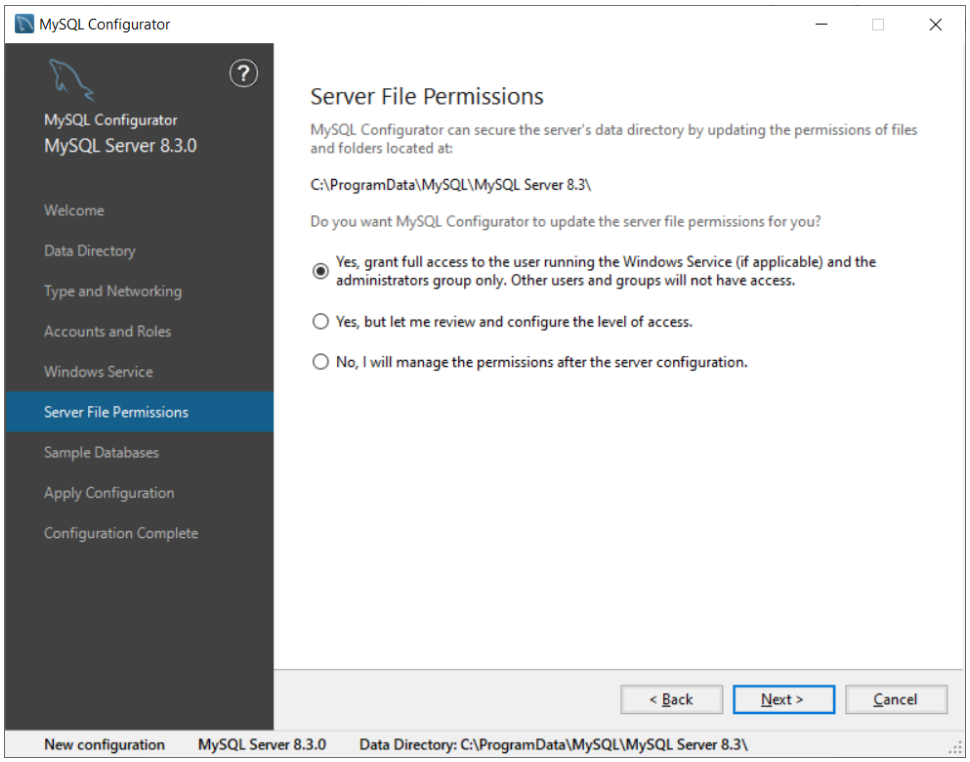
Choose the root password in the next window as follows (password: admin).



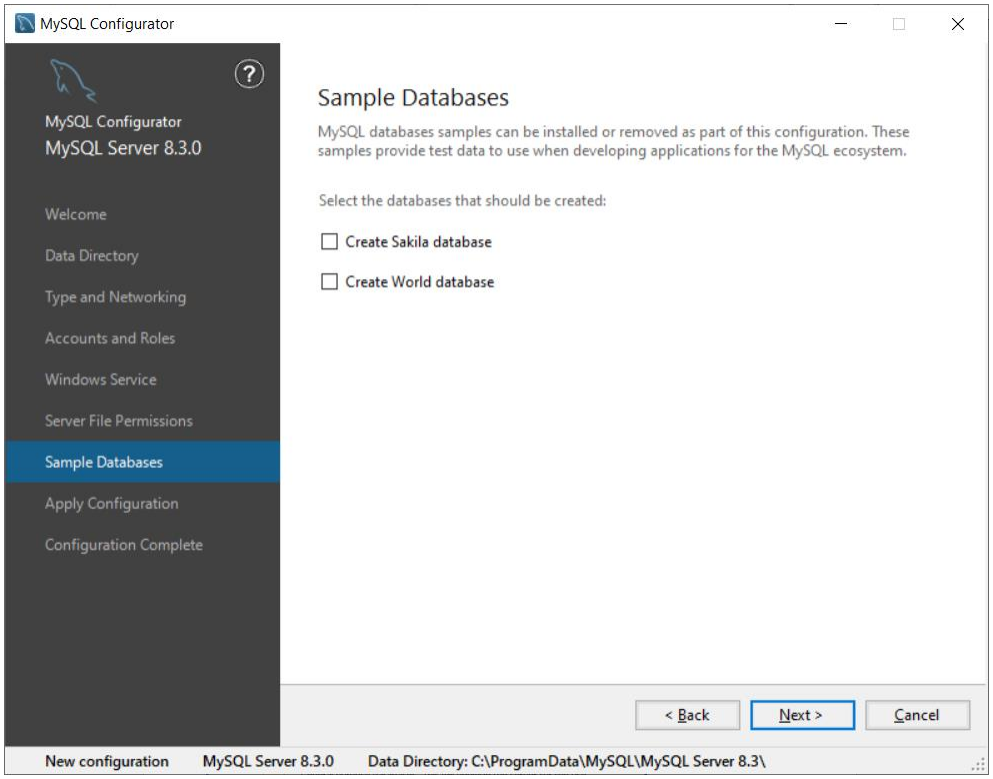
Configure MySQL as a service as follows:



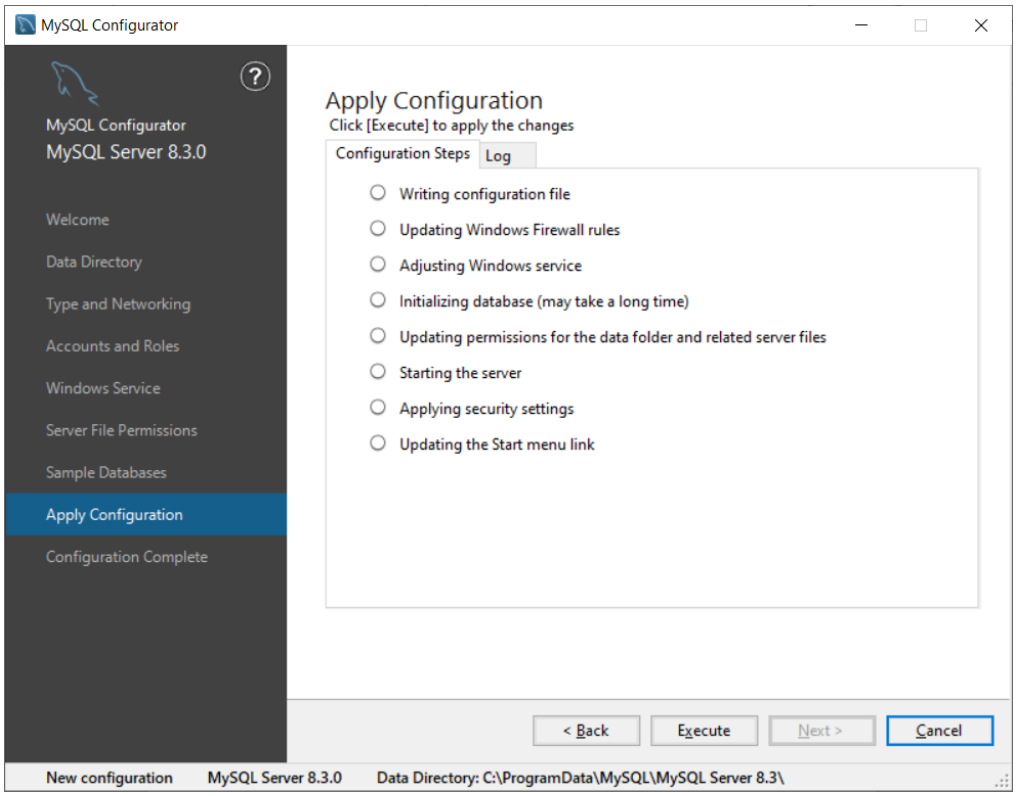
Grant rights to the users as follows:



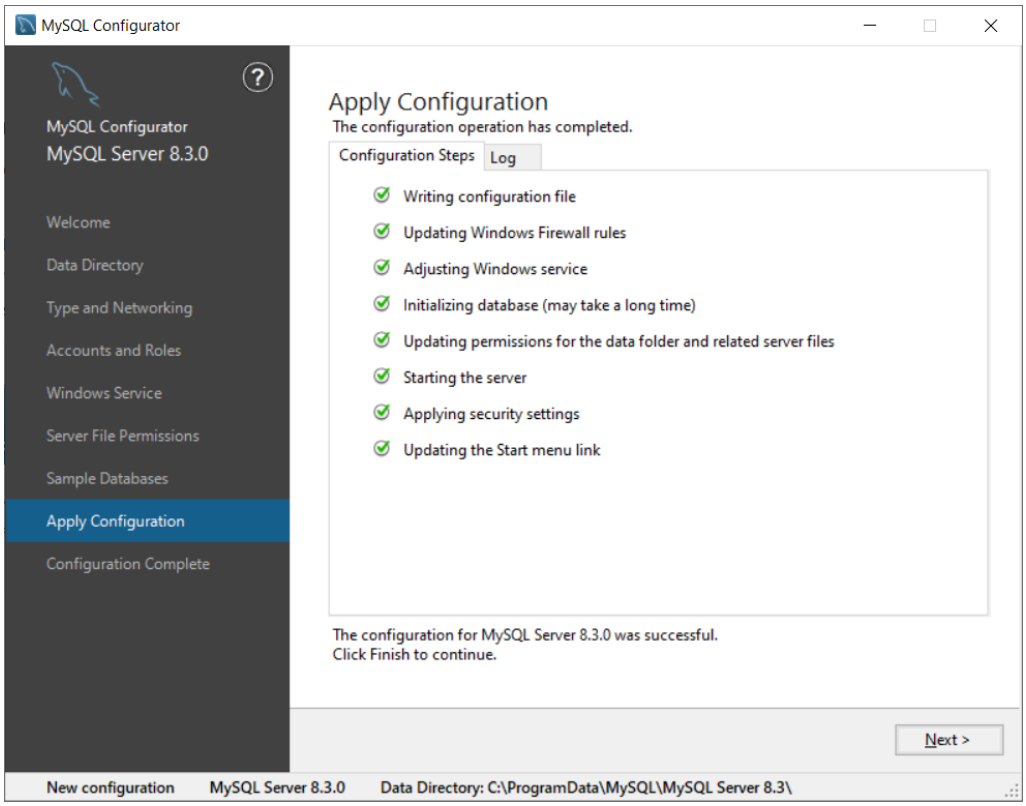
You may choose to install sample databases or not:



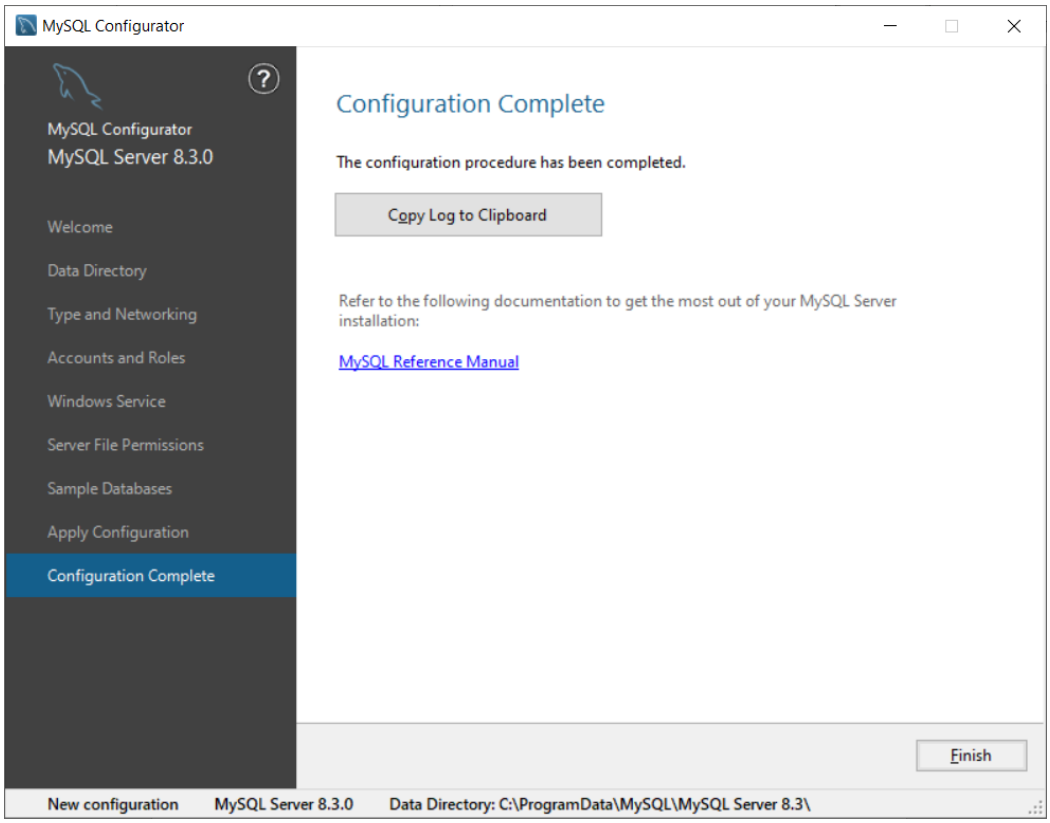
In the final window wait for the setup program to complete all the points as follows:



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

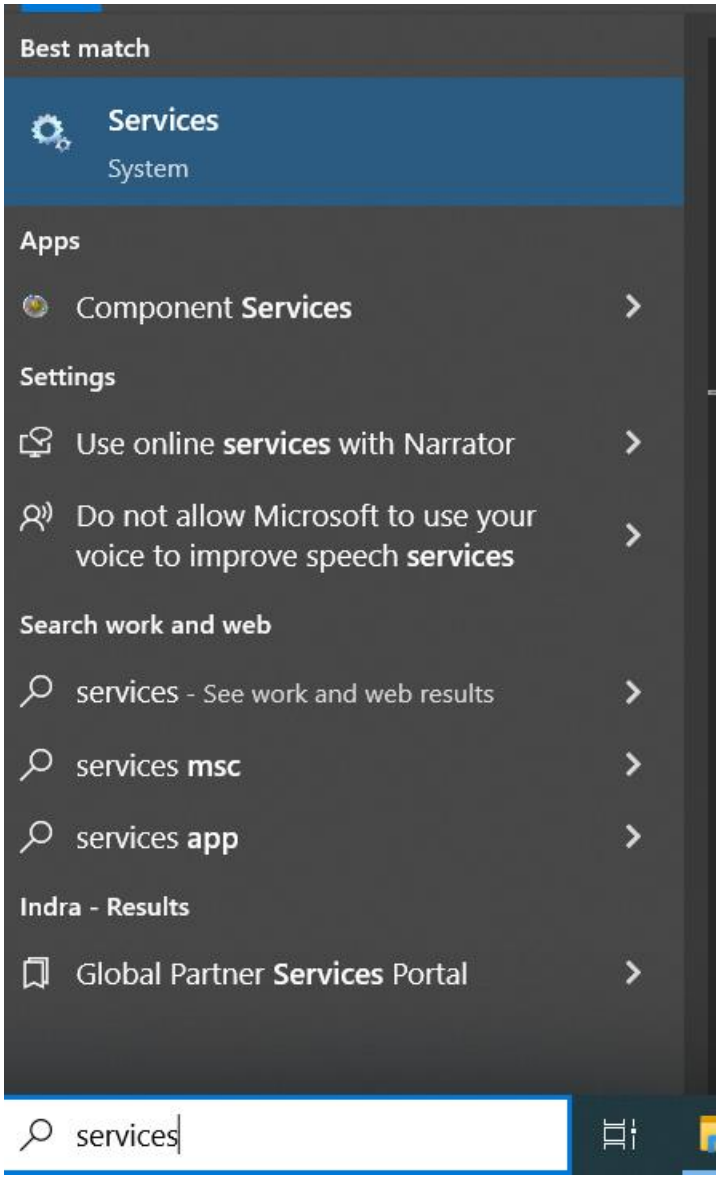


You may want to copy the log of the setup procedure as follows:



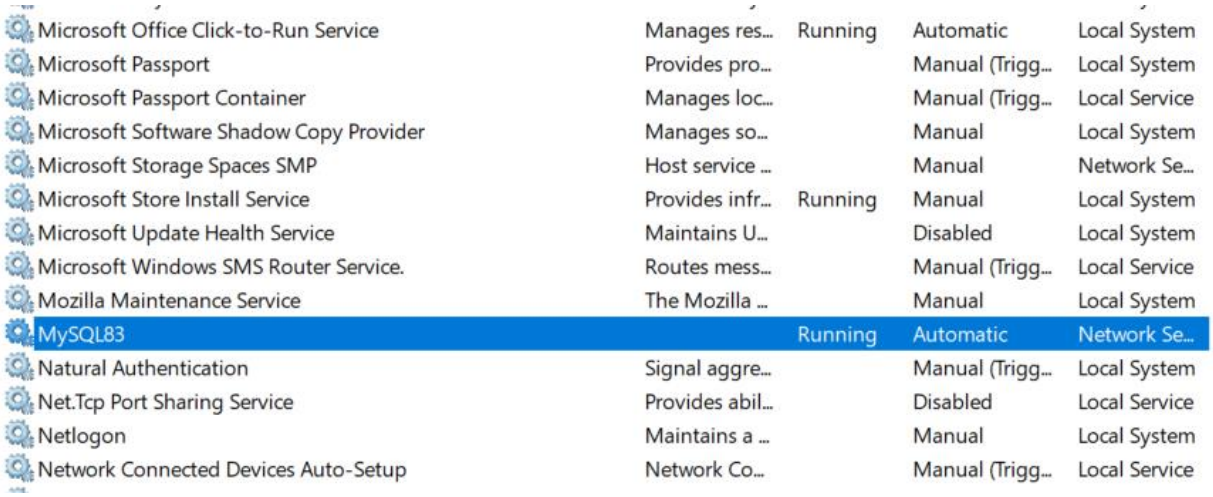
To verify that all has been setup correctly and the service is working fine go to Services and check the MySQL service as follows:

Type Services in the search bar of Windows:



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

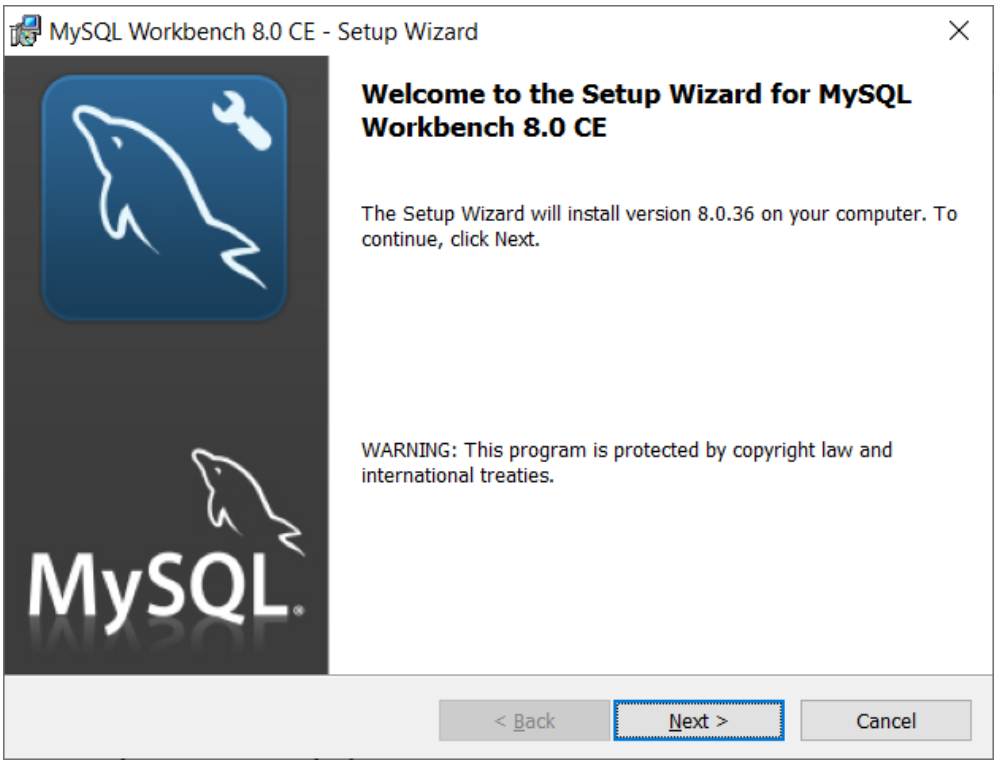
Download MySQL Workbench

MySQL Workbench 8.0.36

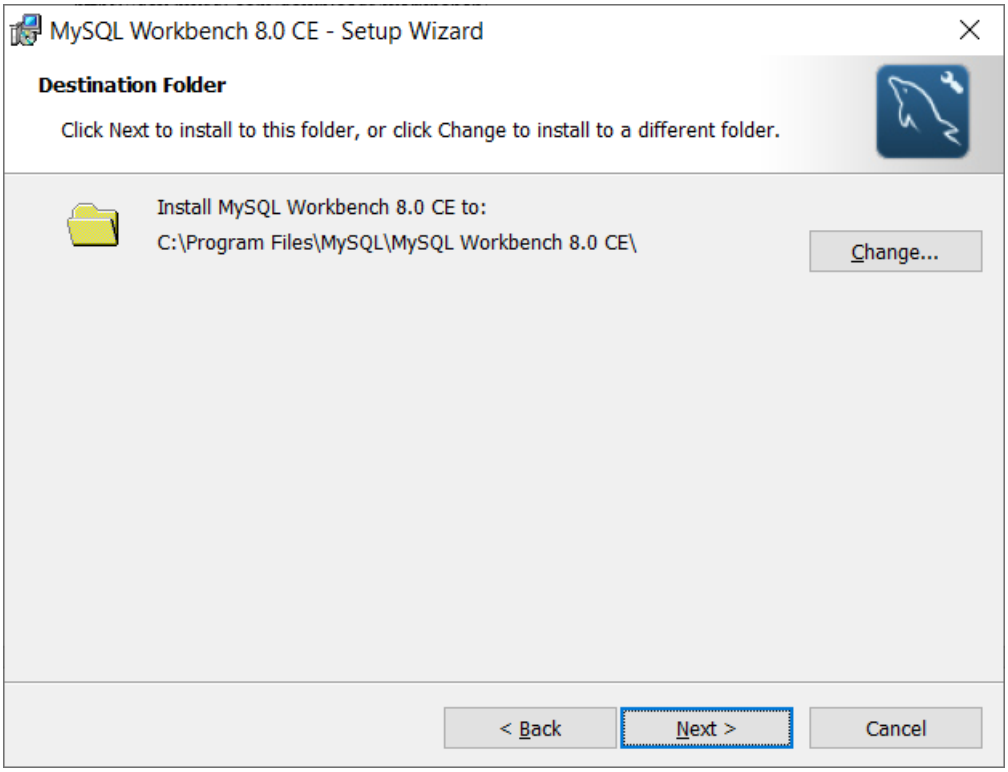
https://dev.mysql.com/downloads/workbench/

Run the setup file

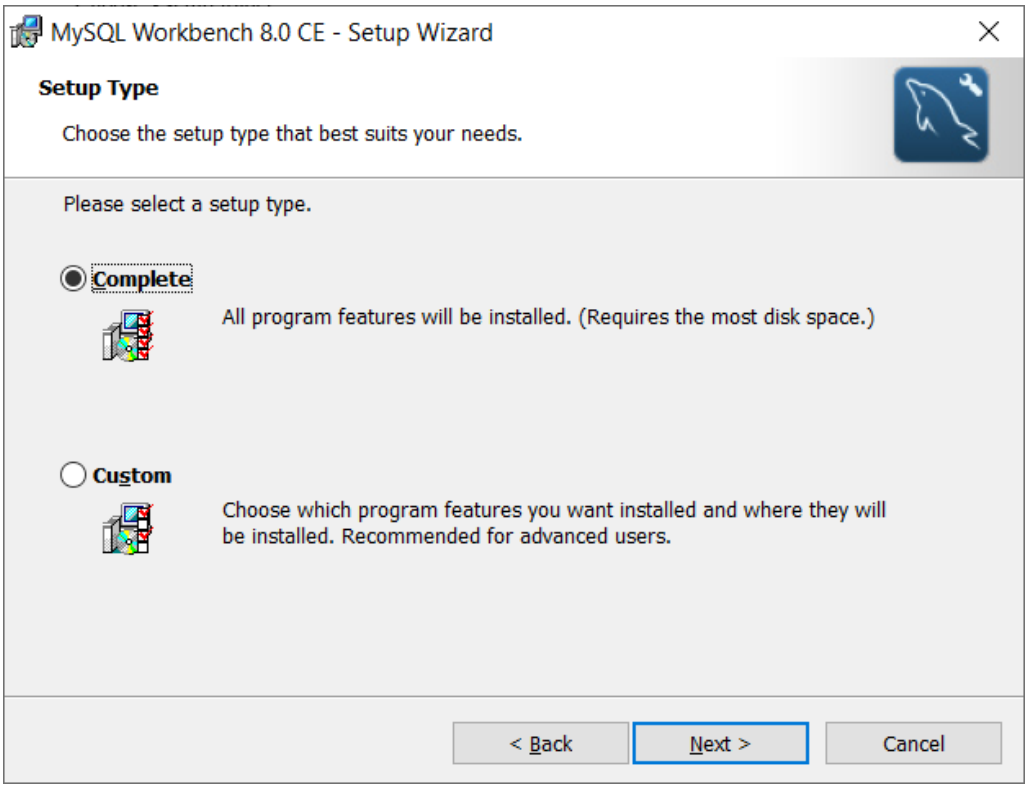
Continue with next on the following window:



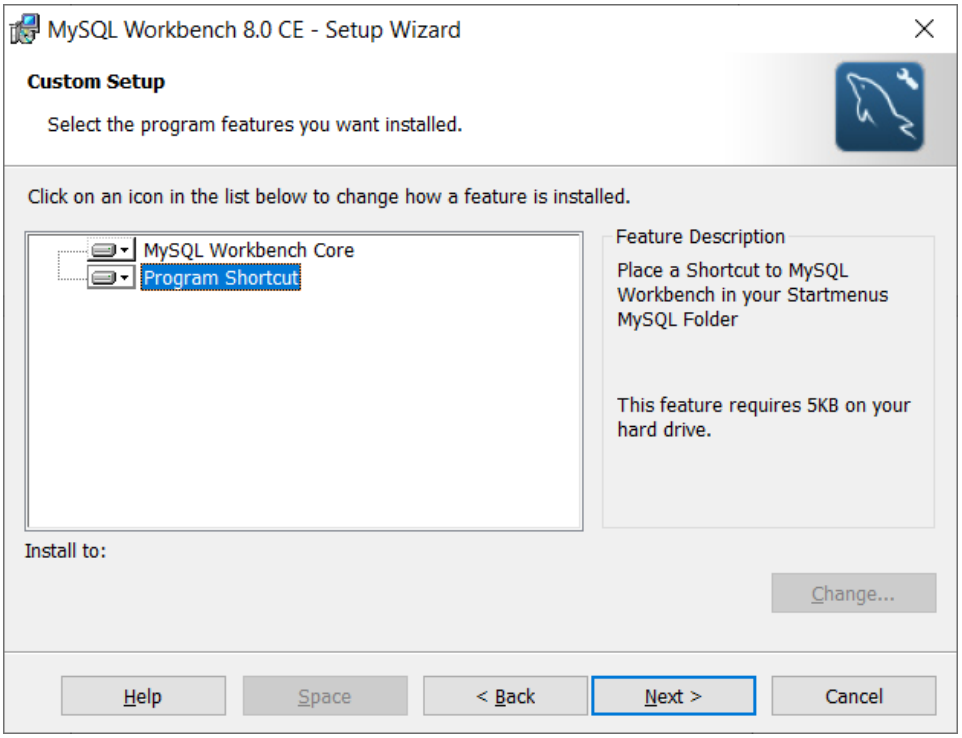
Choose a setup folder:



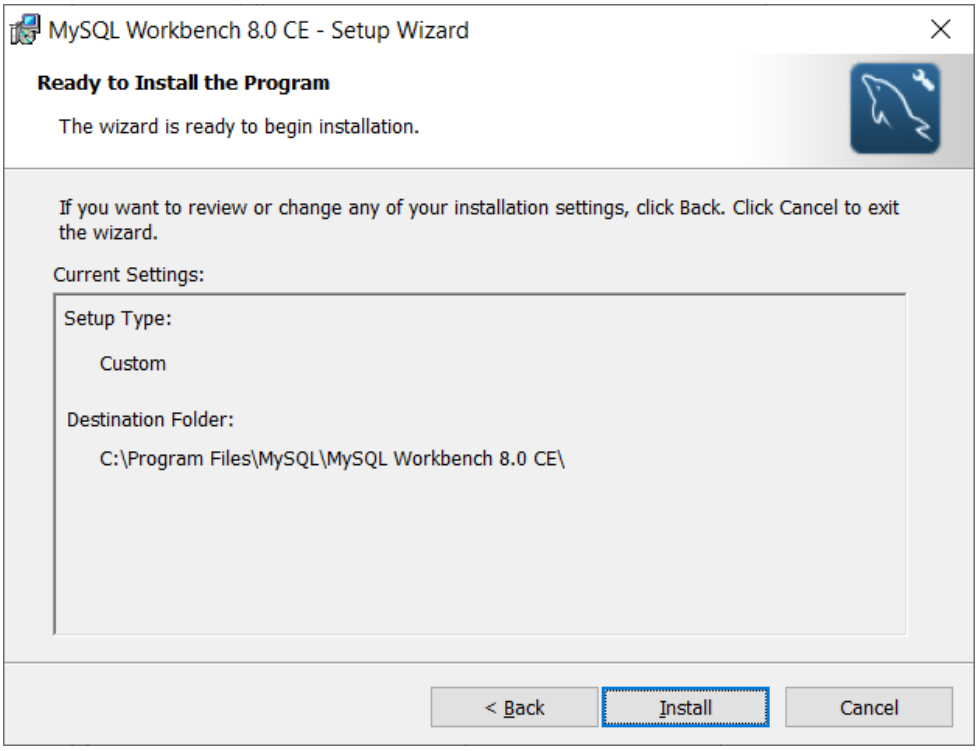
Choose Custom in the following window:



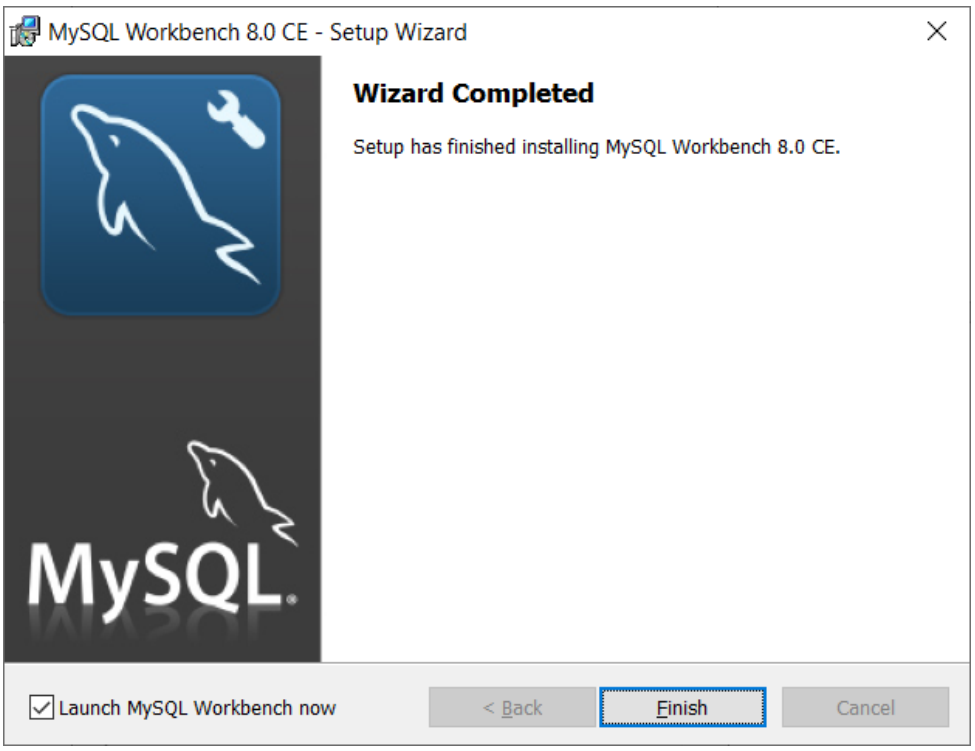
Choose to install all from local hard drive in the following window:



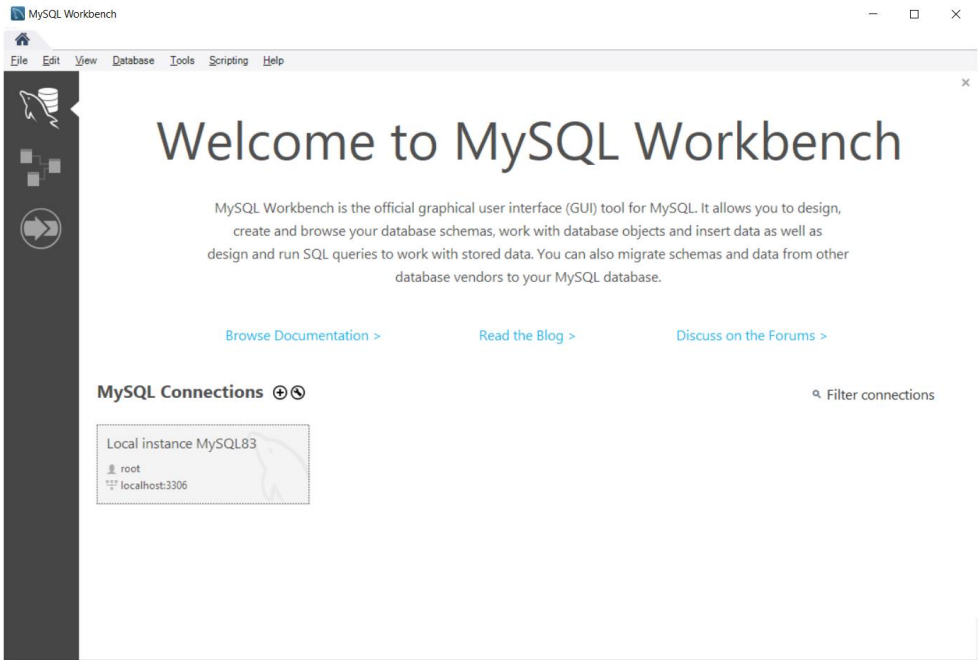
Continue with Install in the next window:



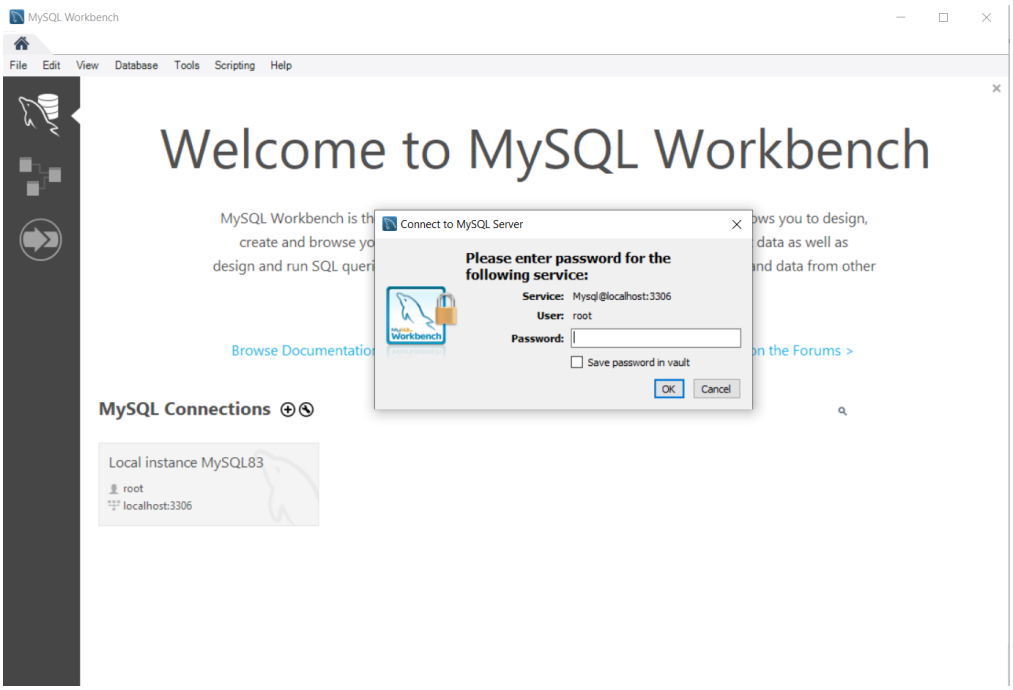
The setup program will complete the installation procedure and you should see the following window:



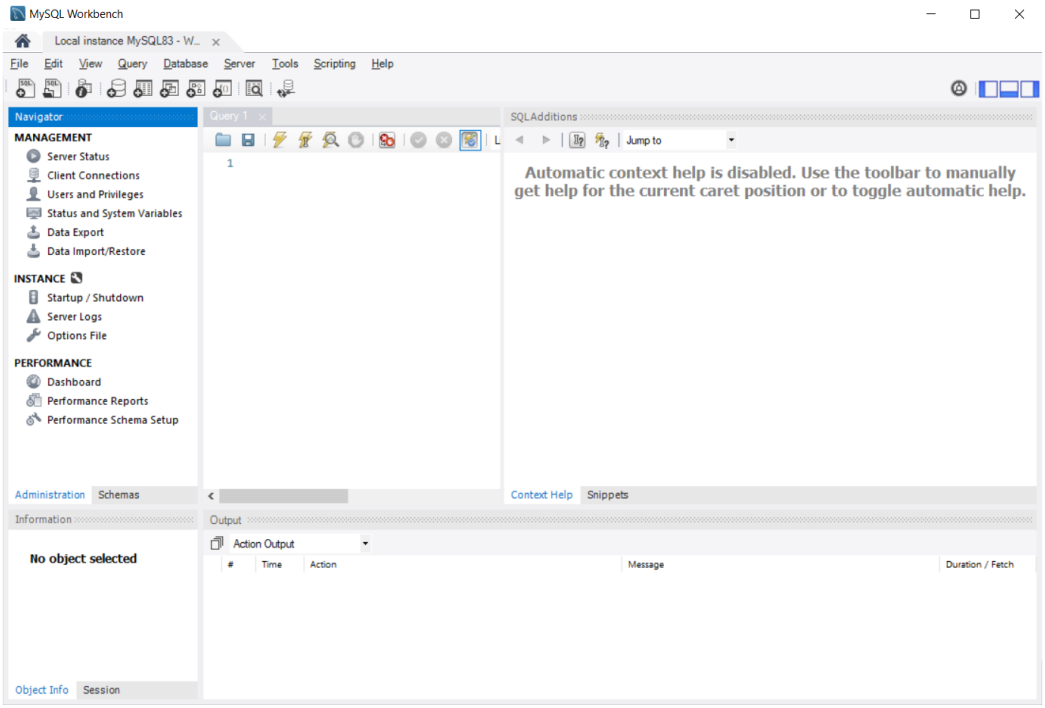
Click Finish and the program will start as follows:



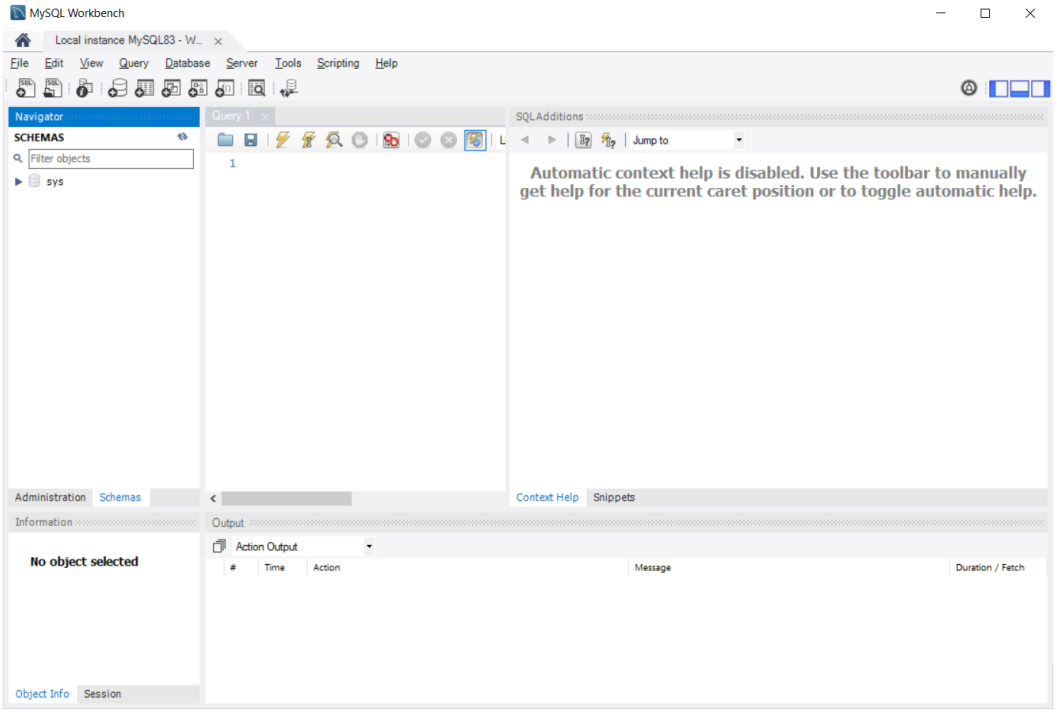
If you click on Local instance you will see the following:



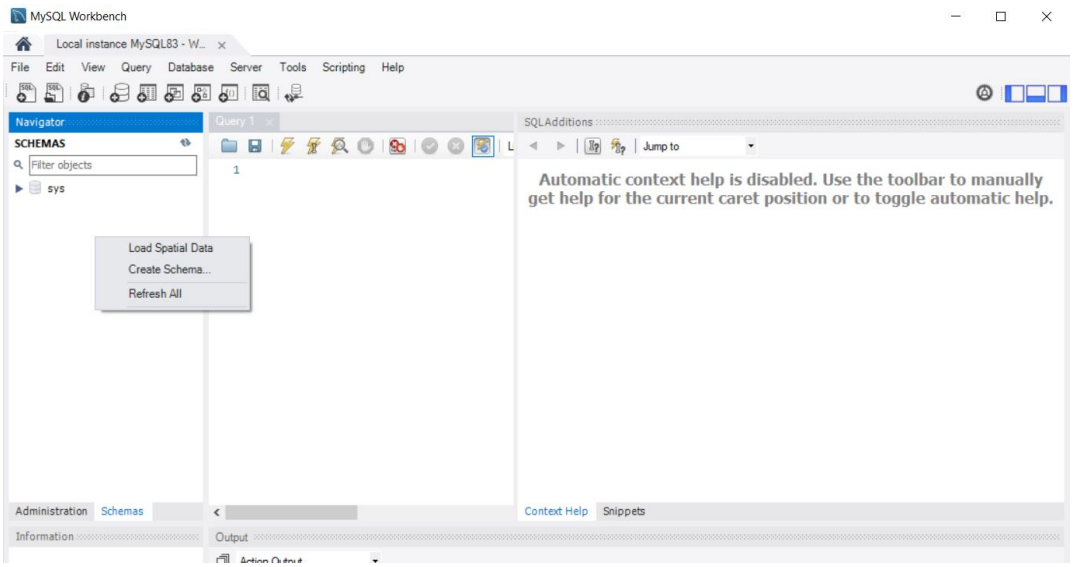
Insert the root password we used during installation of MySQL. You should see the following window:



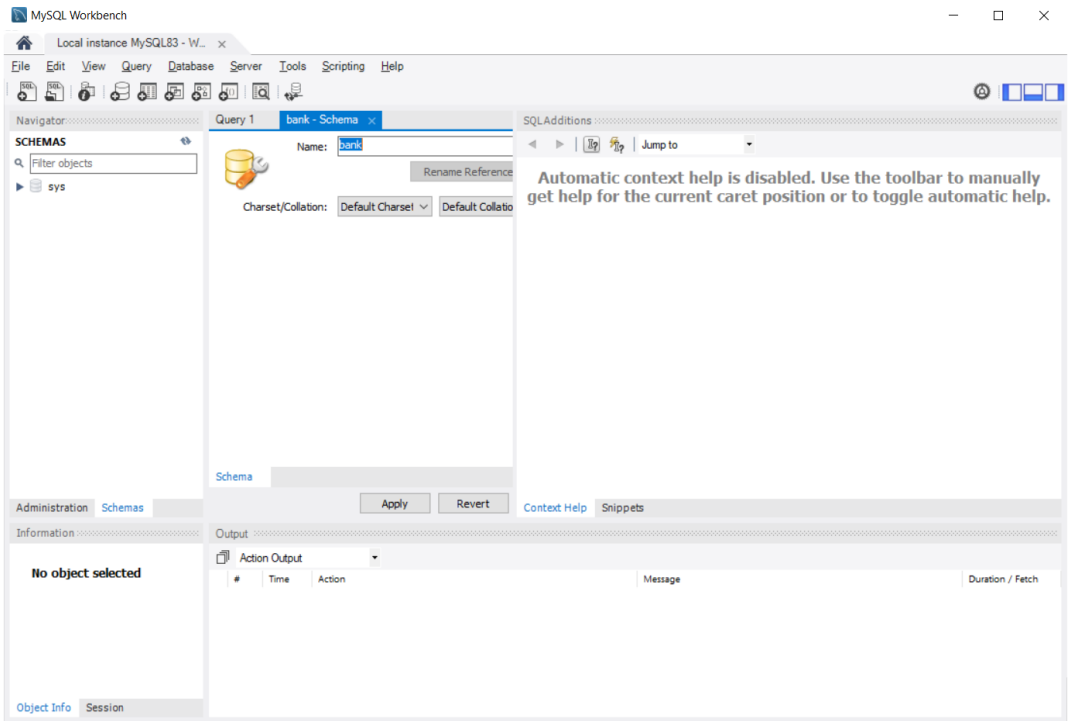
On the left panel, click on Schemas:



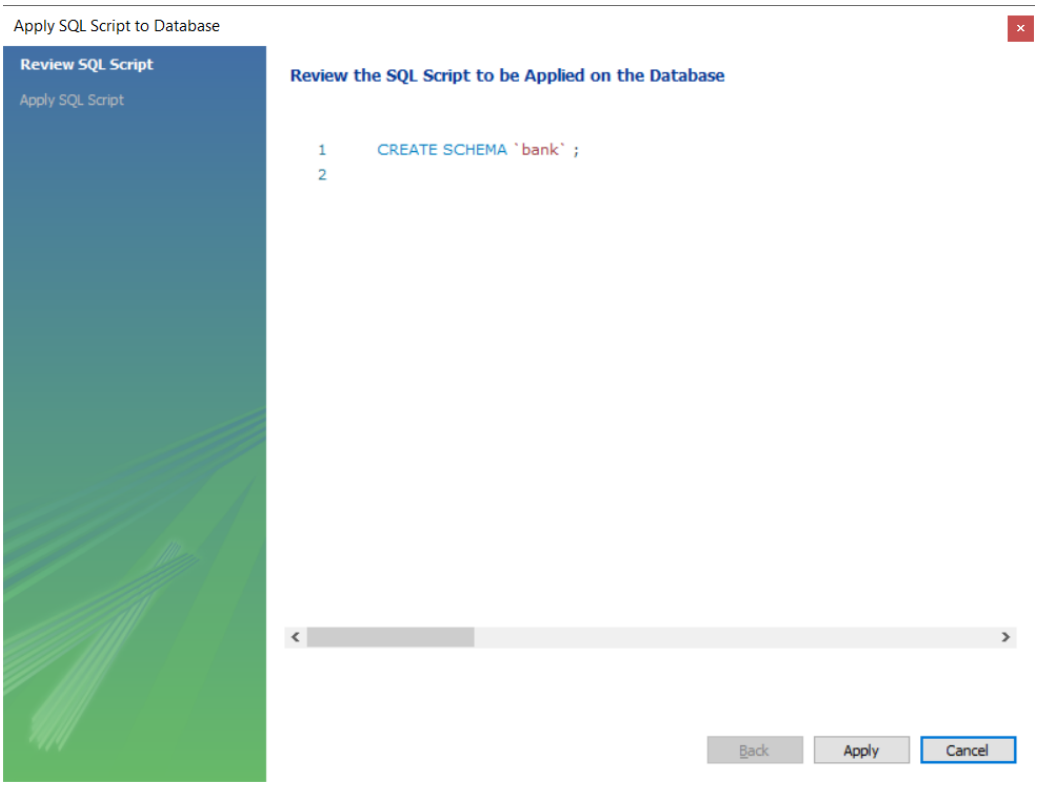
Click with the right button of the mouse on the left panel and select create schema:



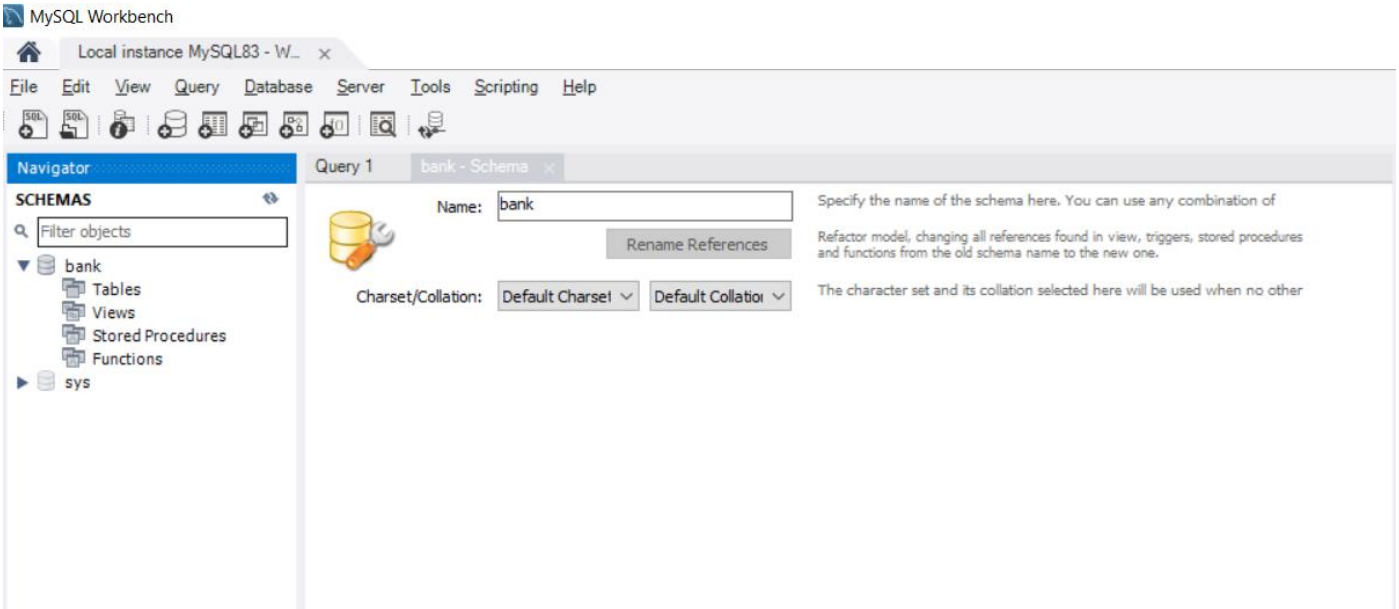
Name the database as Bank as follows:



Click Apply in the following window:

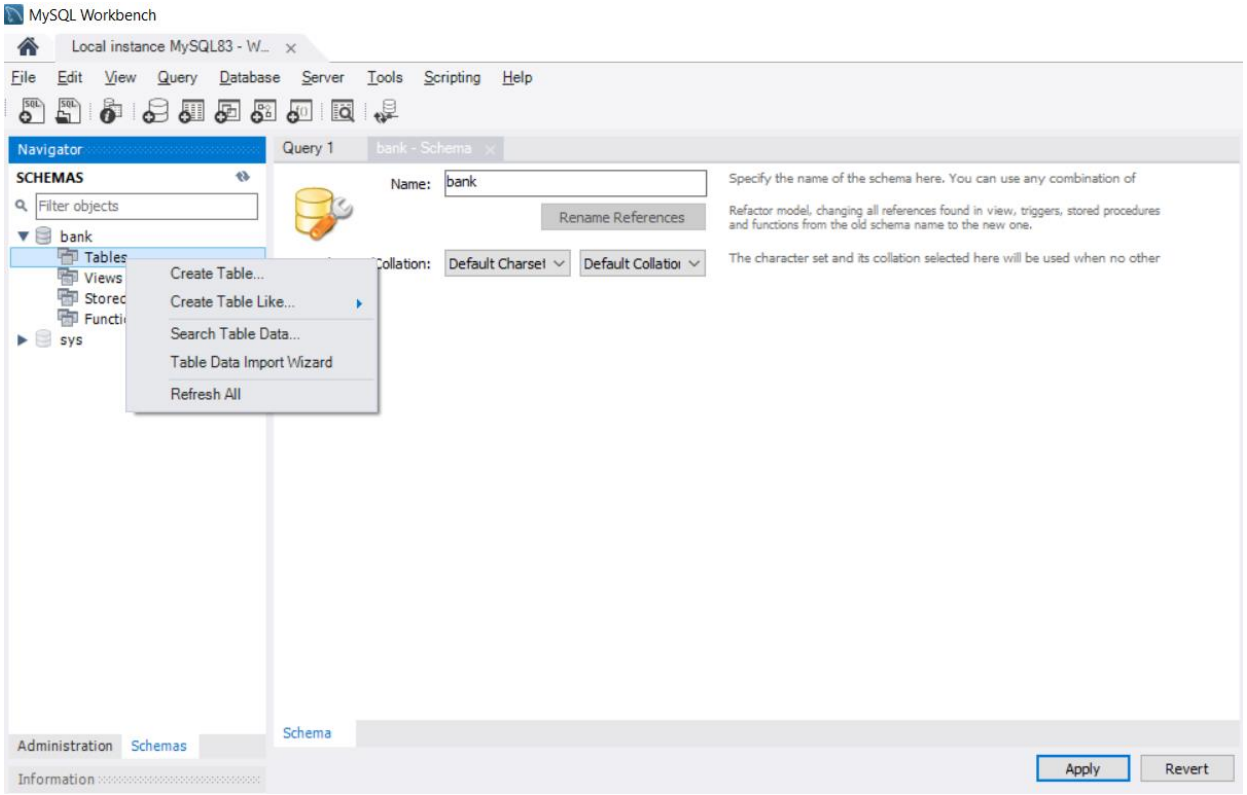


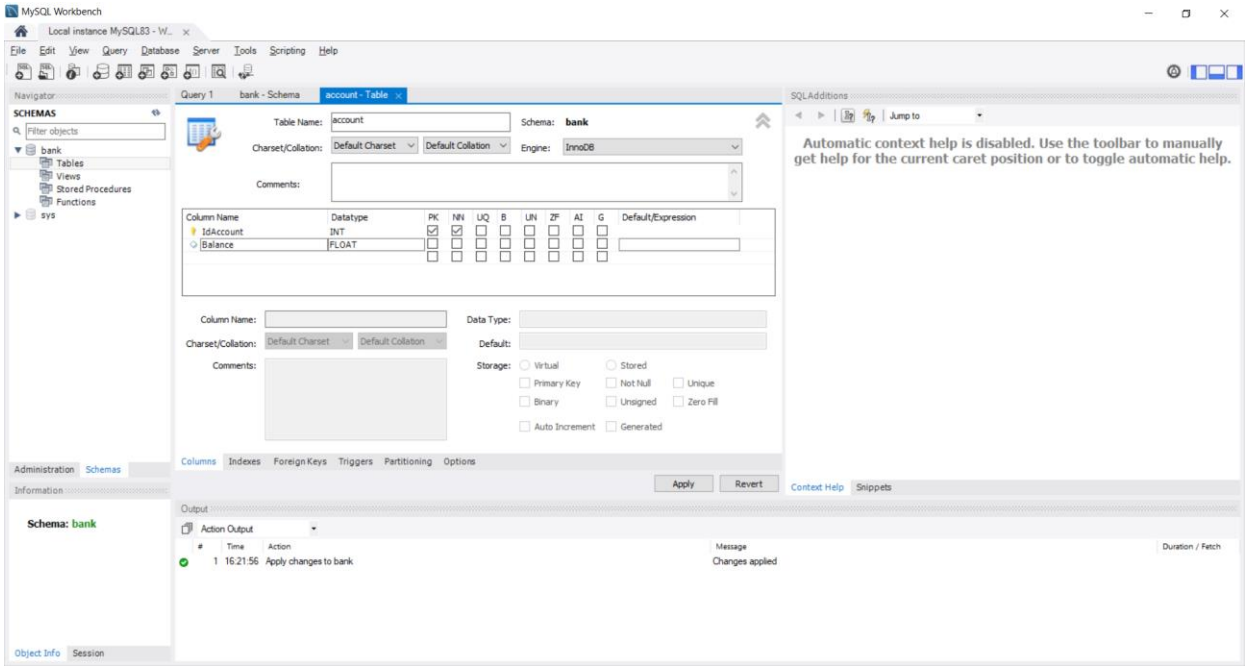
In the next window, click Finish and you will see the following:



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

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


MySQL Workbench
Local instance MySQL83 - W...
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
bank
Tables
Views
Stored
Functi
sys
Create Table...
Create Table Like...
Search Table Data...
Table Data Import Wizard
Refresh All
Query 1
bank - Schema
Name: bank
Rename References
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.
Collation: Default Charset Default Collatio
The character set and its collation selected here will be used when no other
Schema
Administration Schemas
Information
Apply
Revert


MySQL Workbench
Local instance MySQL83 - W...
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
bank
Tables
Views
Stored Procedures
Functions
sys
Query 1
bank - Schema
account - Table
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
Columns Indexes Foreign Keys Triggers Partitioning Options
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.
Context Help Snippets
Administration Schemas
Information
Schema: bank
Object Info Session
Output
Action Output
Time Action Message Duration / Fetch
1 16:21:56 Apply changes to bank Changes applied

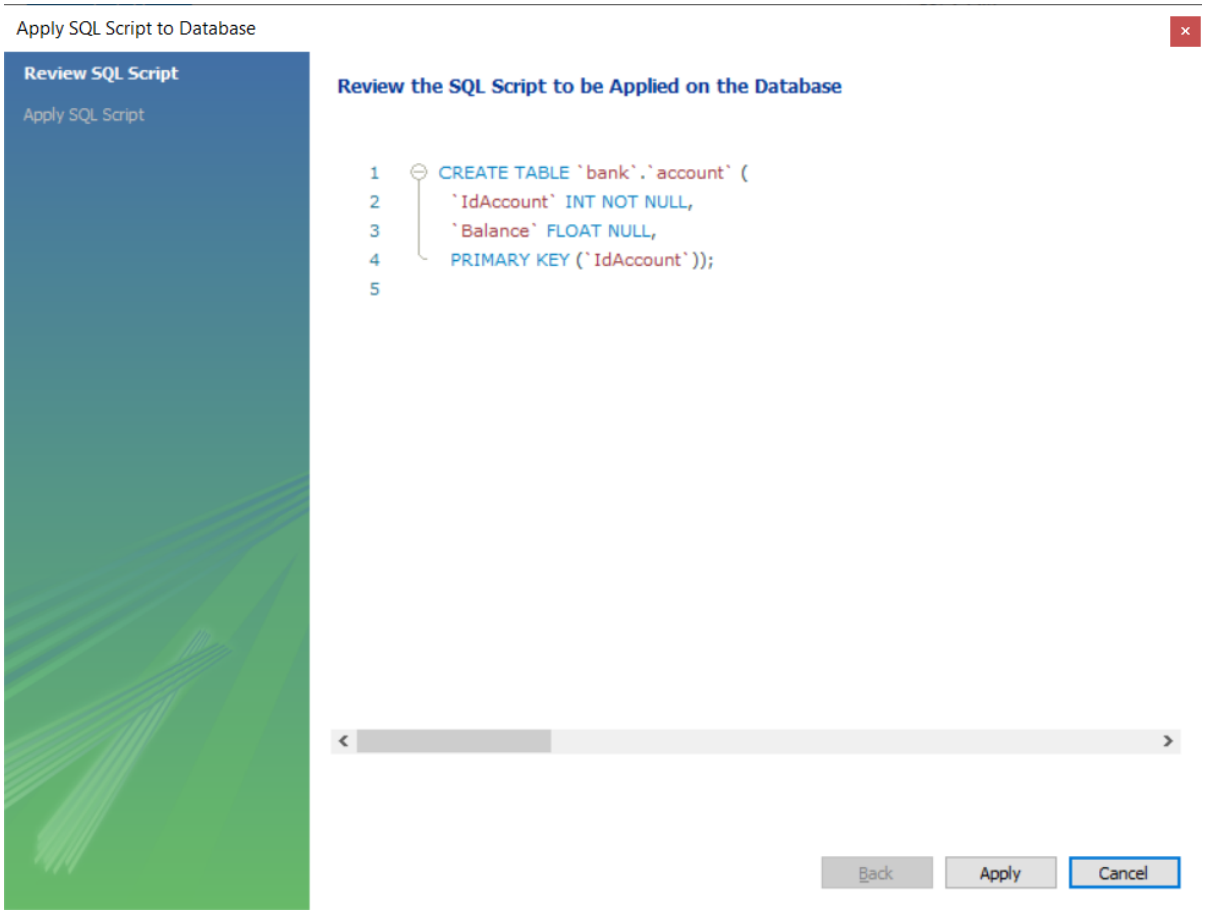


Table Customer
Fields: IdCustomer(int), Name (Varchar), Surname (Varchar)

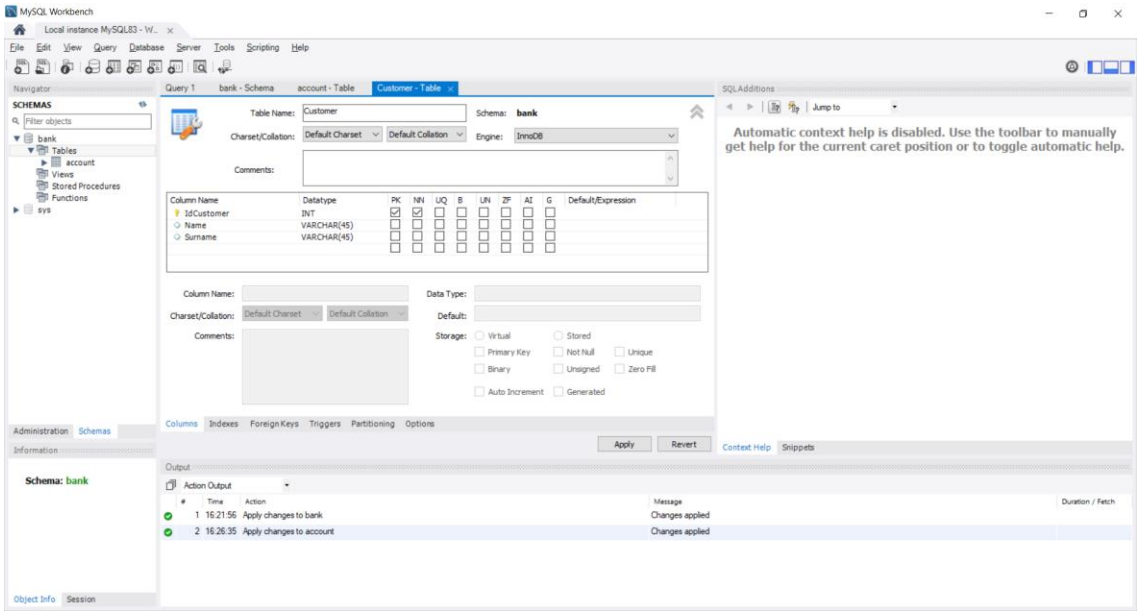
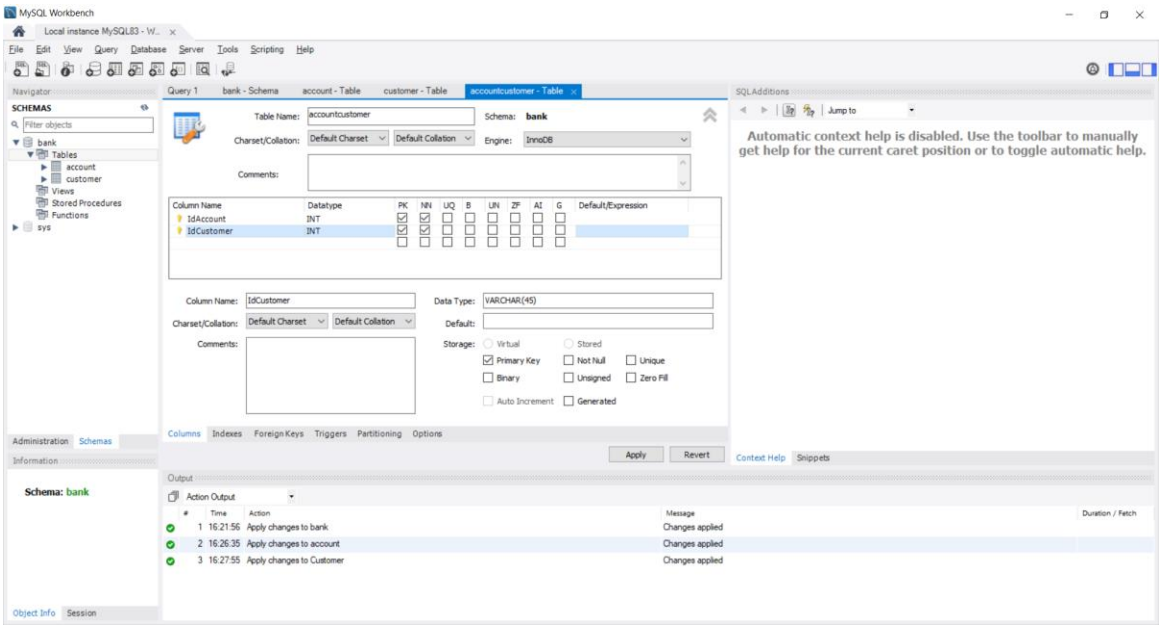
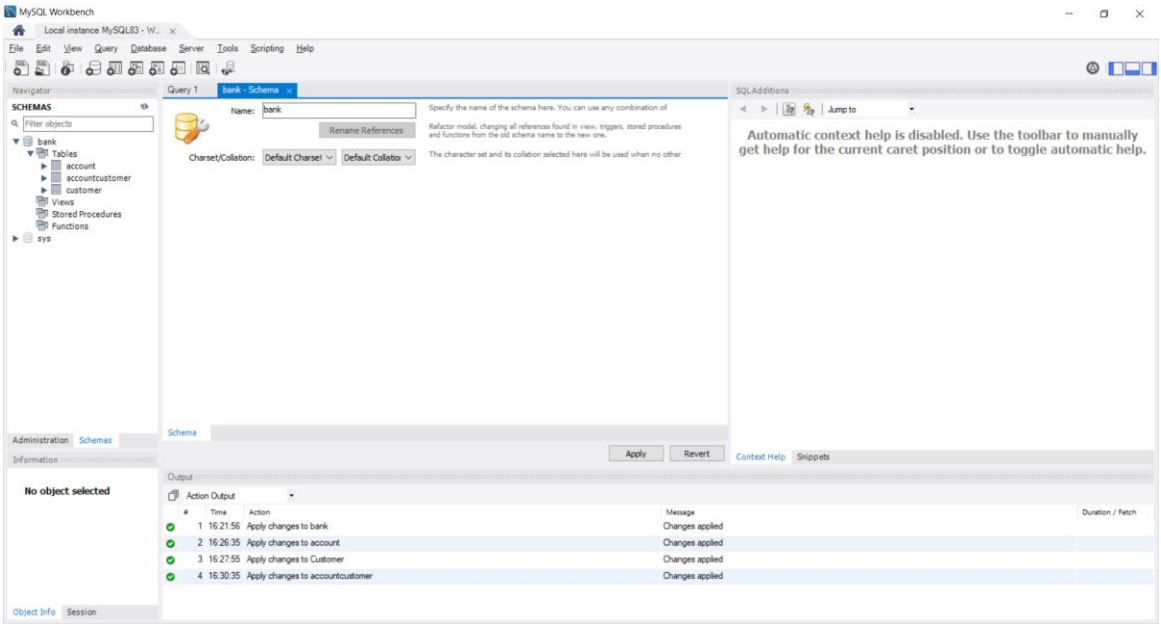


Table AccountCustomer
Fields: IdAccount, IdCustomer



Your database should now have three tables as follows:



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:

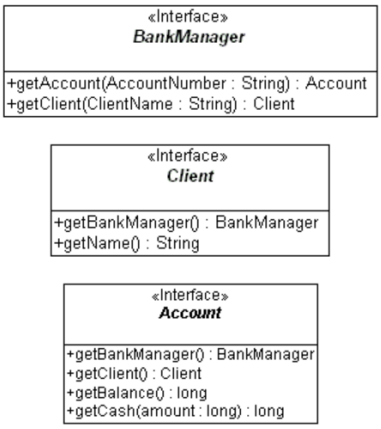
- The **RMI Registry** that hold references to the remote services.
- The **RMI host server** program that creates the remote services, registers them with the registry and waits for client requests.
- The **RMI client program.** A program that obtains references to remote service object from the RMI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



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.

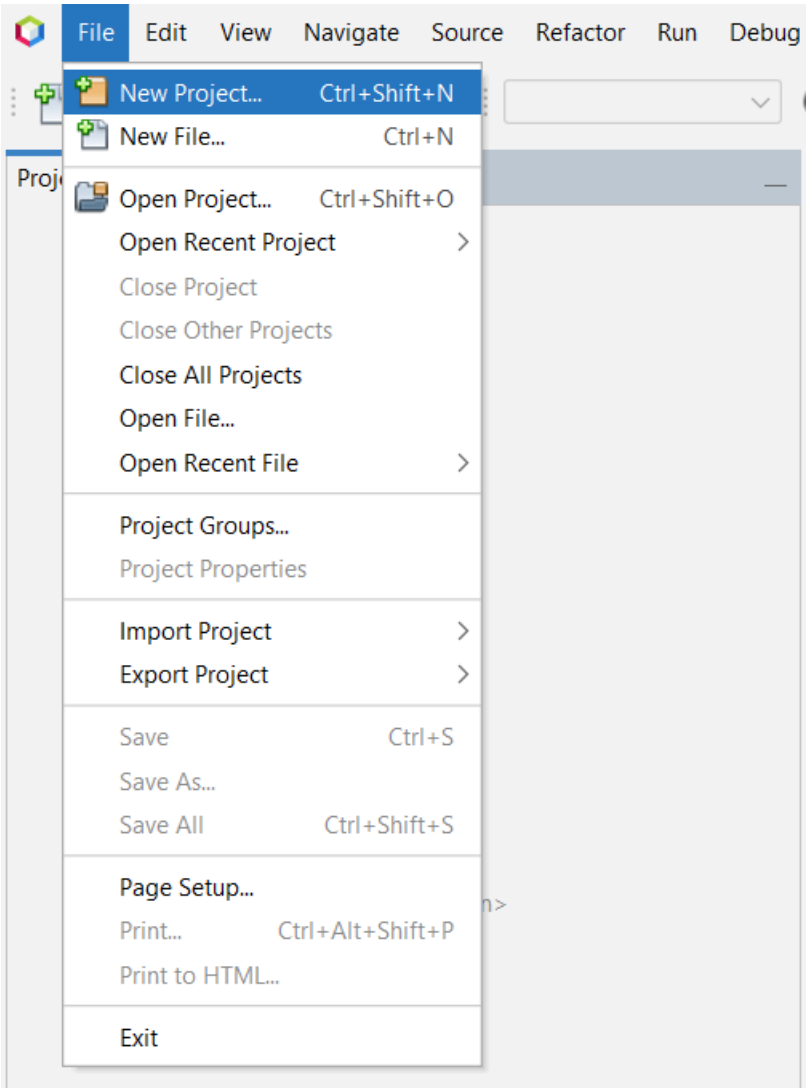
Educational goals:

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

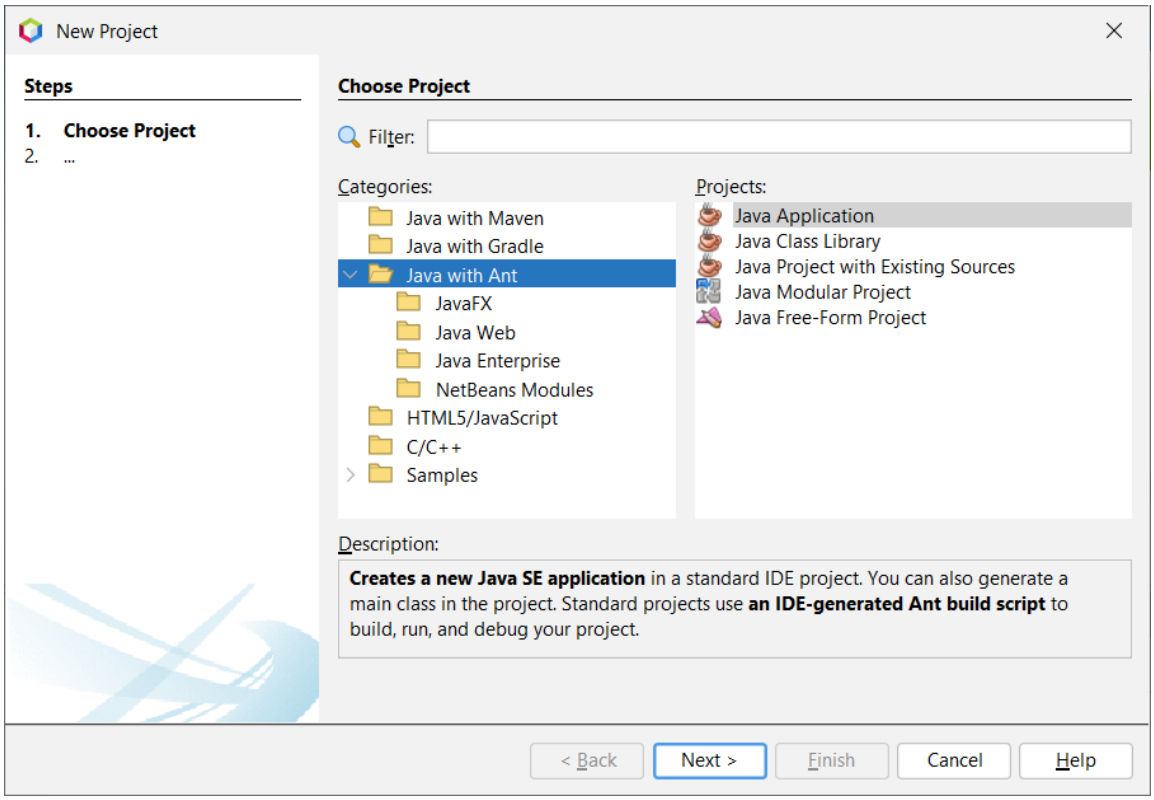
The **RMI Registry** manages the publication of the RMI remote services. You have to run a server program that creates the actual remote services, and finally, finish coding the program BankUser, which will use the RMI remote services.

Step 1: Code Development

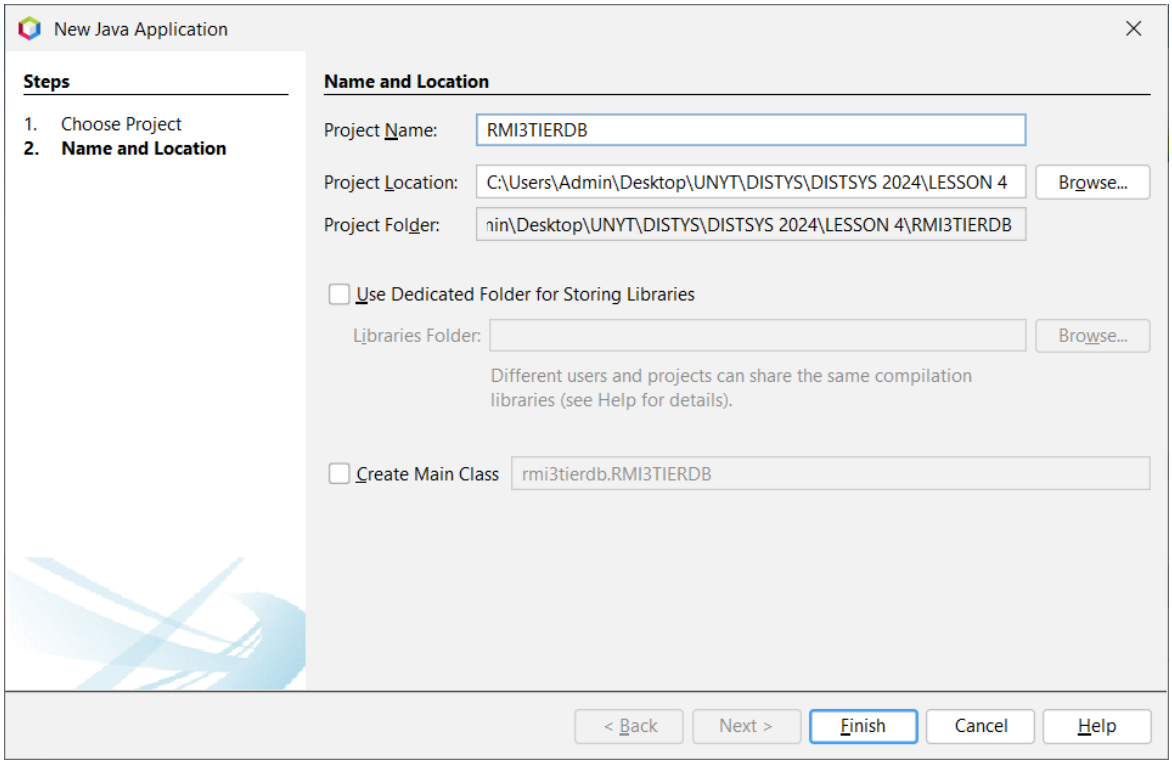
Create a new project in NetBeans as follows, in File, select New Project.

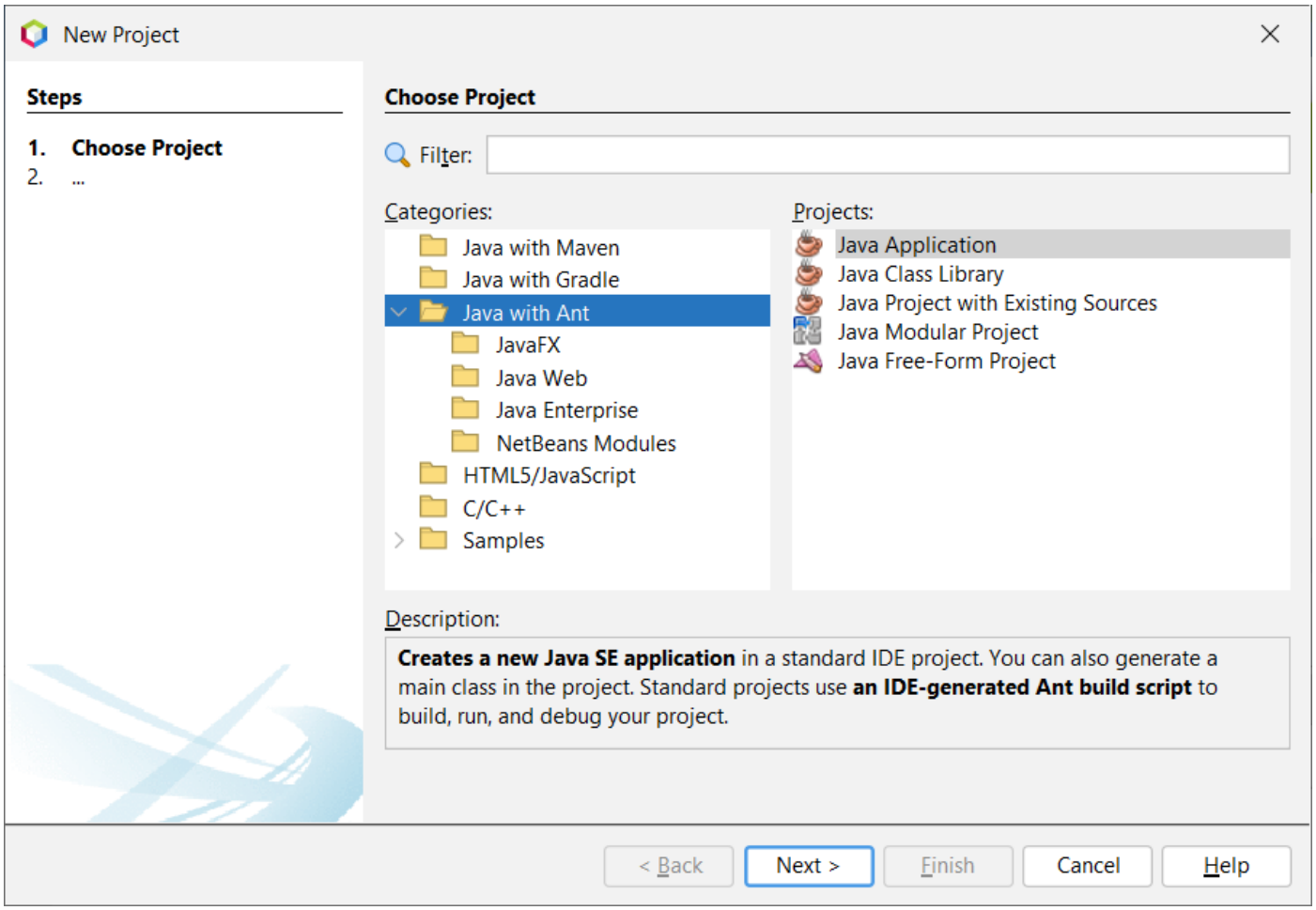


In the following window select Java with Ant and Java Application.

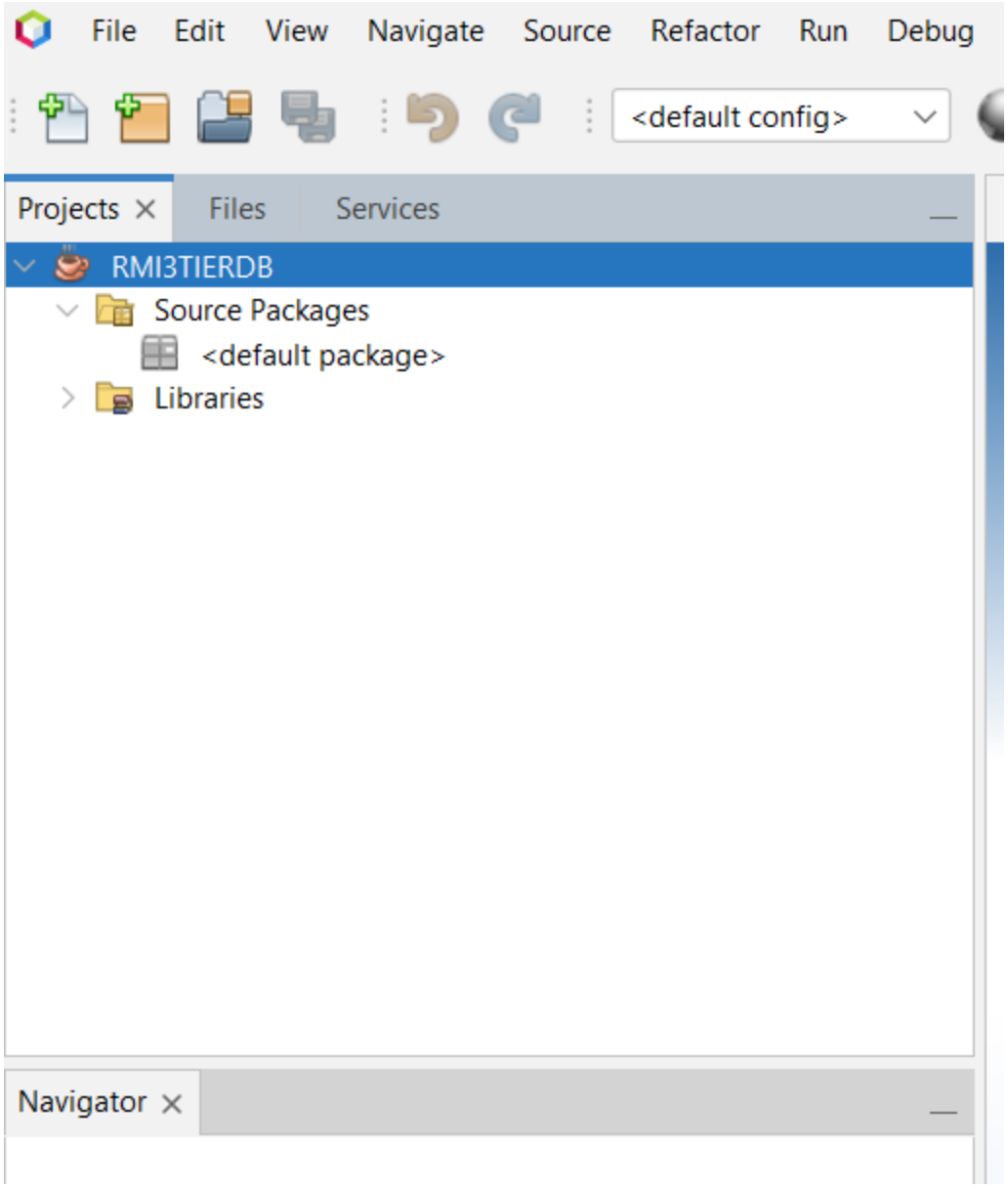


Then enter the name of the project and the respective project folder as follows:





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 {
```

```
        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", "admin");
    } 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("The customer with this account is: " + getCustomerId(3).getFirst());
        }

    public static ArrayList<Integer> getCustomerId(int idAccount){
      ArrayList<Integer> ids = new  ArrayList<Integer>();
                        try {
                                Statement s = conn.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;
    }
}
```

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 long getBalance()
      throws RemoteException;

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

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 testConn() 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 long        balance;
  private String      accountNumber;

  // public constructor
  public AccountImpl (
      BankManager bankManager,
      Client client,
      String accountNumber) {
    this.bankManager = bankManager;
    this.client      = client;
    this.balance     = 0;
    this.accountNumber = accountNumber;
  }

  public void deposit(long amount) {
```

```
    balance += amount;
  }

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

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

  @Override
  public long getBalance()
      throws RemoteException {
    return balance;
  }

  @Override
  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.rmi.RemoteException;
import java.rmi.registry.LocateRegistry;
```

```
import java.rmi.registry.Registry;

public class BankSystemServer {

    public static void main(String args[]) {
        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());
        } catch (RemoteException remoteException) {
            System.err.println(
                    "Failure during object export to RMI: "
                    + remoteException);
        }
        System.out.println("Server started.");
        System.out.println("Enter <CR> to end.");
        try {
            int i = System.in.read();
        } catch (Exception exception) {
        }
        System.exit(0);
    }
}
```

BankUser.java

```
import java.rmi.*;
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;


public class BankUser {

    public static void main(String[] args) {
        BankManager bm;
        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) {
            System.err.println("Not Bound: " + notBoundException);
        } catch (RemoteException remoteException) {
            System.err.println("Remote Exception: " + remoteException);
        }
    }
}
```

Step 2: Compile the code following these steps:

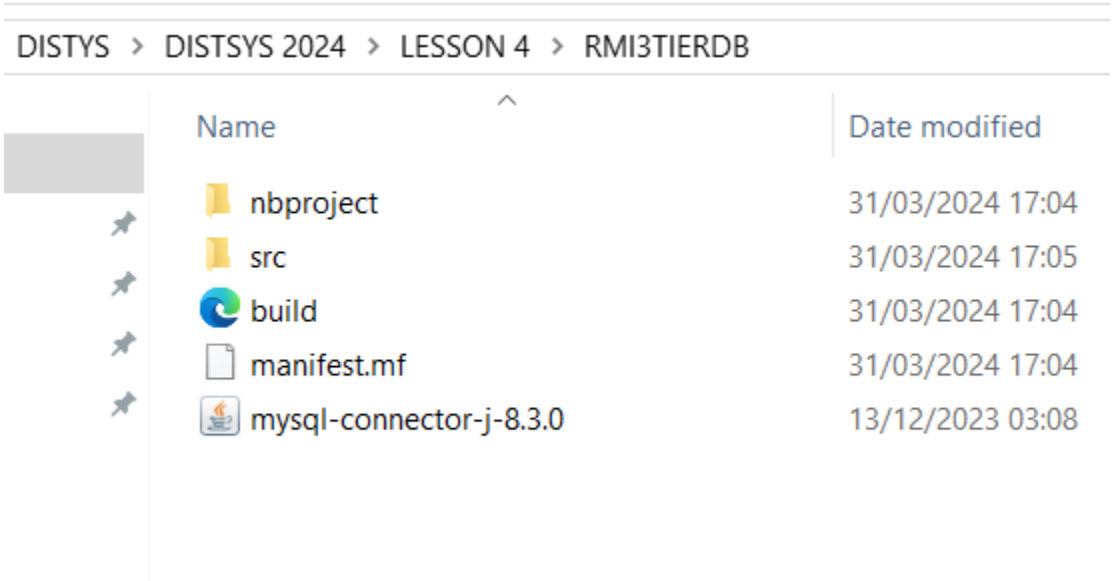
In order to connect Java with MySQL we need the following connector:

mysql-connector-j-8.3.0.jar

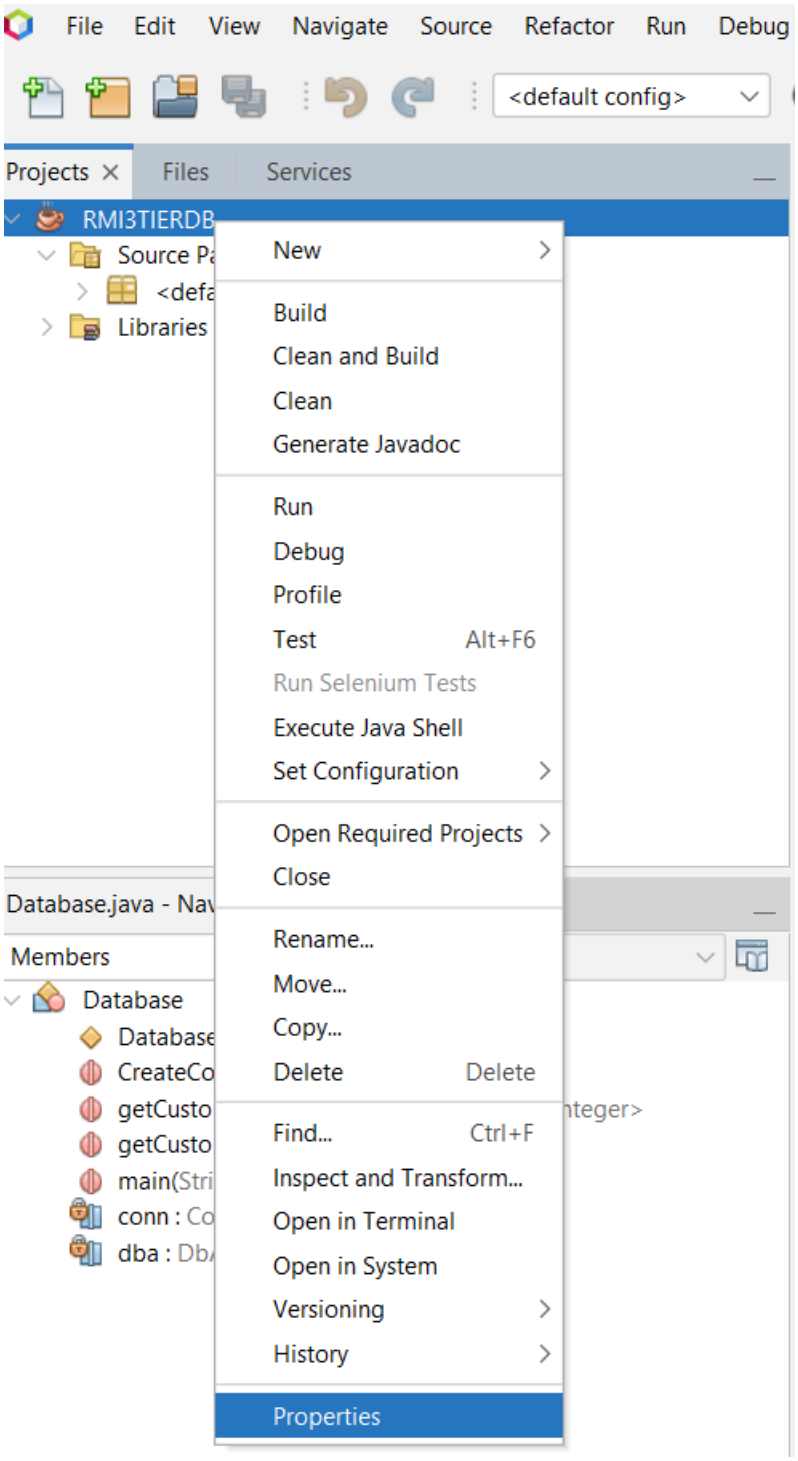
Download such connector from the following site:

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

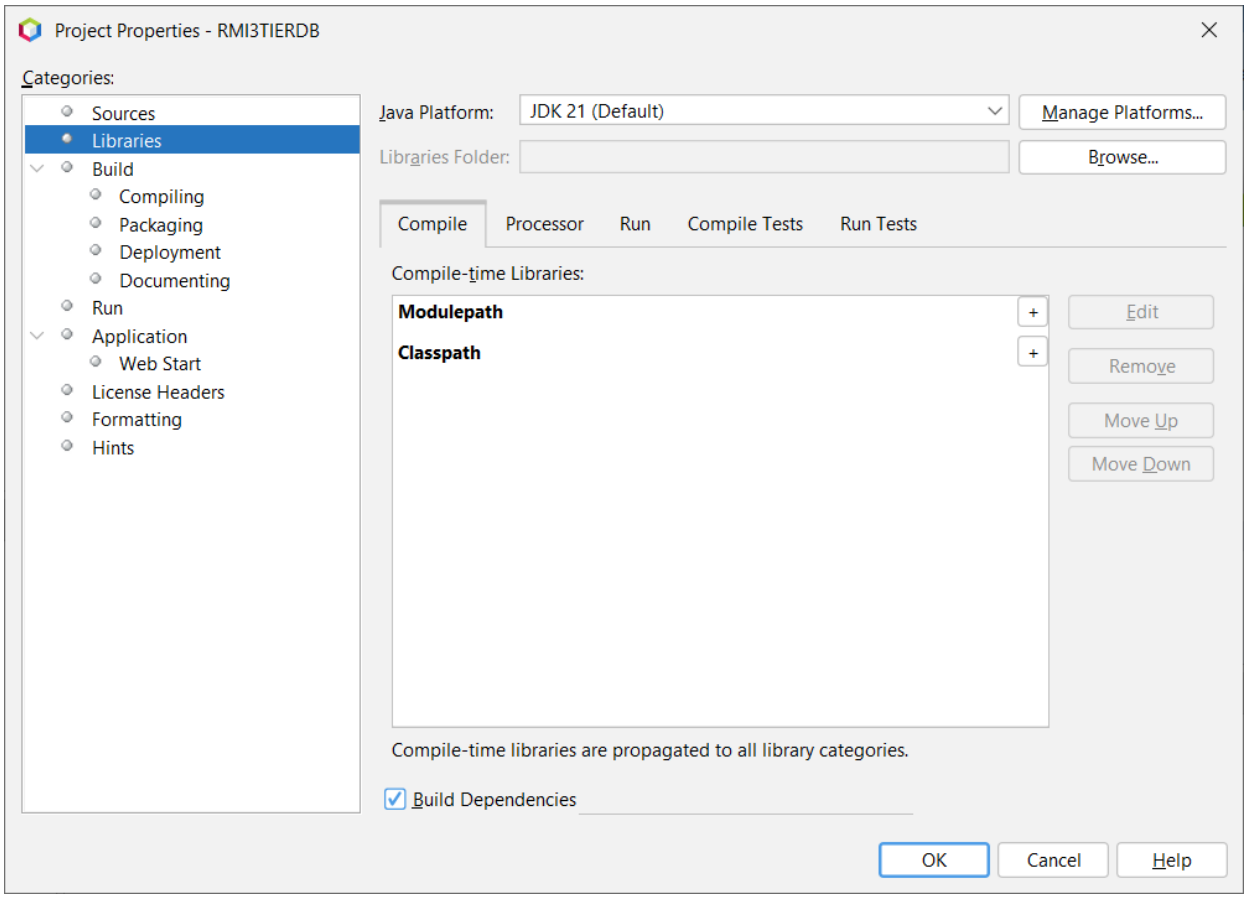
as platform independent file which is a zip file inside which you will find the file mysql-connector-j-8.3.0.jar. Copy such file in the main folder of the NetBeans project folder as follows:



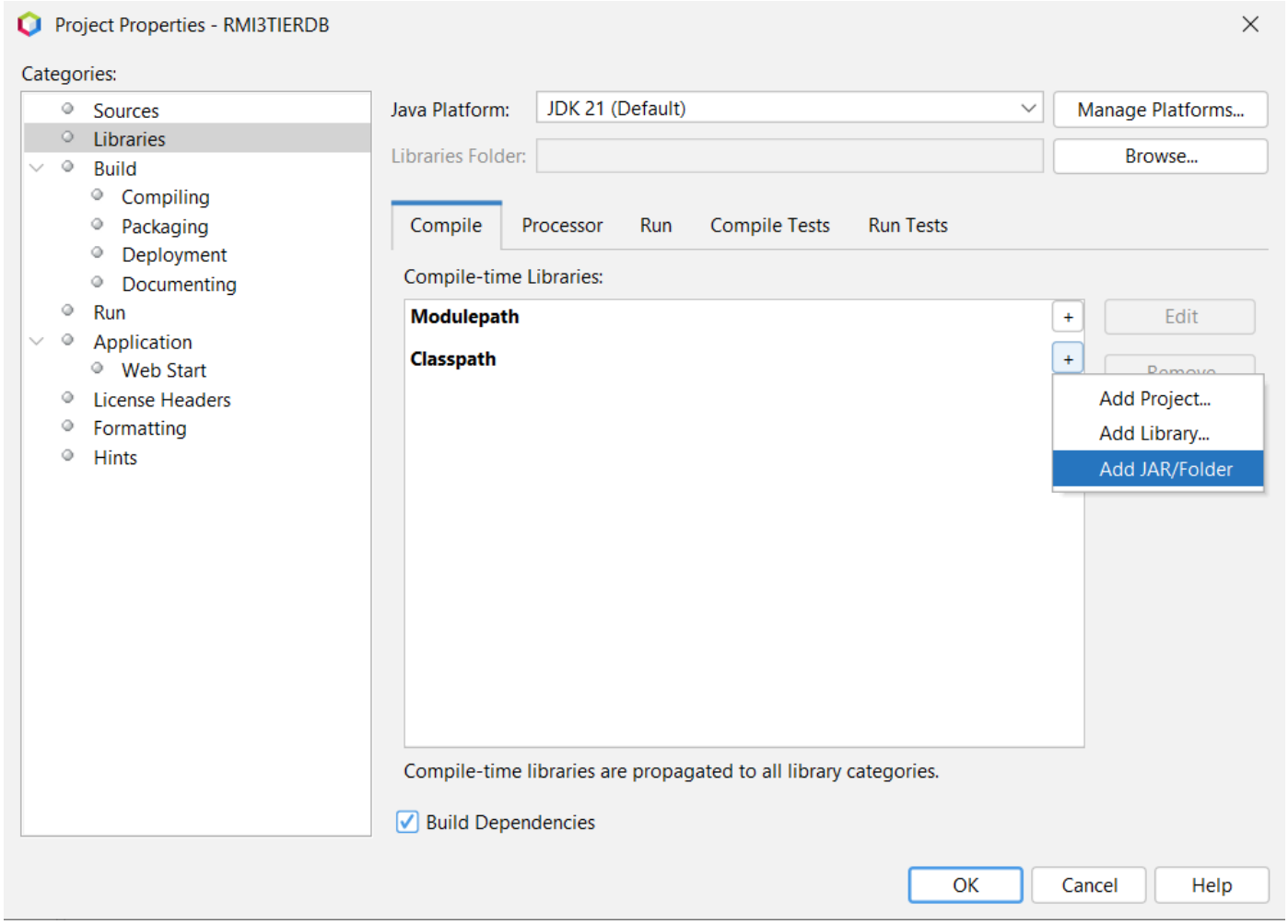
Add the connector to the libraries of the project in Netbeans as follows:



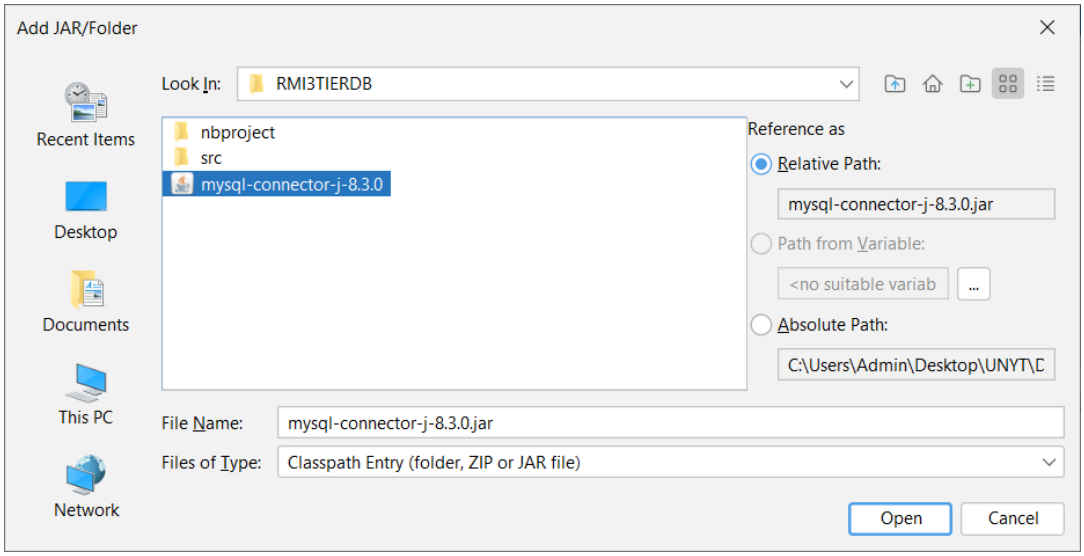
Click with right of the mouse on the Project. Select Properties and in the next window select Libraries:



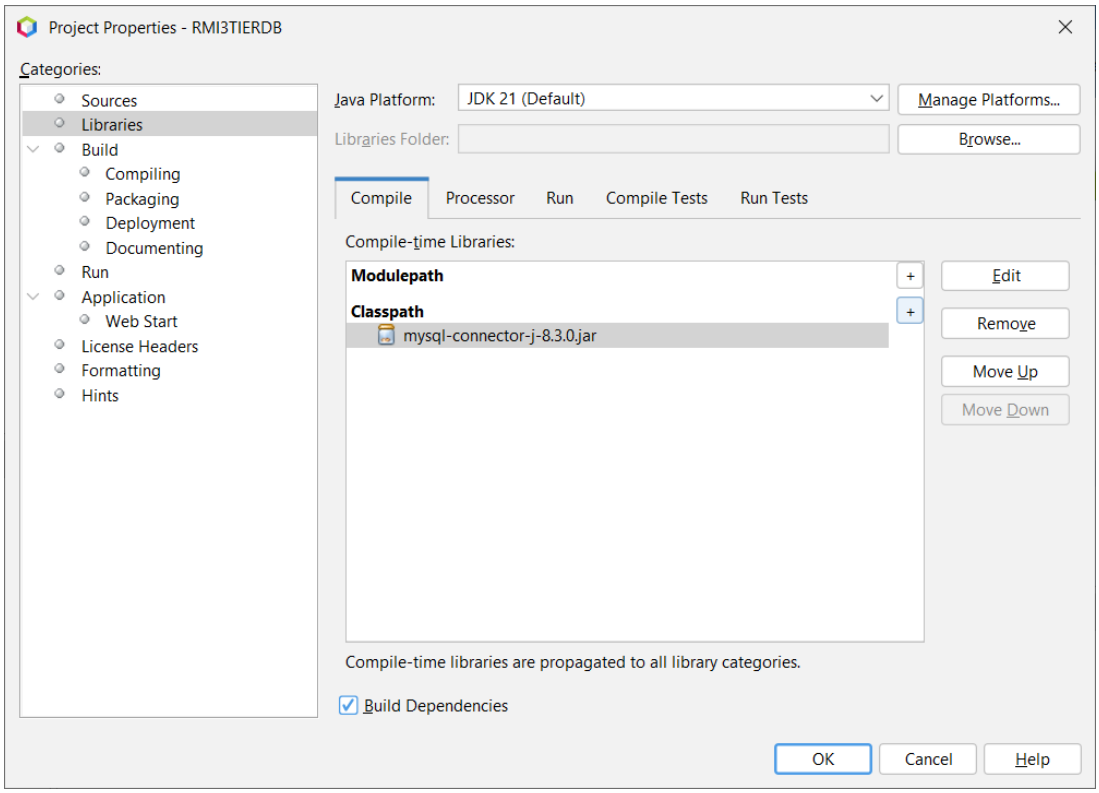
Click on the right + symbol of the Classpath item as follows, and select Add Jar/Folder:



Select relative path:



You should see the following window:

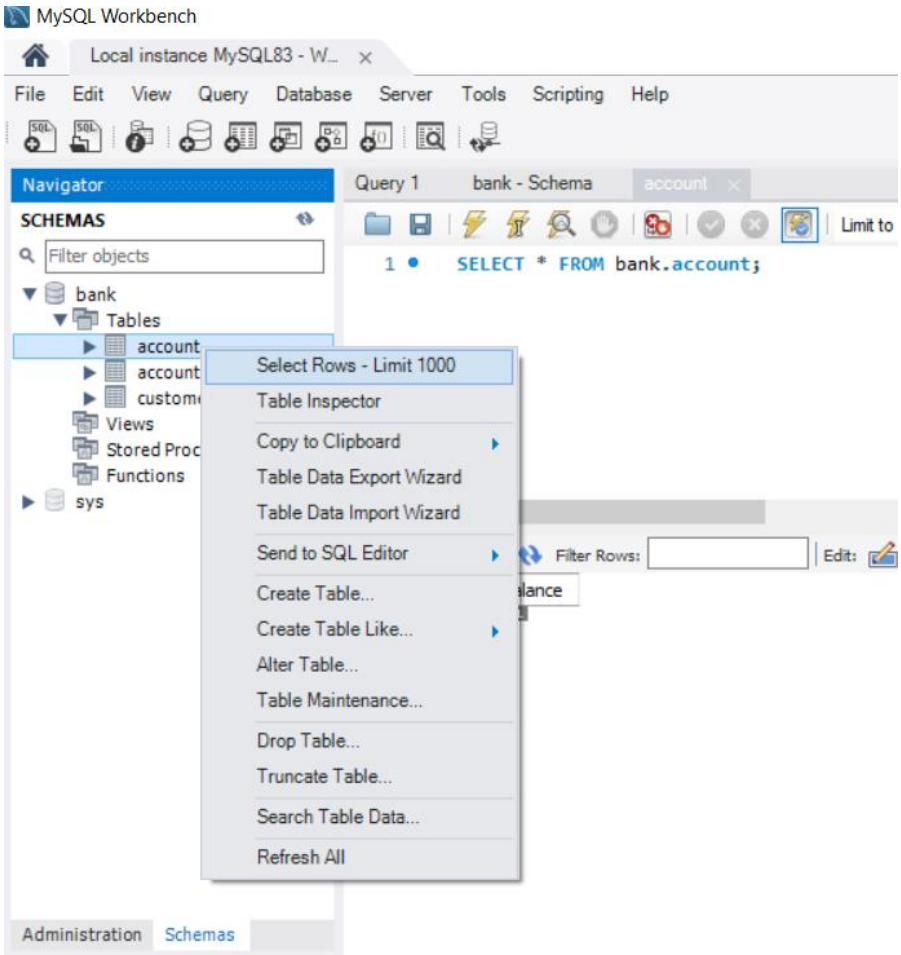


Test the connection with MySQL as follows by using the following classes:

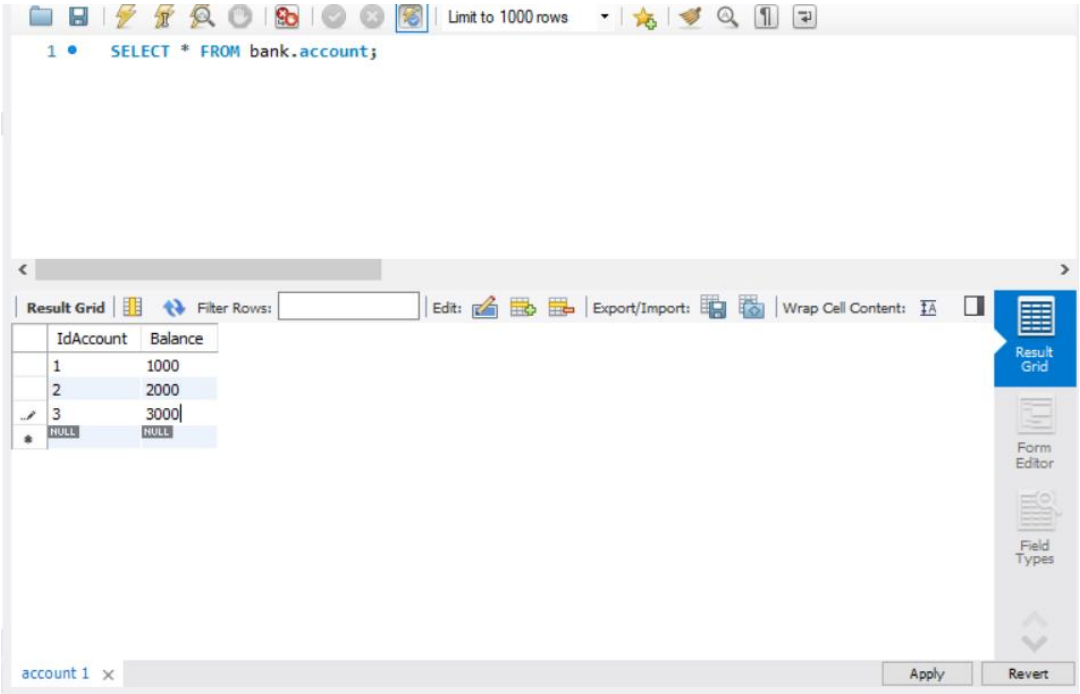
DbAccess.java and Database.java

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

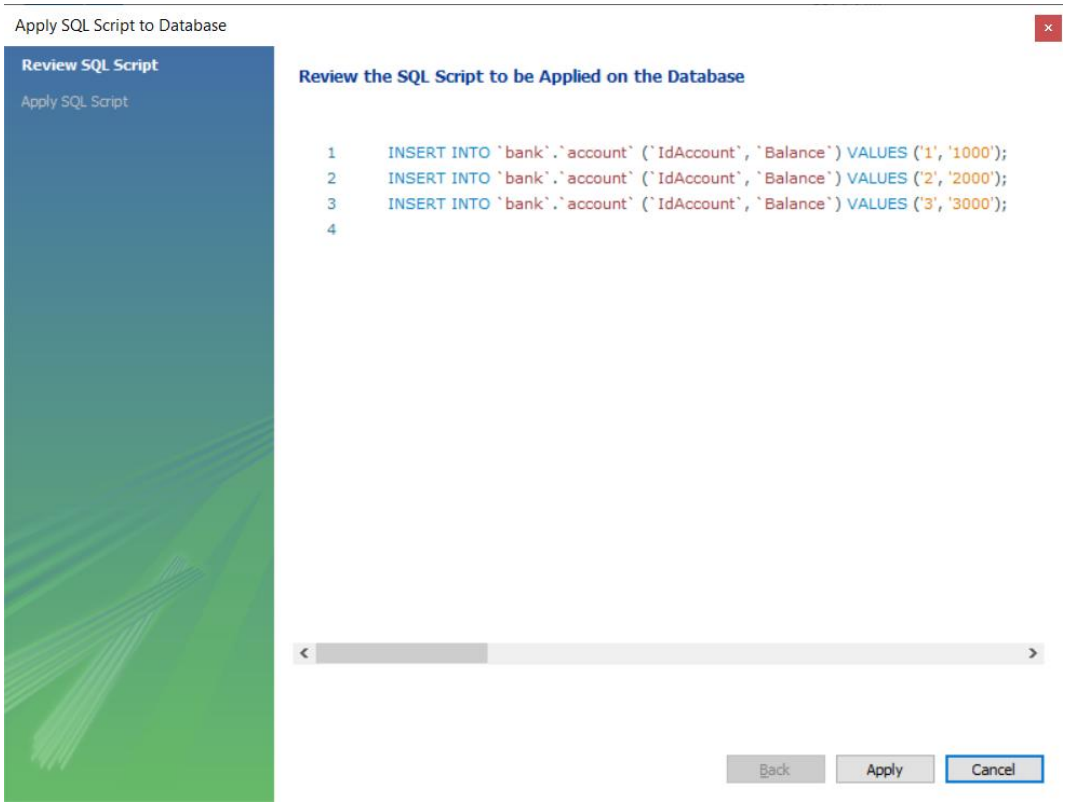
Click with the right button of the mous on the table account and then click on Select Rows as follows:



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



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



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

Table customer

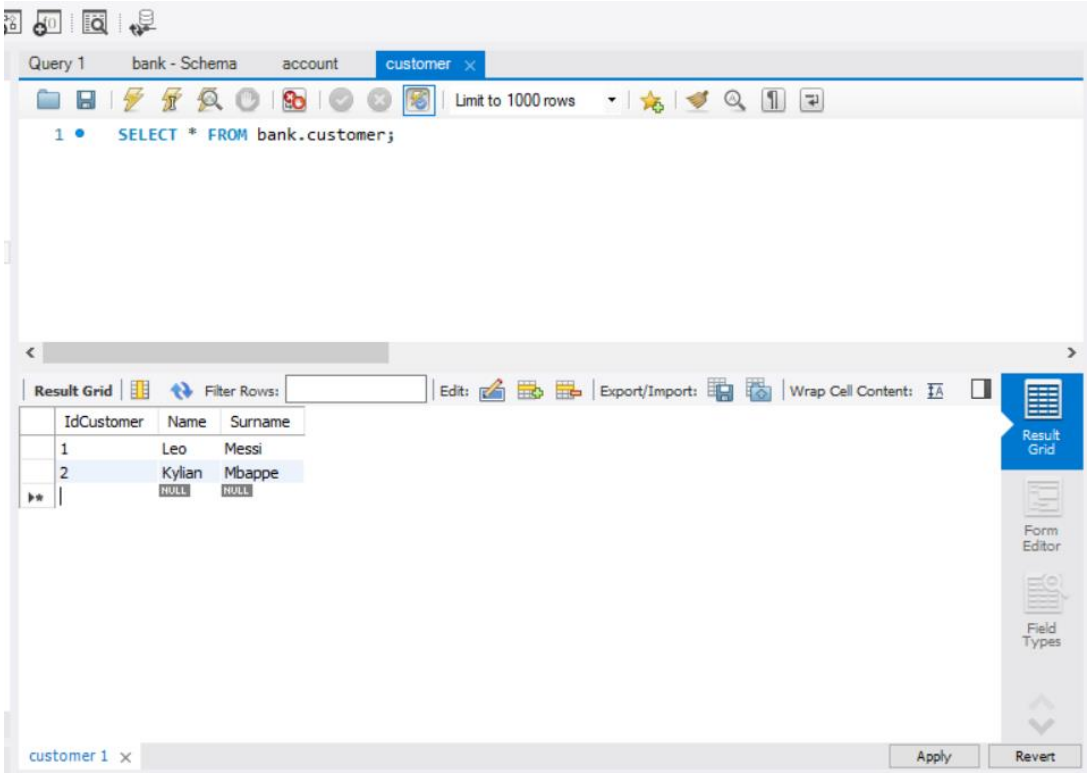
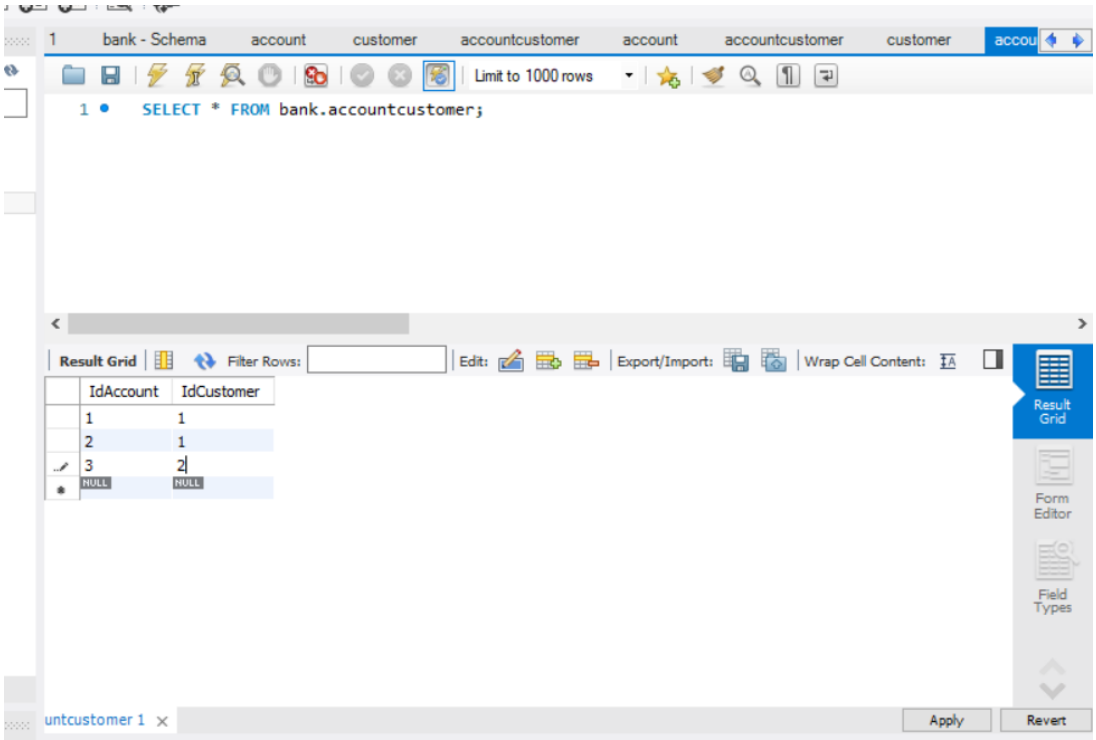


Table accountcustomer:



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:

```
Database > main >
Output
Debugger Console   RMI3TIERDB (run)
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. You will get an error due to the absence of the JDBC connector missing as follows:

```
Command Prompt
(c) Microsoft Corporation. All rights reserved.

C:\Users\Admin>cd C:\Users\Admin\Desktop\UNYT\DISTYS\DISTSYS 2024\LESSON 4\RMI3TIERDB\build\classes

C:\Users\Admin\Desktop\UNYT\DISTYS\DISTSYS 2024\LESSON 4\RMI3TIERDB\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 DbAccess.initializeConnection(DbAccess.java:13)
        at DbAccess.CreateConnection(DbAccess.java:28)
        at Database.CreateConnection(Database.java:12)
        at BankManagerImpl.testConn(BankManagerImpl.java:26)
        at BankManagerImpl.<init>(BankManagerImpl.java:9)
        at BankSystemServer.main(BankSystemServer.java:11)
java.lang.NullPointerException: Cannot invoke "java.sql.Connection.createStatement()" because "Database.conn" is null
        at Database.getCustomerId(Database.java:25)
        at BankManagerImpl.testConn(BankManagerImpl.java:27)
        at BankManagerImpl.<init>(BankManagerImpl.java:9)
        at BankSystemServer.main(BankSystemServer.java:11)
Exception in thread "main" java.lang.IndexOutOfBoundsException: Index 0 out of bounds for length 0
        at java.base/jdk.internal.util.Preconditions.outOfBounds(Preconditions.java:100)
        at java.base/jdk.internal.util.Preconditions.outOfBoundsCheckIndex(Preconditions.java:106)
        at java.base/jdk.internal.util.Preconditions.checkIndex(Preconditions.java:302)
        at java.base/java.util.Objects.checkIndex(Objects.java:385)
        at java.base/java.util.ArrayList.get(ArrayList.java:427)
        at BankManagerImpl.testConn(BankManagerImpl.java:27)
        at BankManagerImpl.<init>(BankManagerImpl.java:9)
        at BankSystemServer.main(BankSystemServer.java:11)

C:\Users\Admin\Desktop\UNYT\DISTYS\DISTSYS 2024\LESSON 4\RMI3TIERDB\build\classes>
```

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 4 > RMI3TIERDB > build > classes

Name	Date modified	Type	Size
Account.class	01/04/2024 14:21	CLASS File	1 KB
AccountImpl.class	01/04/2024 14:21	CLASS File	2 KB
BankManager.class	01/04/2024 14:21	CLASS File	1 KB
BankManagerImpl.class	01/04/2024 14:21	CLASS File	1 KB
BankSystemServer.class	01/04/2024 14:21	CLASS File	2 KB
BankUser.class	01/04/2024 14:21	CLASS File	2 KB
Client.class	01/04/2024 14:21	CLASS File	1 KB
ClientImpl.class	01/04/2024 14:21	CLASS File	1 KB
Database.class	01/04/2024 14:21	CLASS File	3 KB
DbAccess.class	01/04/2024 14:21	CLASS File	3 KB
NoCashAvailableException.class	01/04/2024 14:21	CLASS File	1 KB
mysql-connector-j-8.3.0	13/12/2023 03:08	Executable Jar File	2,438 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
(c) Microsoft Corporation. All rights reserved.

C:\Users\Admin>cd C:\Users\Admin\Desktop\UNYT\DISTYS\DISTSYS 2024\LESSON 4\RMI3TIERDB\build\classes

C:\Users\Admin\Desktop\UNYT\DISTYS\DISTSYS 2024\LESSON 4\RMI3TIERDB\build\classes>java -cp ".;mysql-connector-j-8.3.0.jar" BankSystemServer
Server started.
Enter <CR> to end.
```

Do not close this window!

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

```
Command Prompt
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 4\RMI3TIERDB\build\classes

C:\Users\Admin\Desktop\UNYT\DISTYS\DISTSYS 2024\LESSON 4\RMI3TIERDB\build\classes>java -cp ".;mysql-connector-j-8.3.0.jar" BankUser
Testing connection with ID oustomer for IDAccount = 1:  1

C:\Users\Admin\Desktop\UNYT\DISTYS\DISTSYS 2024\LESSON 4\RMI3TIERDB\build\classes>
```

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:

```
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) {
```

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

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: