Lesson 01 Demo 01

Configuring JDBC and Creating Connection to MySQL DB

Objective: To configure JDBC and create a connection to a MySQL DB by creating a Java project and executing the **createConnection()** method for database operations

Tools required: Eclipse IDE

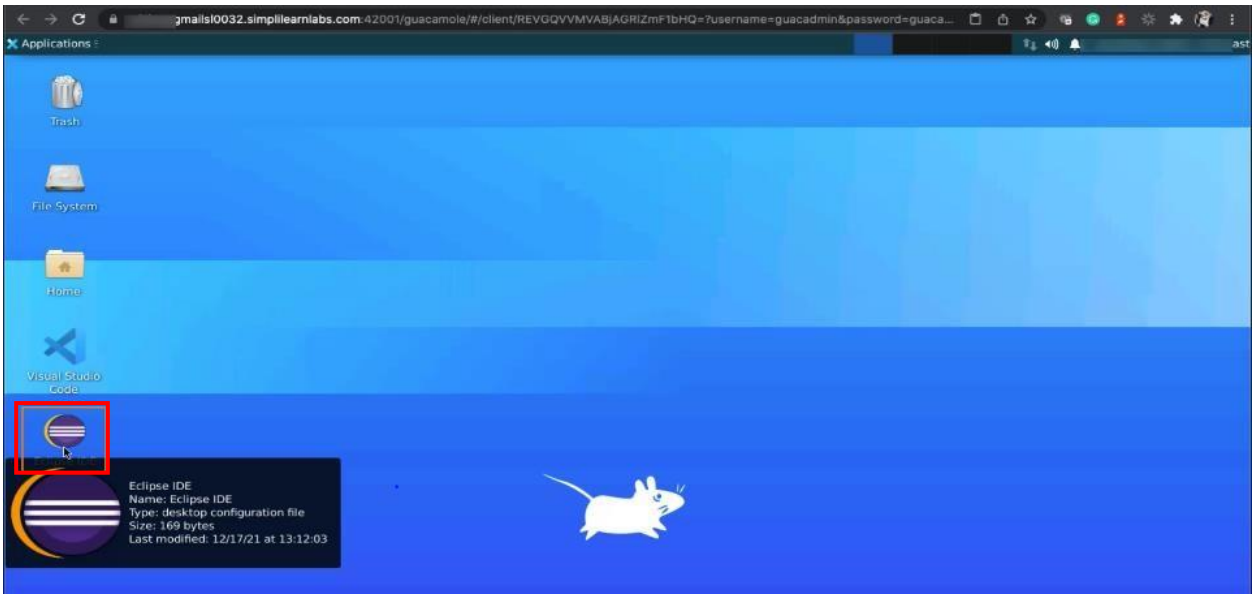
Prerequisites: None

Steps to be followed:

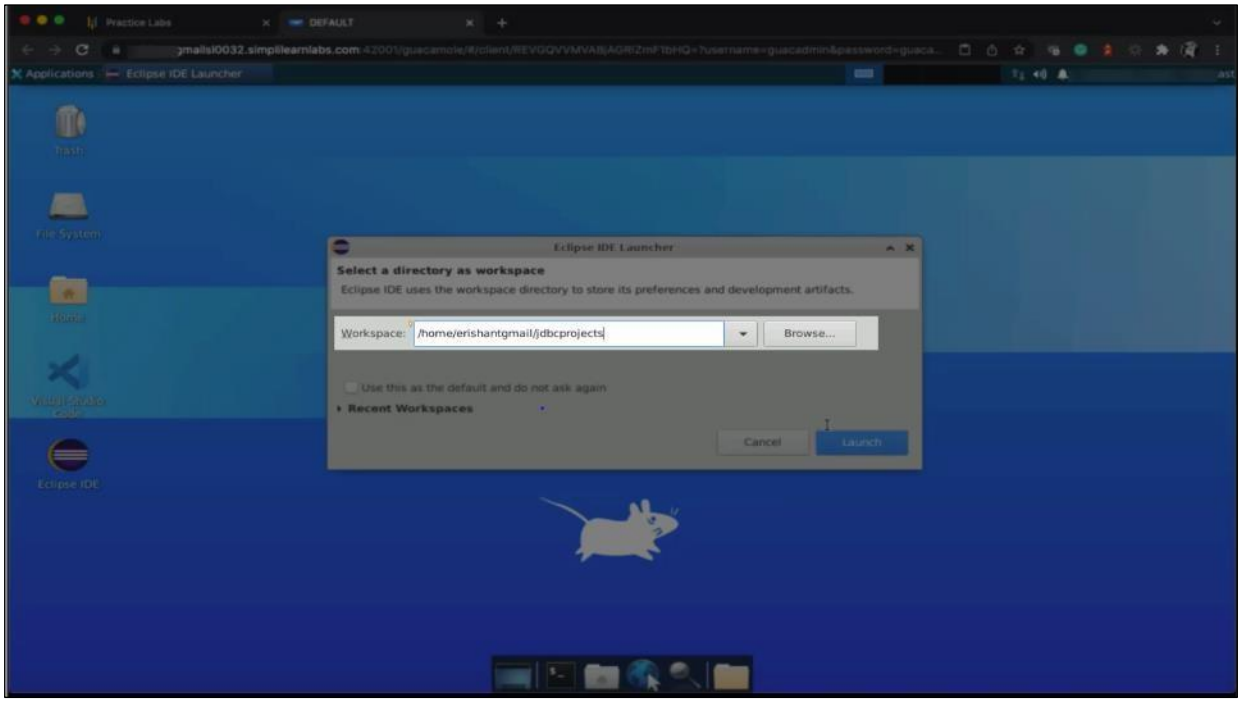
1. Create a Java project
2. Download the MySQL Connector/J library
3. Create a Maven project
4. Create an interface in com.example.cms
5. Create the DB class
6. Create a connection in the DAO.java file
7. Initialize the connection
8. Execute the createConnection method

Step 1: Create a Java project

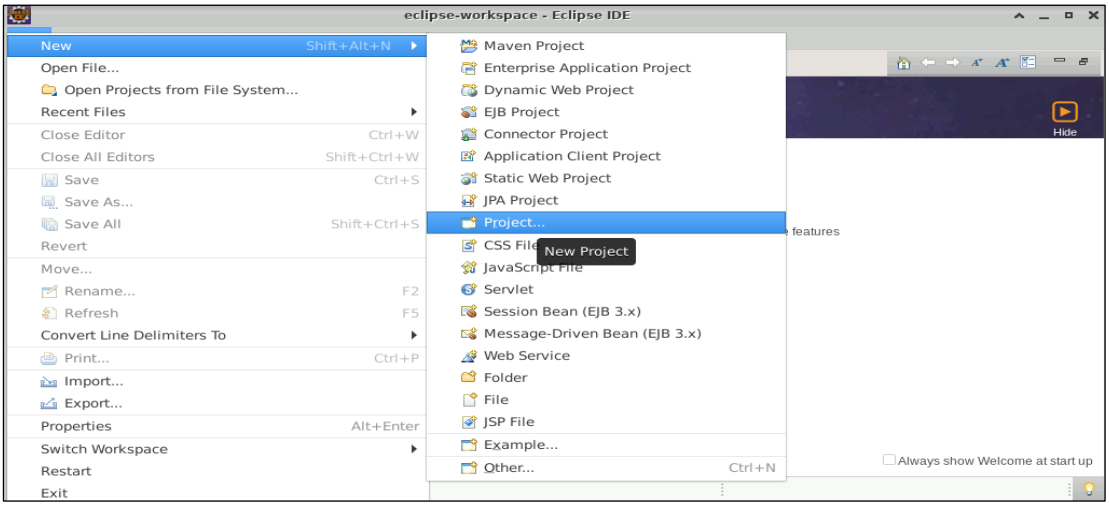
1.1 Open **Eclipse IDE** to create a Java project



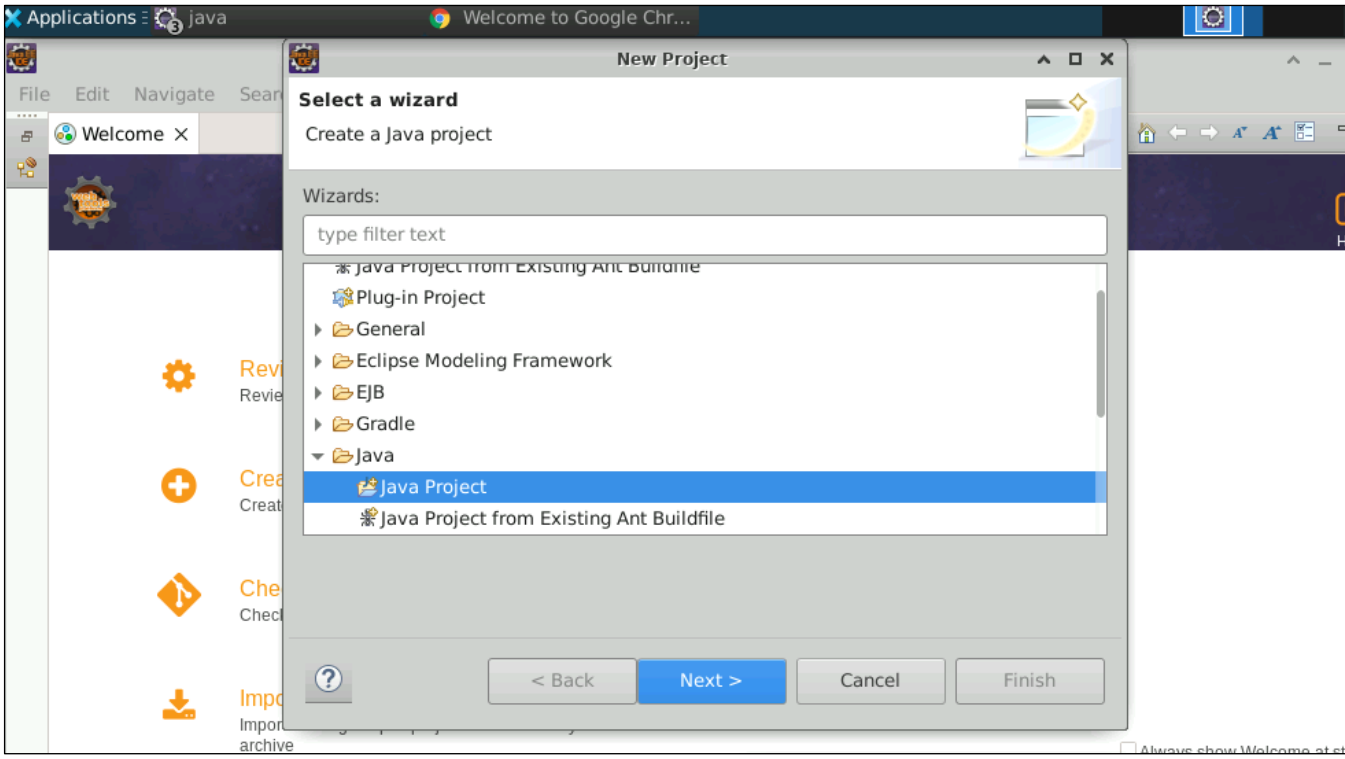
1.2 Select the default workspace and click on **Launch** to open Eclipse IDE



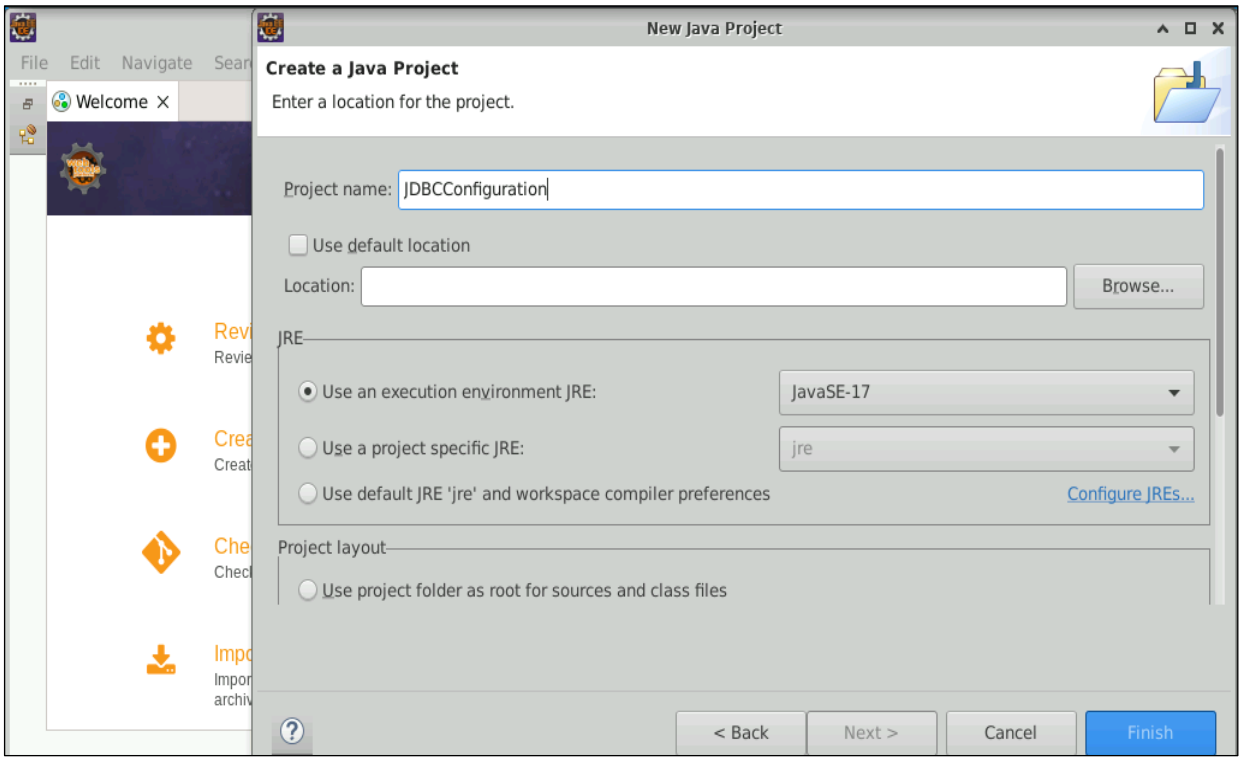
1.3 Create a new Java project, navigate to **File** from the toolbar **> New > Project**



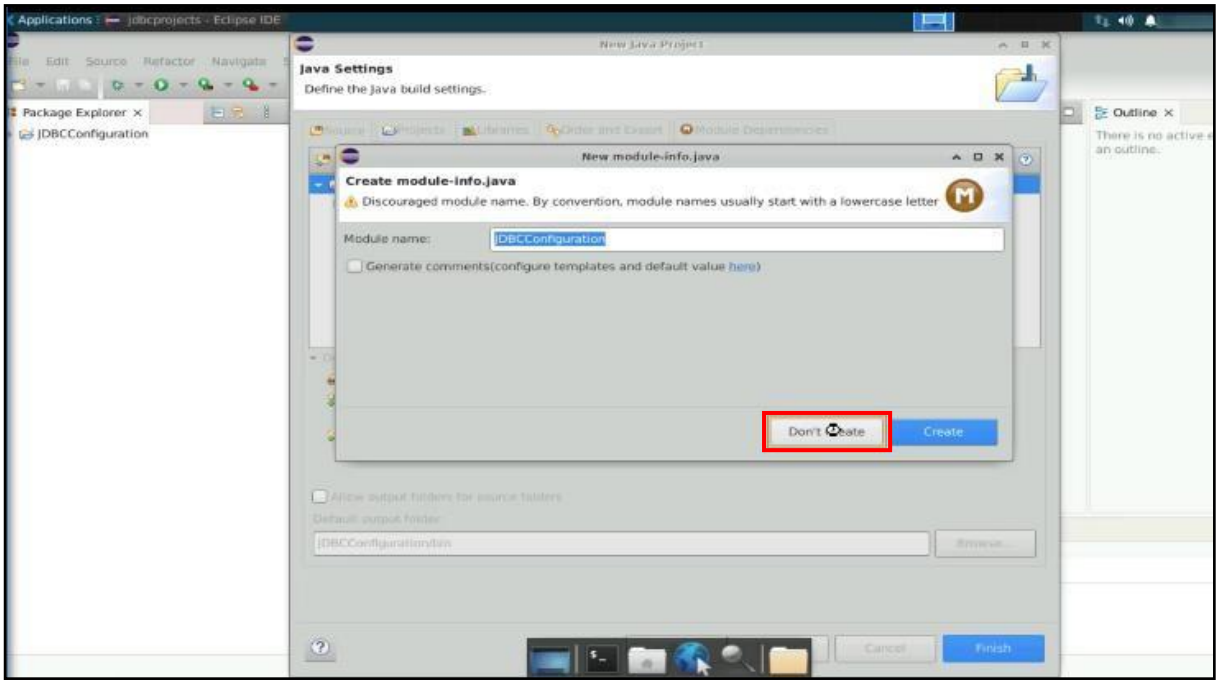
1.4 Select **Java Project** and click on **Next**



1.5 Name the project **JDBCConfiguration** and click on Finish

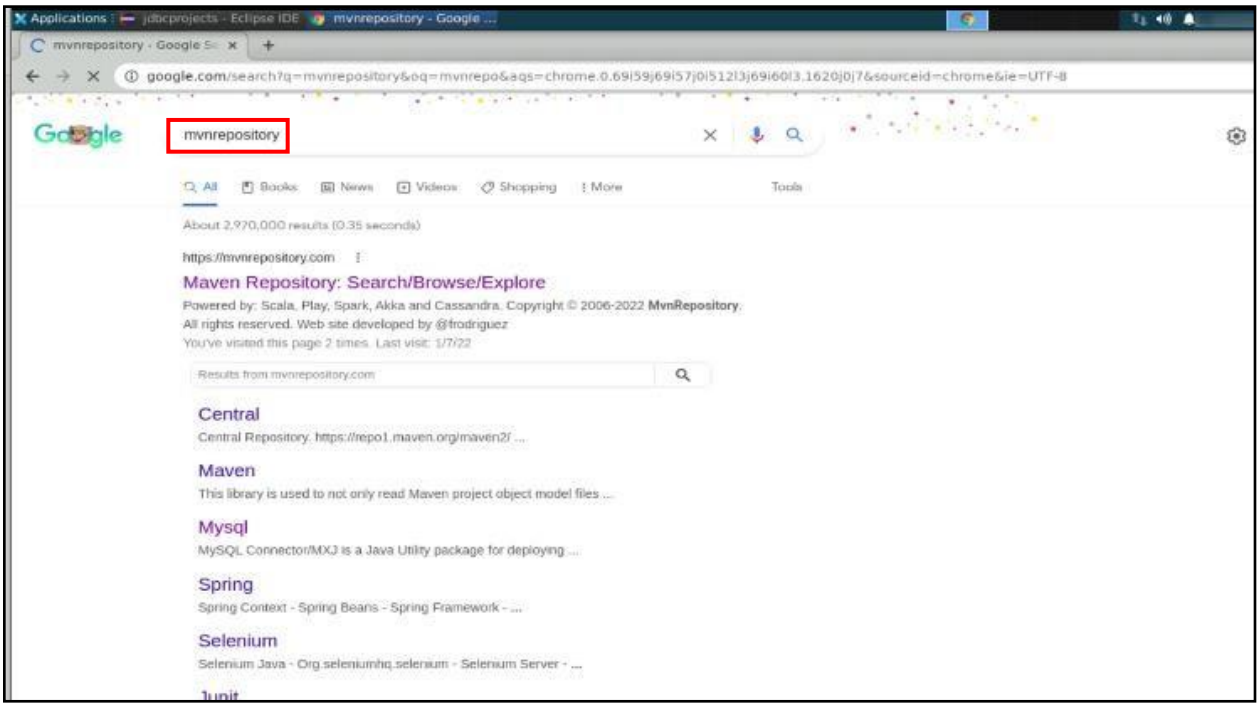


1.6 Click on the **Don't Create** option as there is no need to create a module, as shown in the screenshot below:

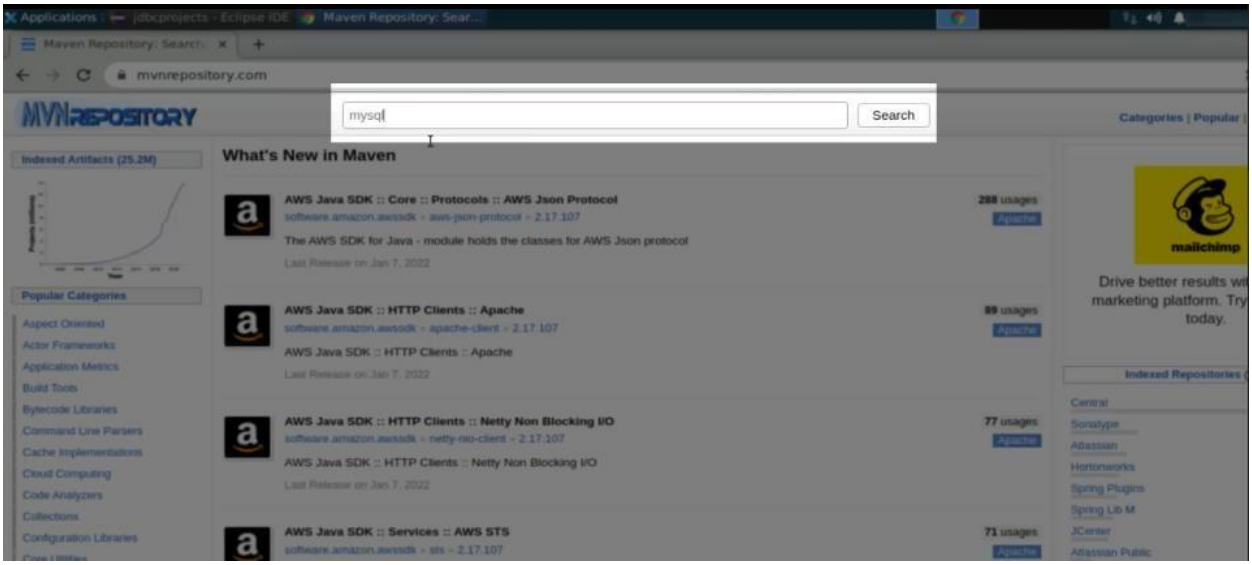


Step 2: Download the MySQL Connector/J library

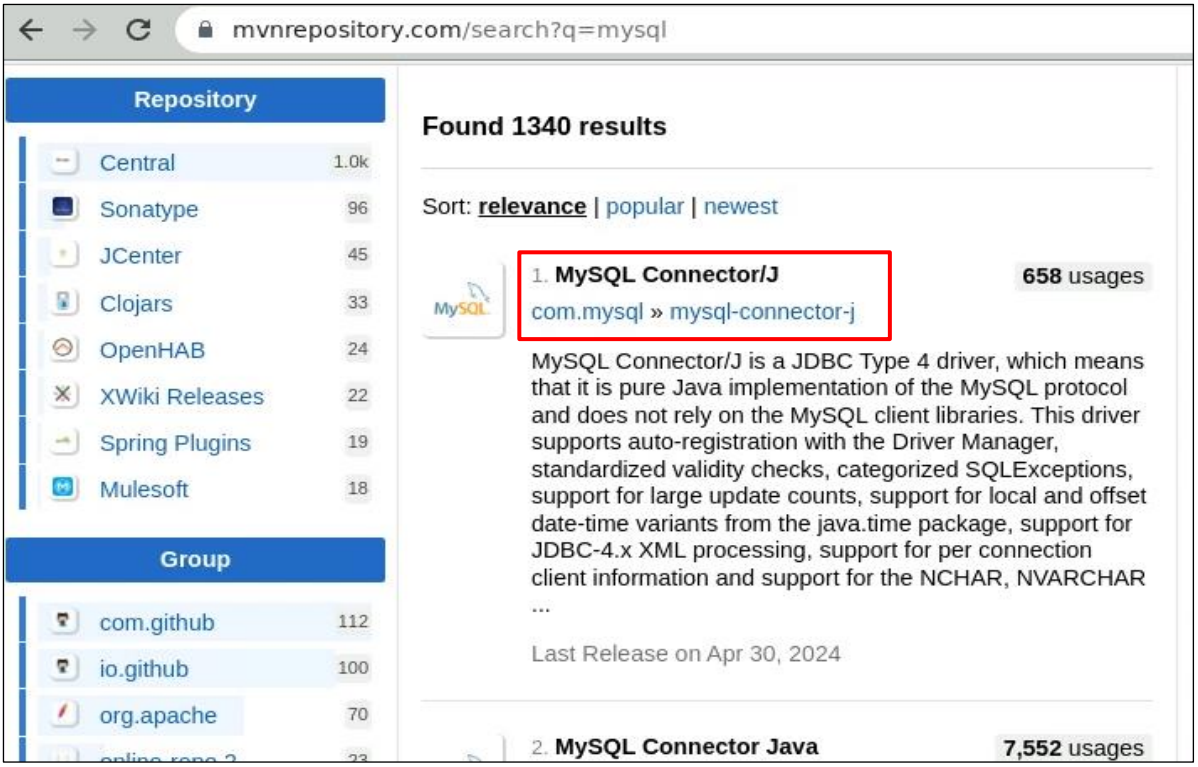
2.1 Open the preferred browser, search for **mvnrepository,** and click the first link of the Maven Repository



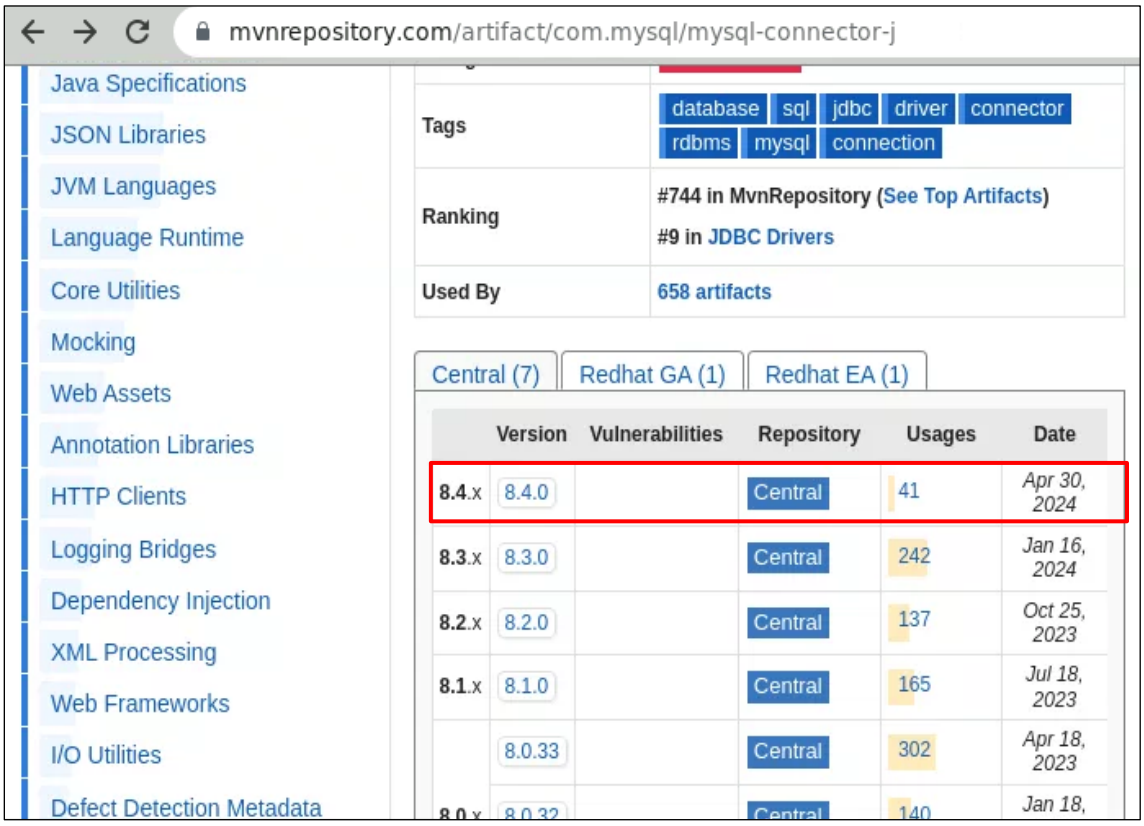
2.2 Search for **mysql** in the search bar



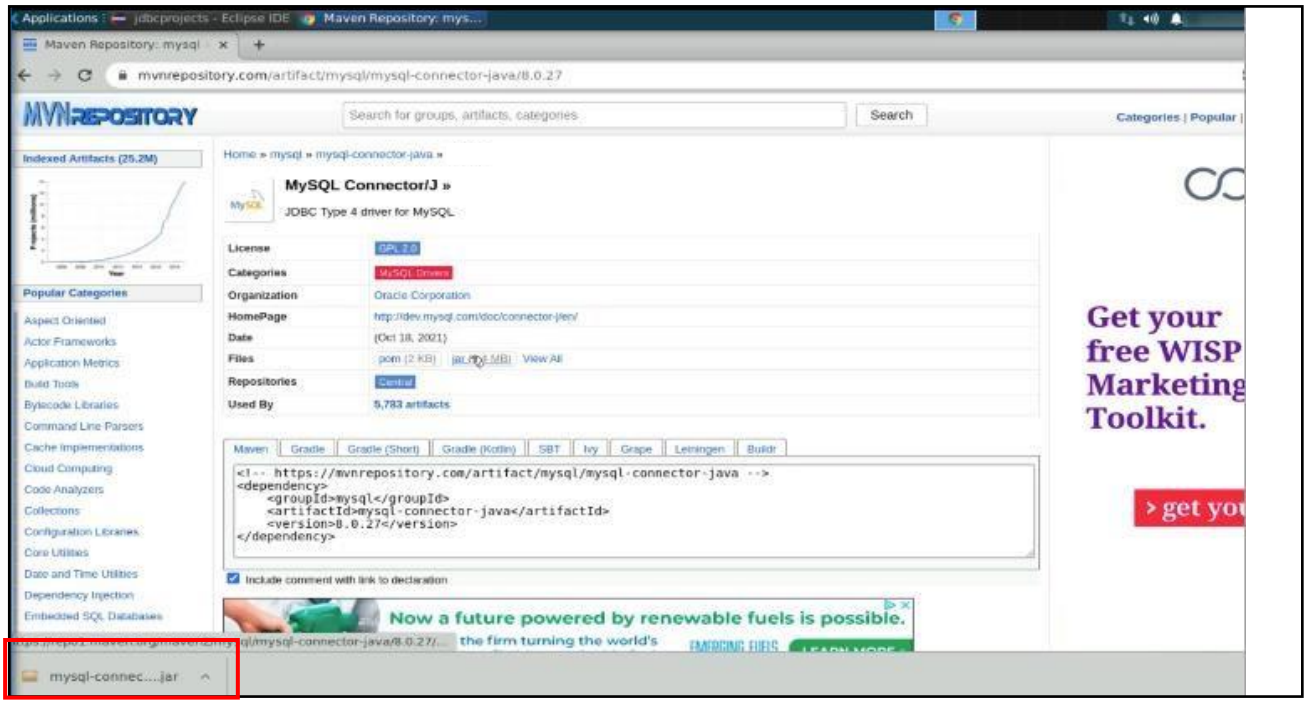
2.3 Select **MySQL Connector/J** as shown in the screenshot below:



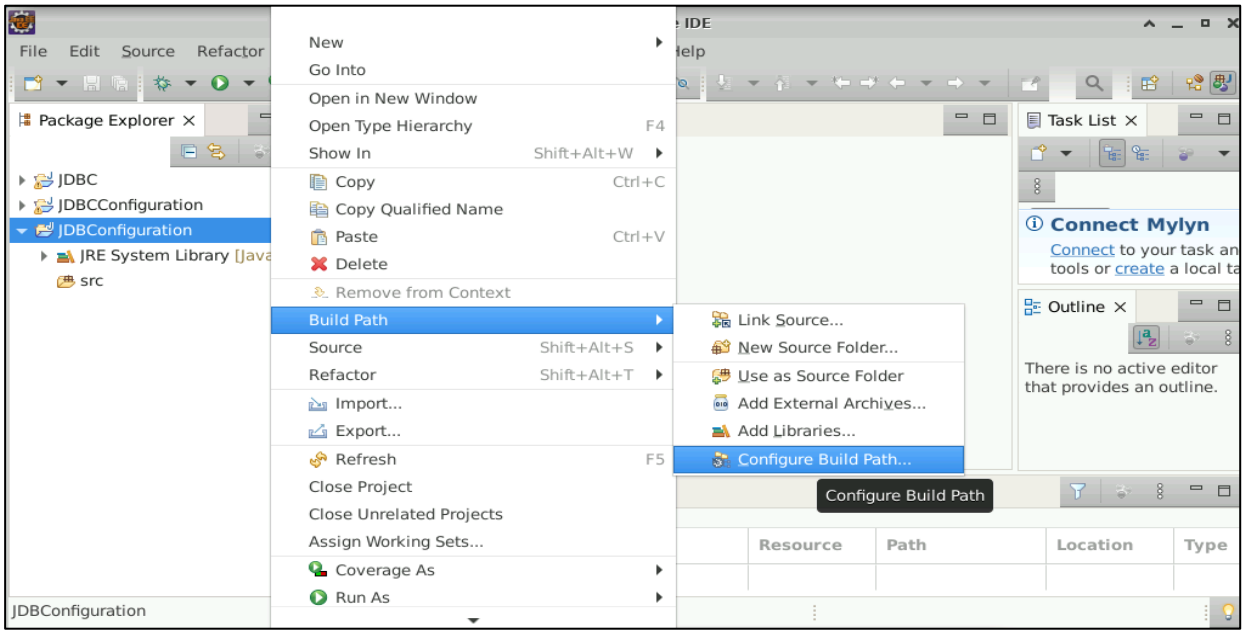
2.4 Check for the latest connector version, for example **8.4.0** as shown in the screenshot below:



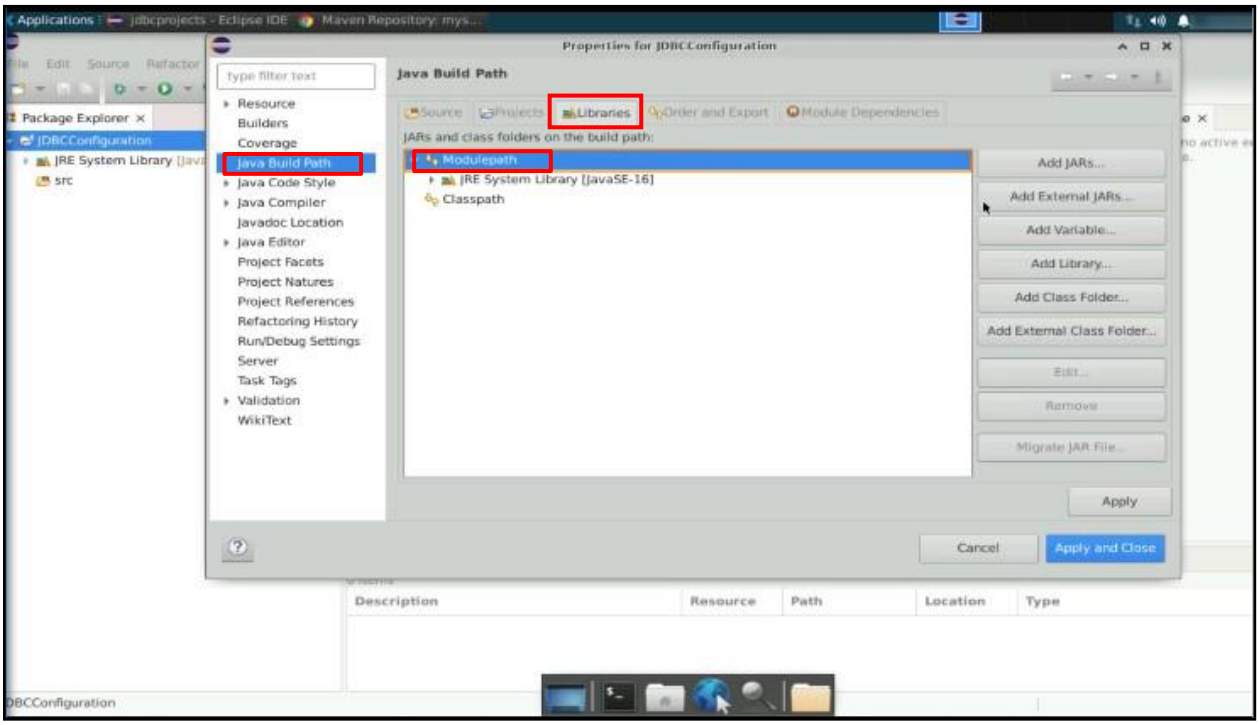
2.5 Click on the latest connector version, then click on JAR file under the files section to download it, as shown in the screenshot below:



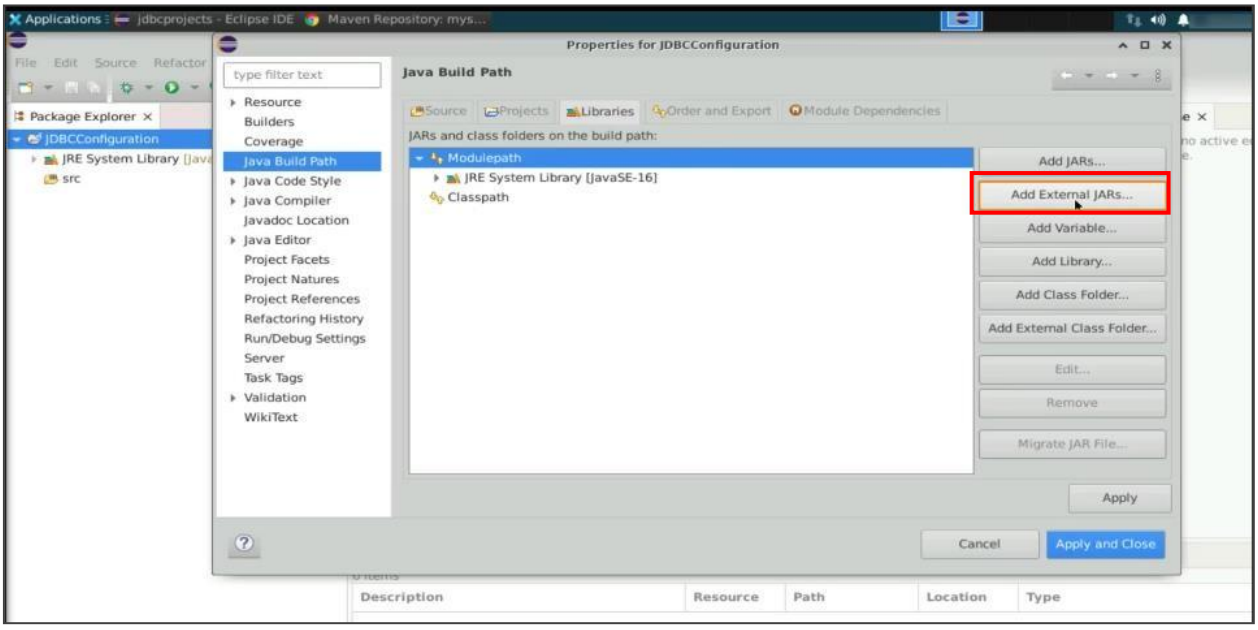
2.6 Right-click on the project file created, select **Build Path,** and choose **Configure Build Path** as shown in the screenshot below:



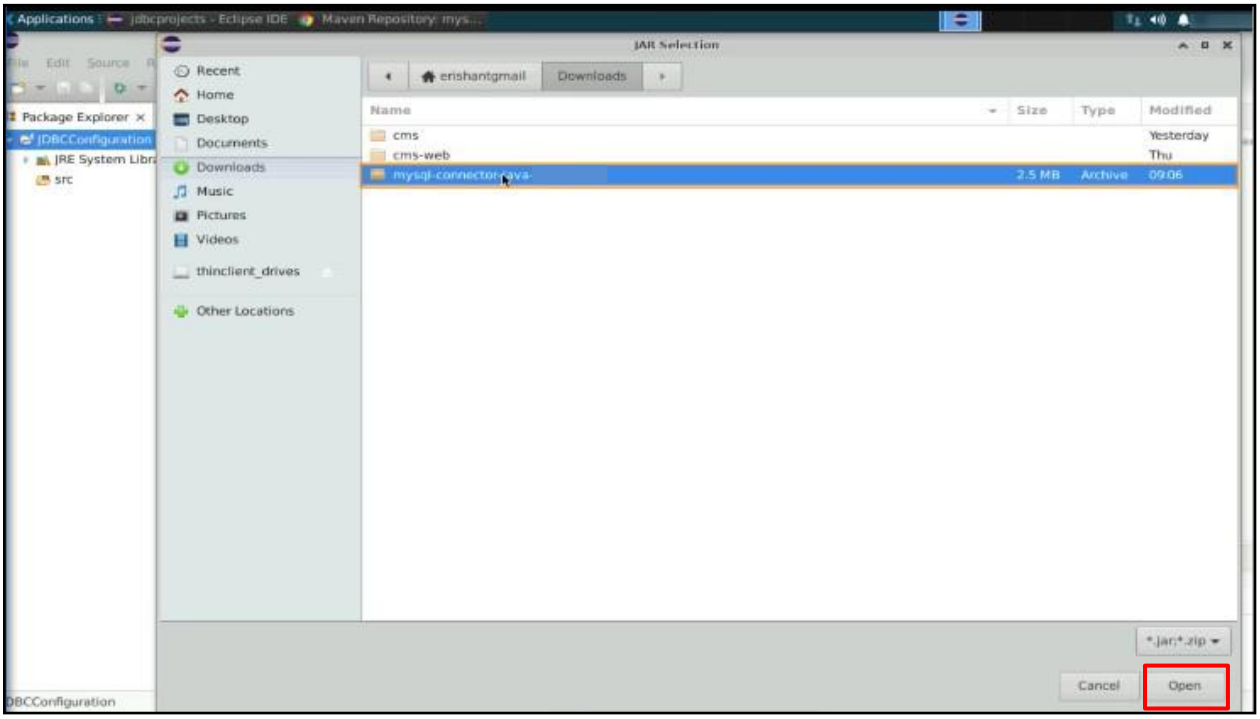
2.7 Go to **Java Build Path > Libraries tab > Modulepath** in the project properties window as shown in the screenshot below:



2.8 Click on the **Add External JARs** button and navigate to the location where the MySQL Connector/J JAR file is saved as shown in the screenshot below:

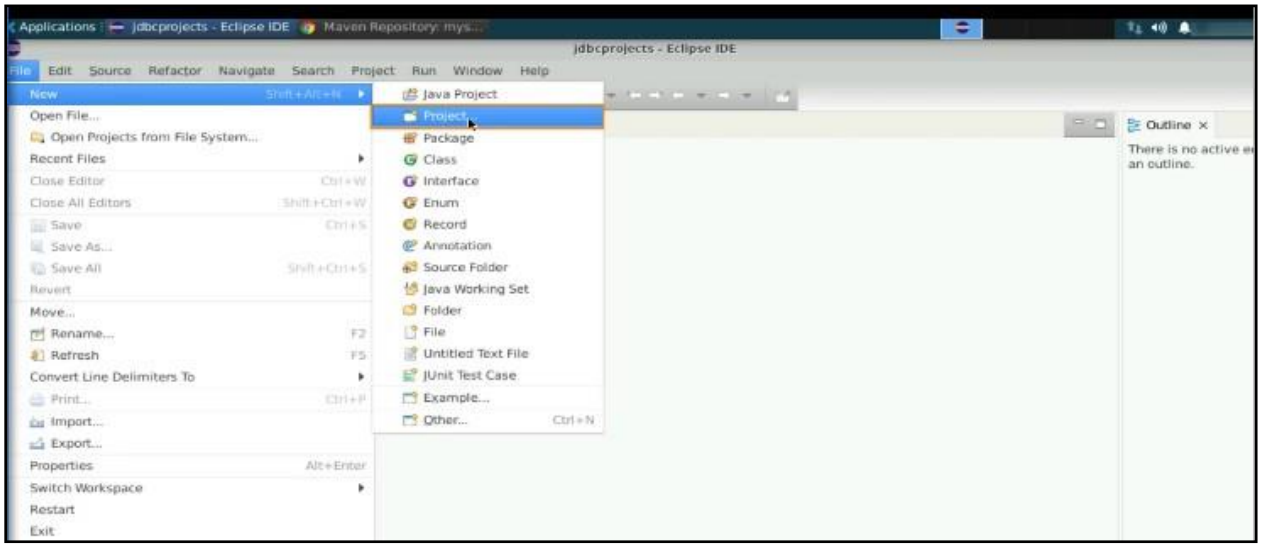


2.9 Select the **mysql-connector-java-8.4.0.jar** file and click **Open** as shown in the screenshot below:

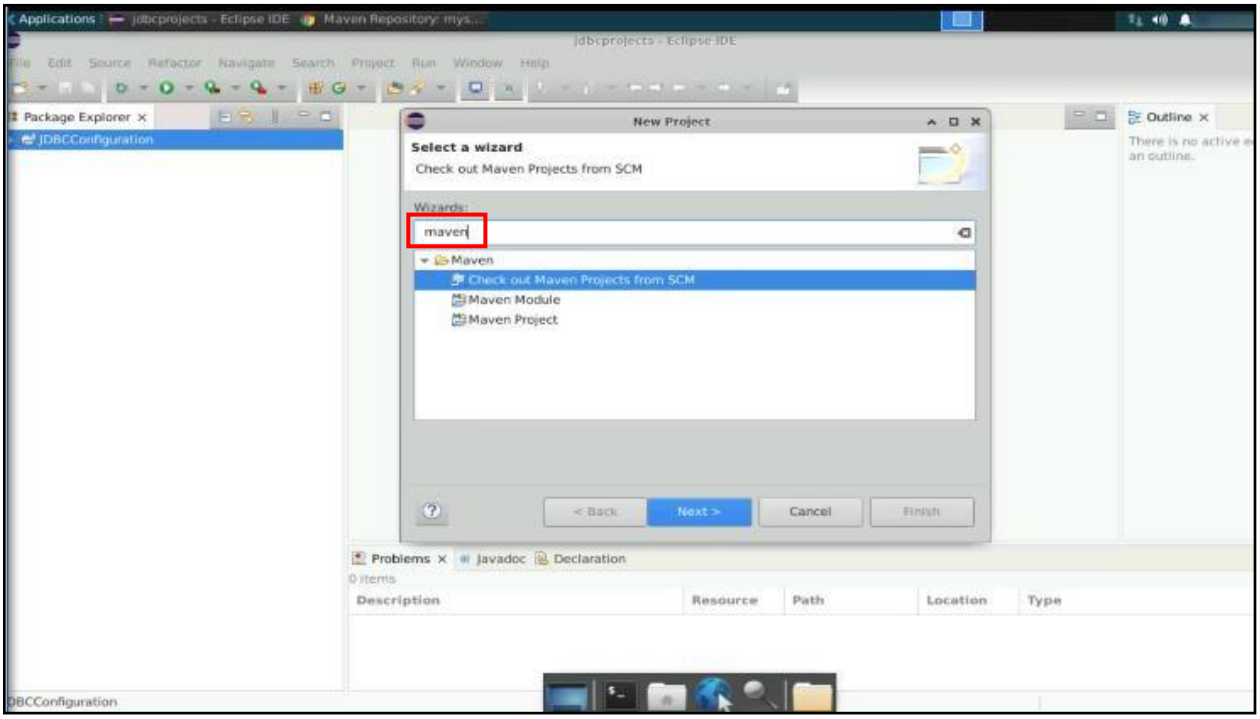


Step 3: Create a Maven project

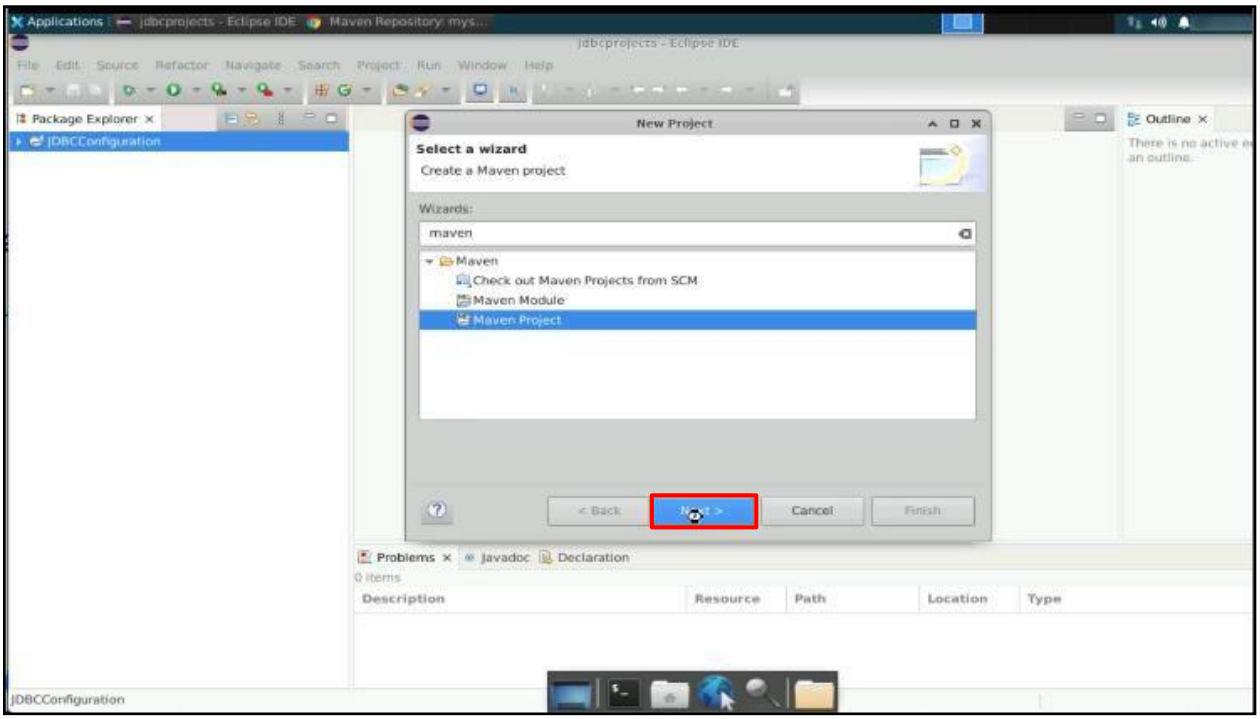
3.1 Create a new project by navigating to **Files > New > Project** as shown in the screenshot below:



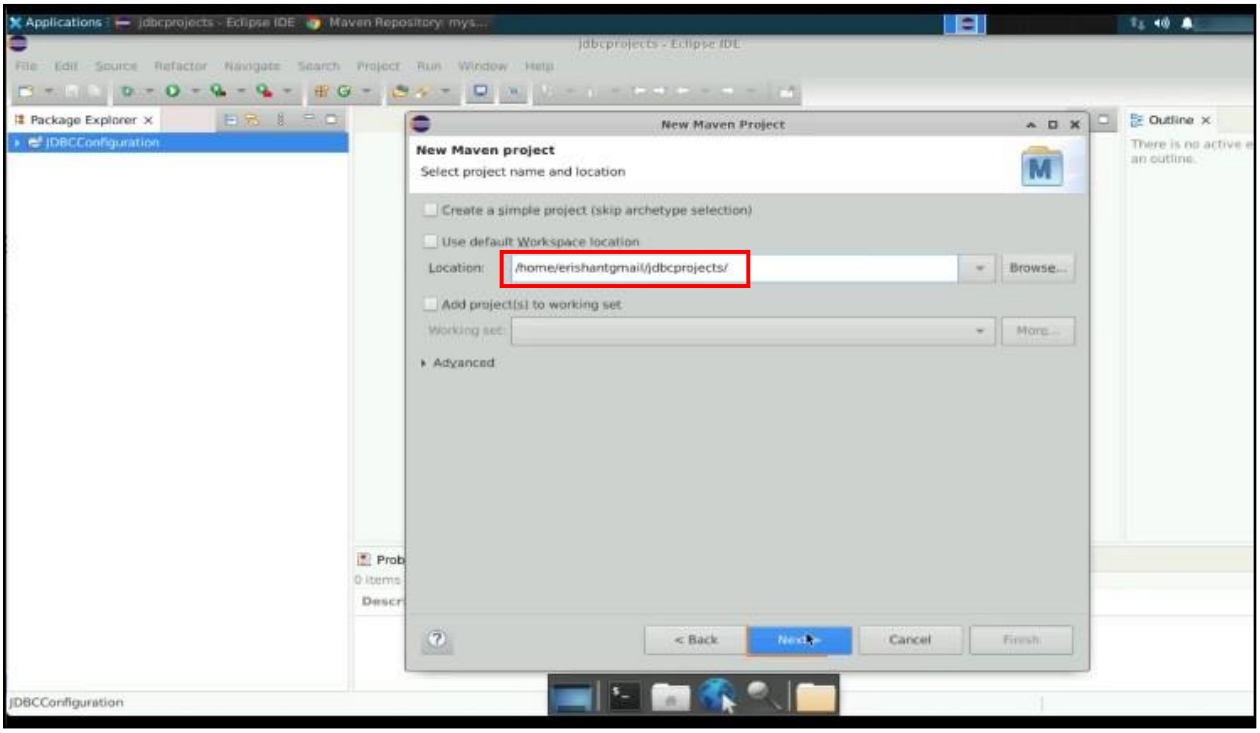
3.2 Search for **Maven** to create a Maven project as shown in the screenshot below:



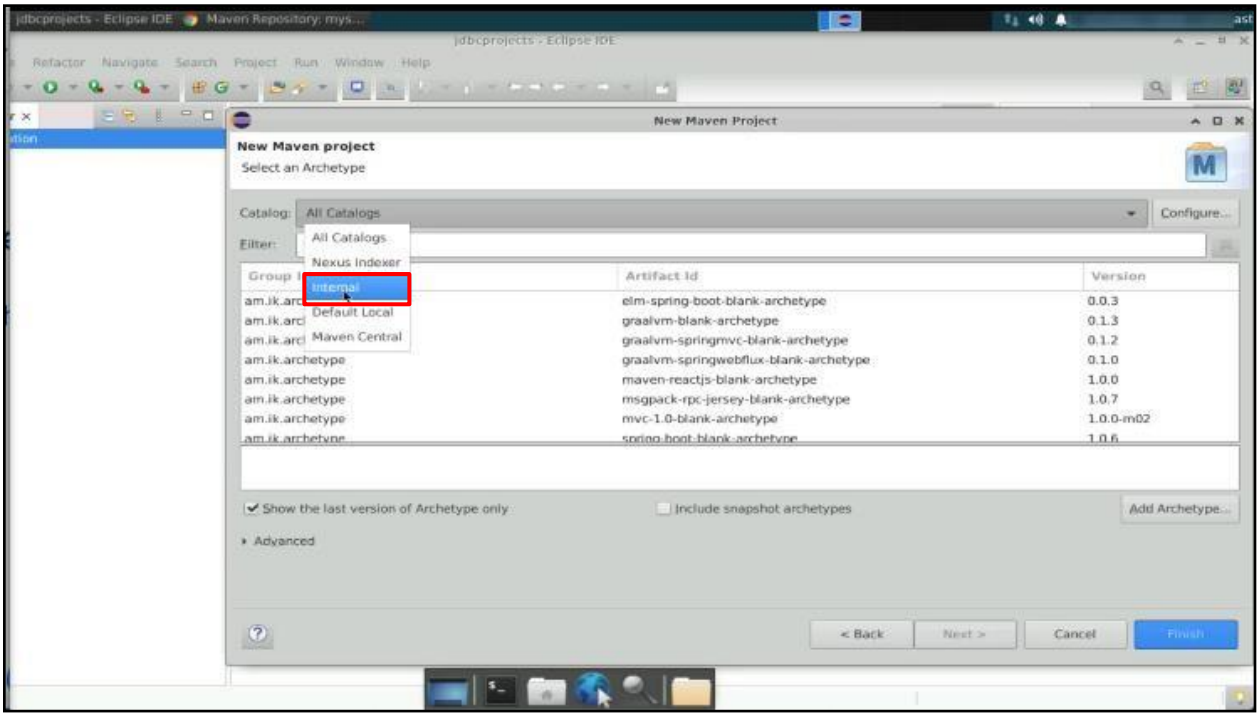
3.3 Click on **Maven Project** and select **Next** as shown in the screenshot below:



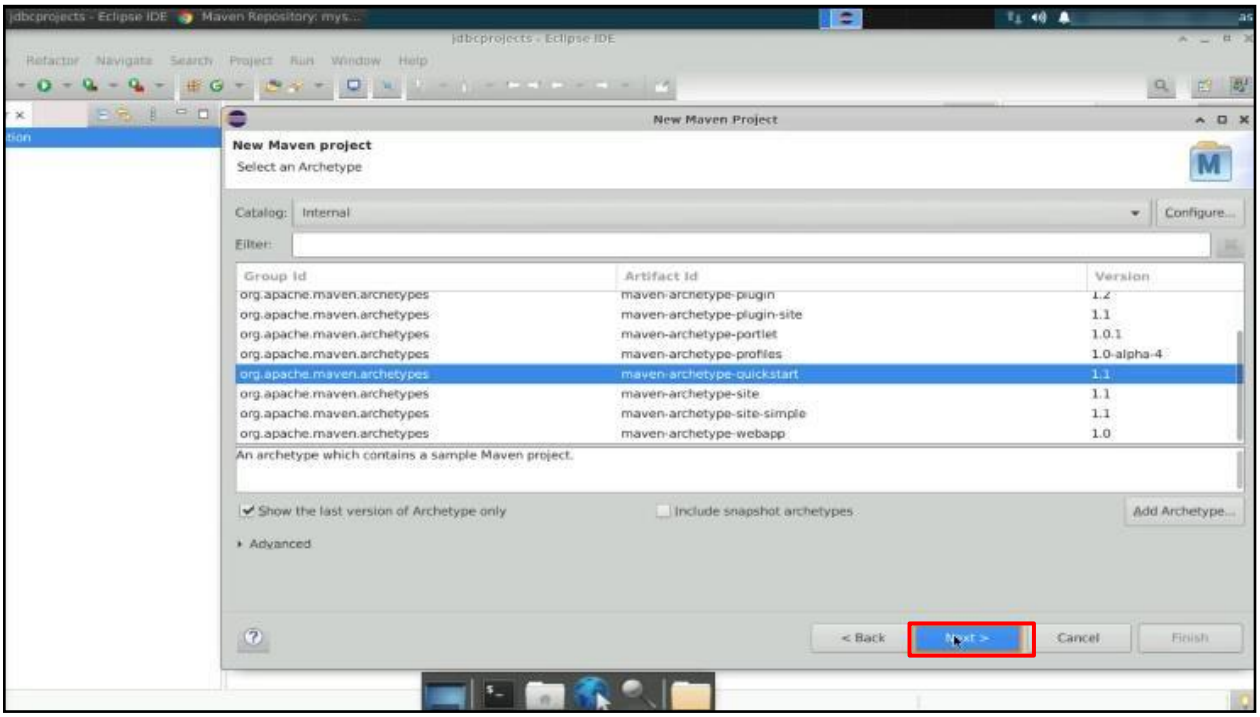
3.4 Set the **workplace location** to match that of the JDBC project, and click **Next** as shown in the screenshot below:



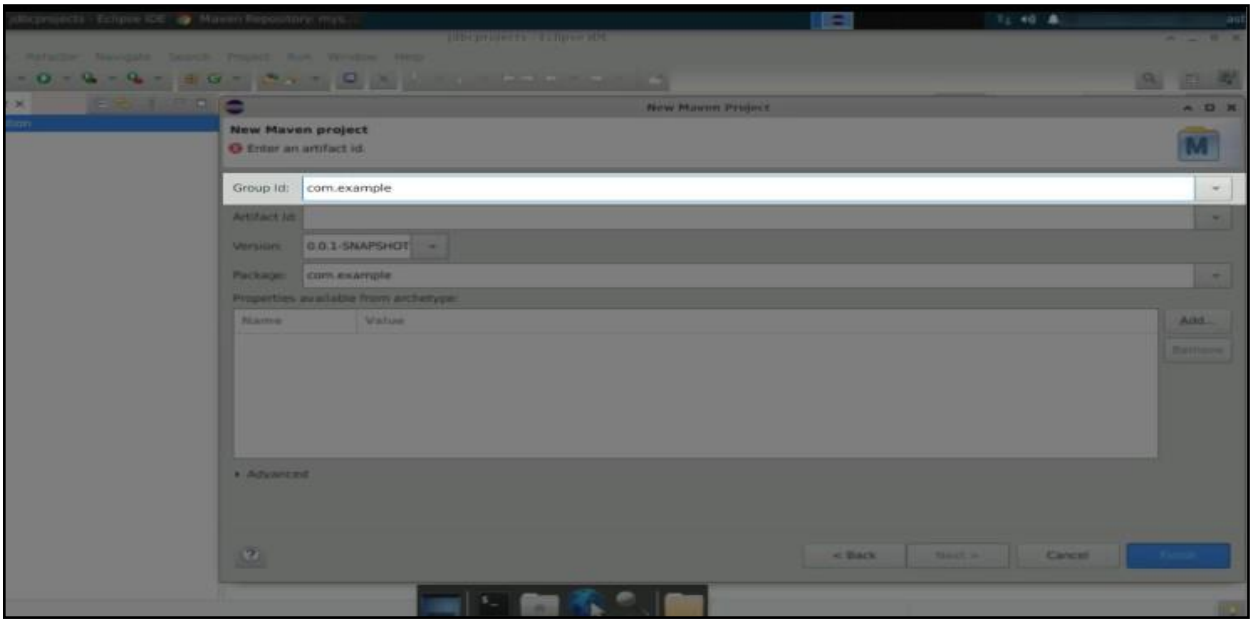
3.5 Select **Internal** from the catalog options as shown in the screenshot below:



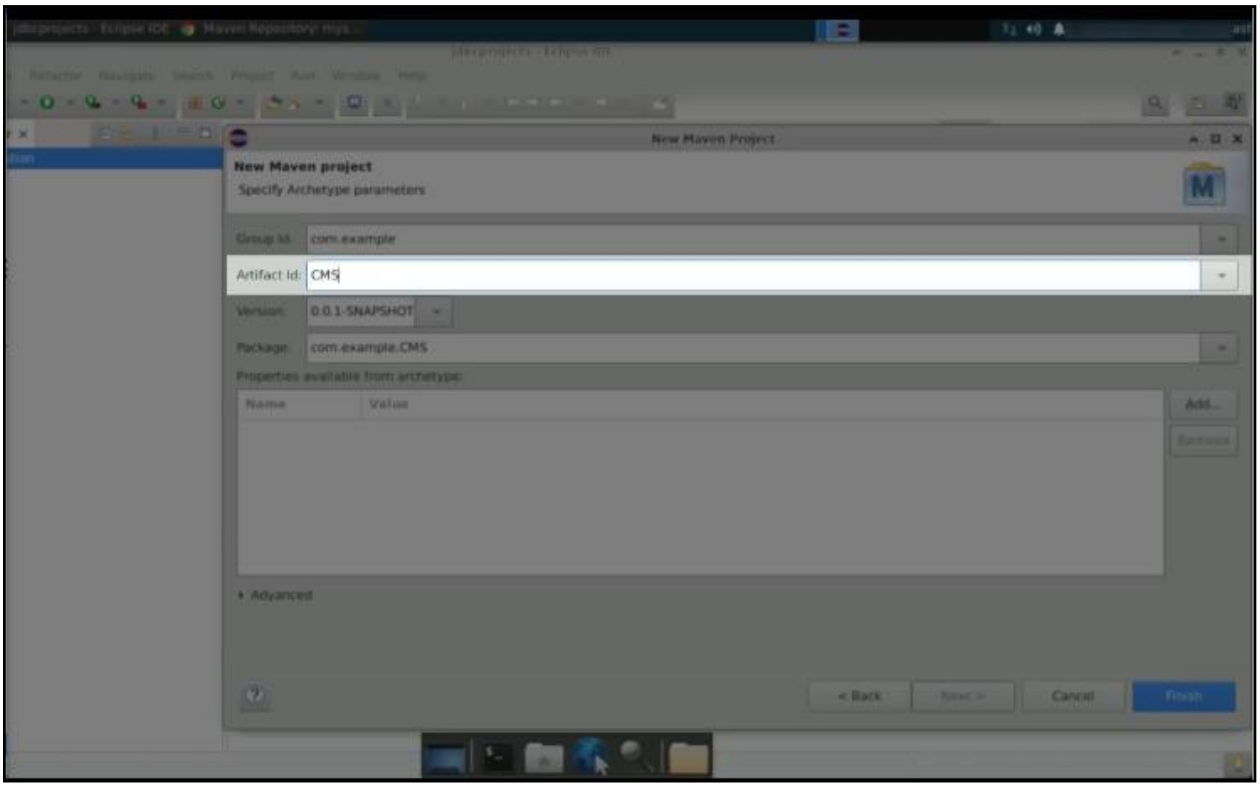
3.6 Select the **maven-archetype-quickstart** option from the options shown, and click on **Next** as shown in the screenshot below:



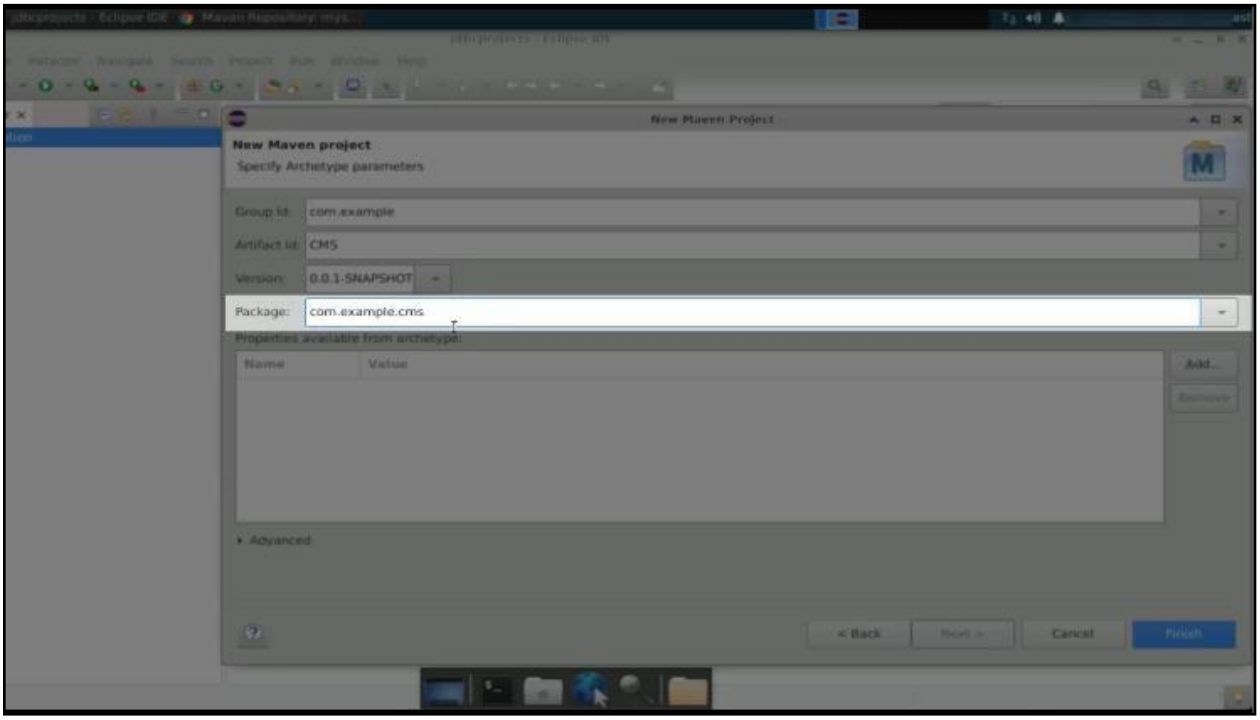
3.7 Provide the **Group Id,** which is your company's domain name in the reverse order:



3.8 Add an **Artifact Id,** which will be your application name, say **CMS** as shown in the screenshot below:

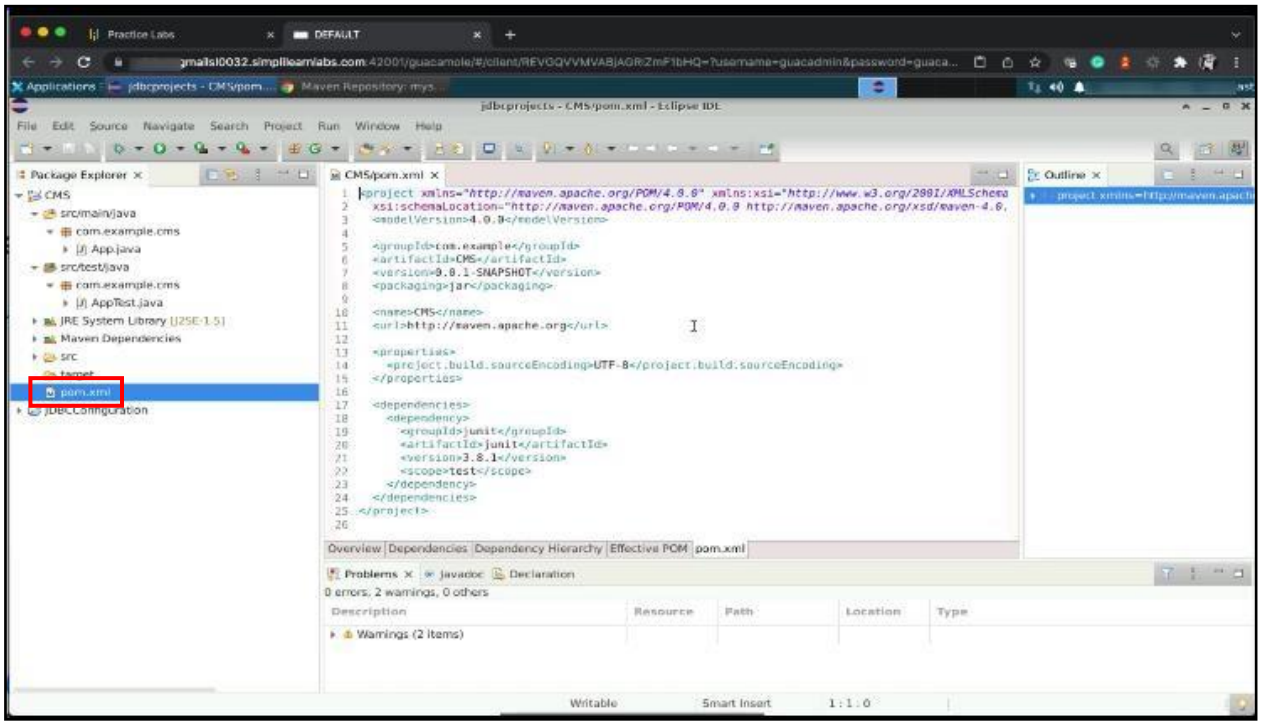


3.9 Add a package name, say **com.example.cms,** and select **Finish** as shown in the screenshot below:

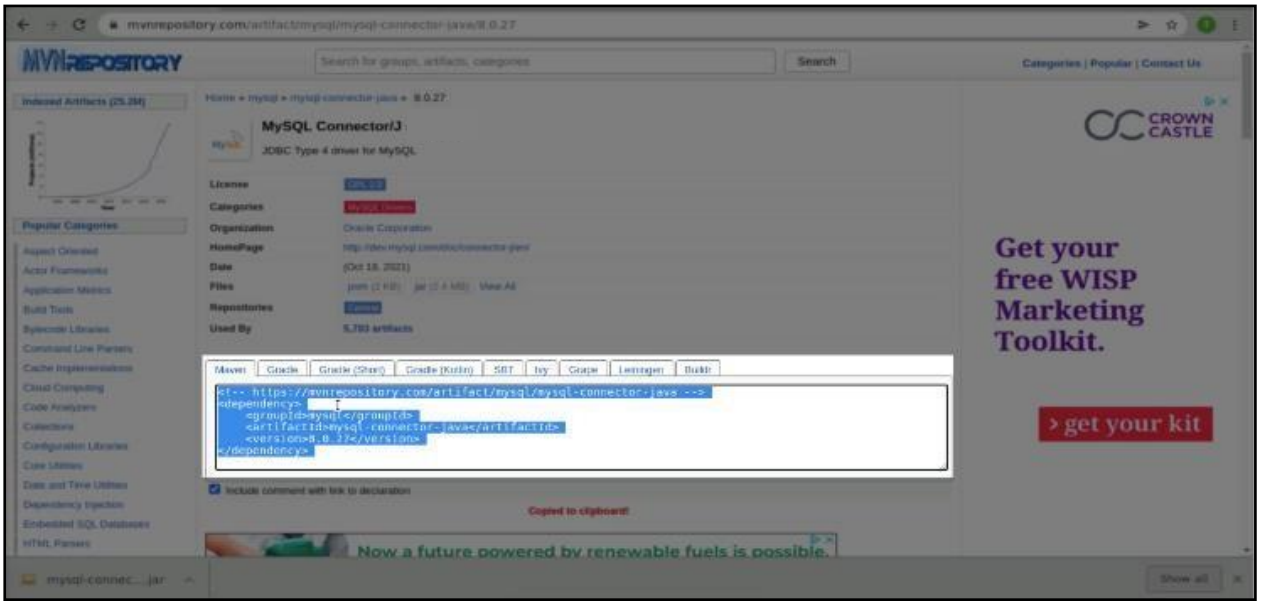


Step 4: Create an interface in com.example.cms

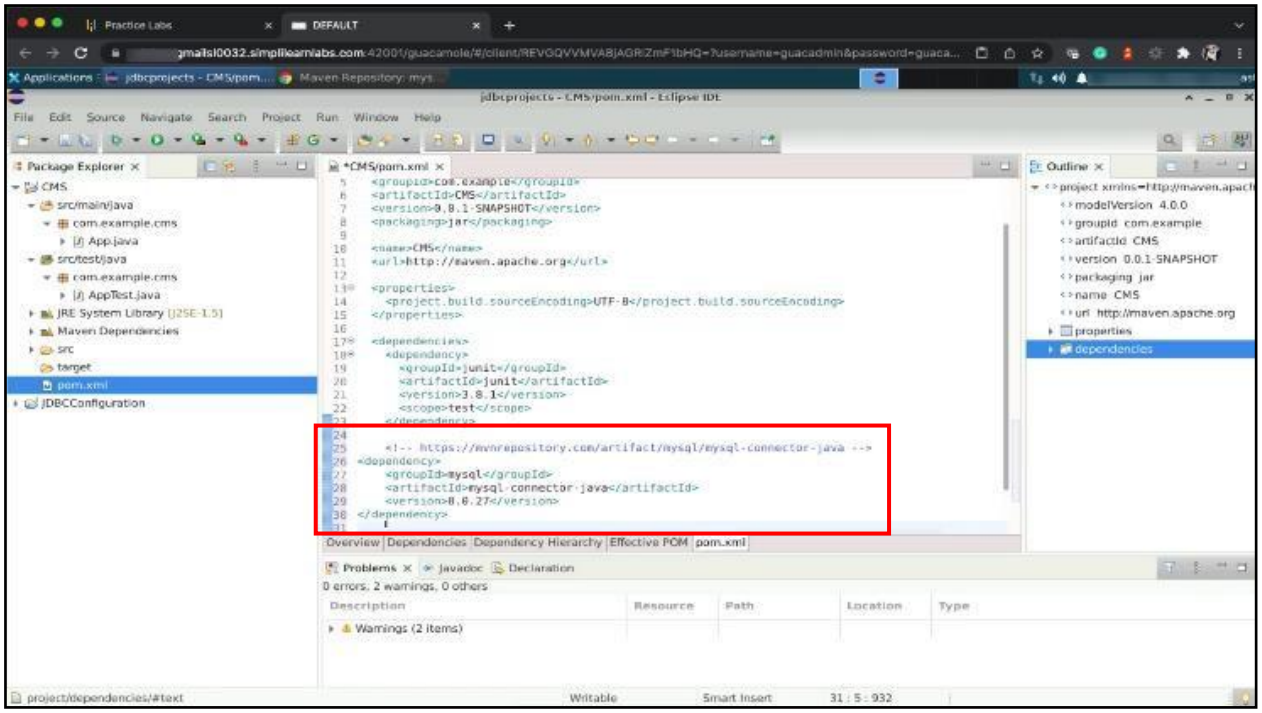
4.1 Go to the **pom.xml** file in the left bar as shown in the screenshot below:



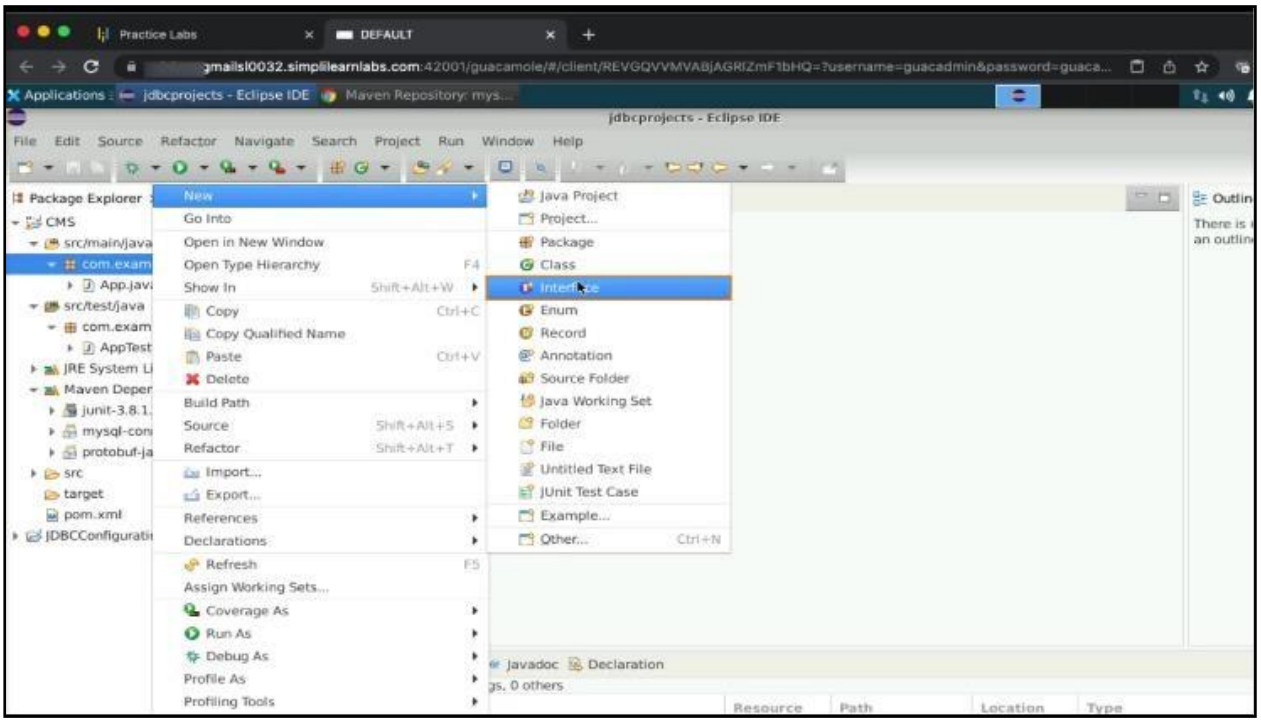
4.2 Return to **mvnrepository** and copy the code for the mysql-connector dependency as shown in the screenshot below:



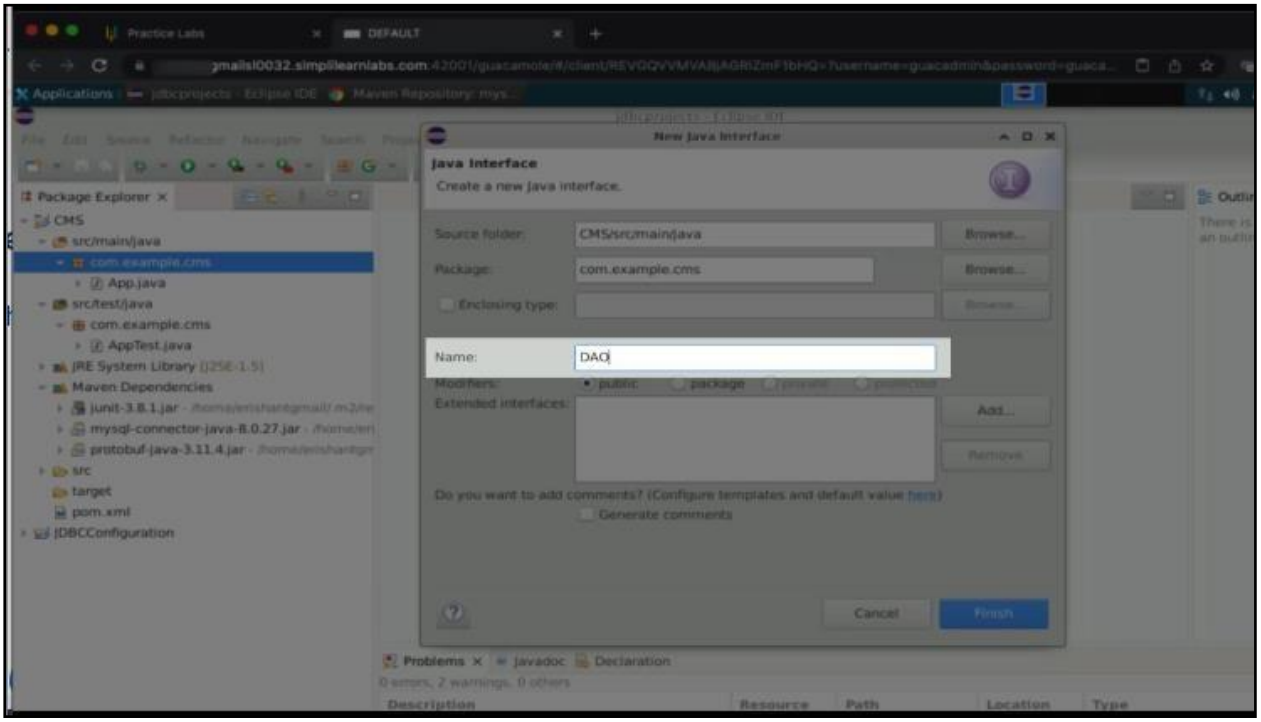
4.3 Paste the dependencies in the **pom.xml** file from line 25 as shown in the screenshot below:



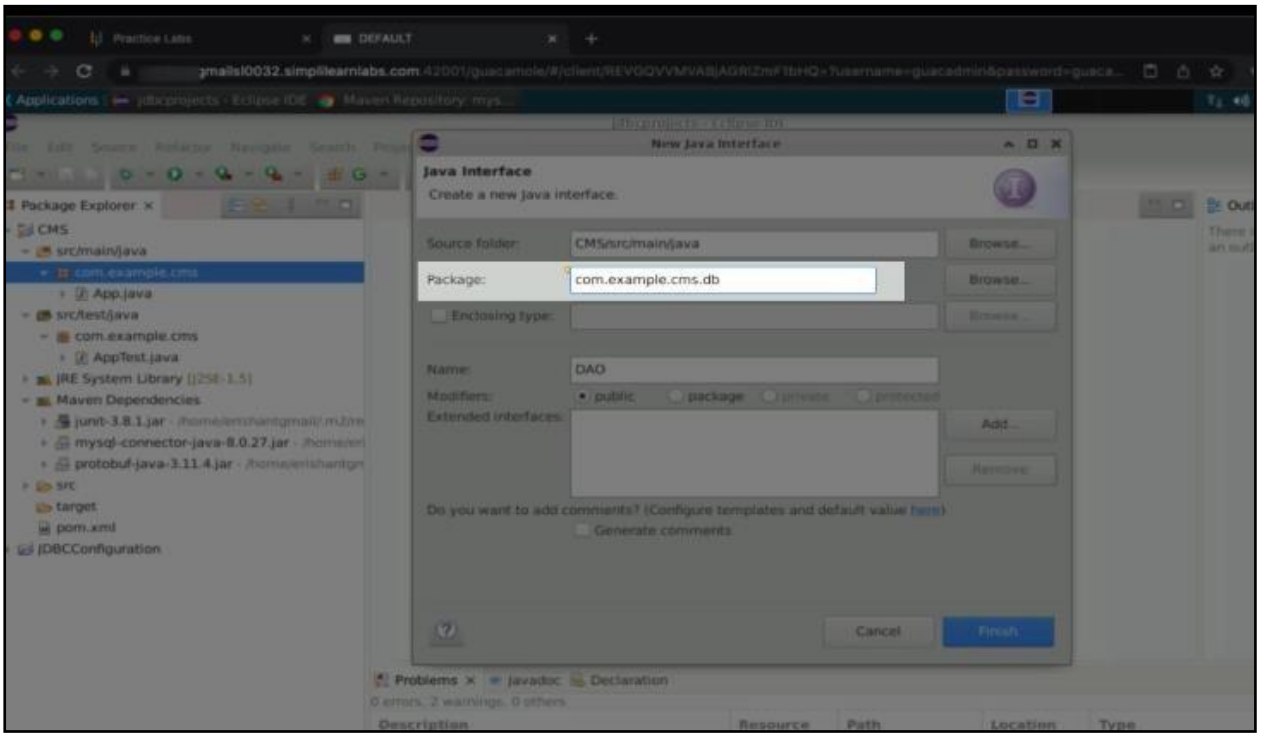
4.4 Right-click on **com.example.cms**, select **New** and create an interface as shown in the screenshot below:



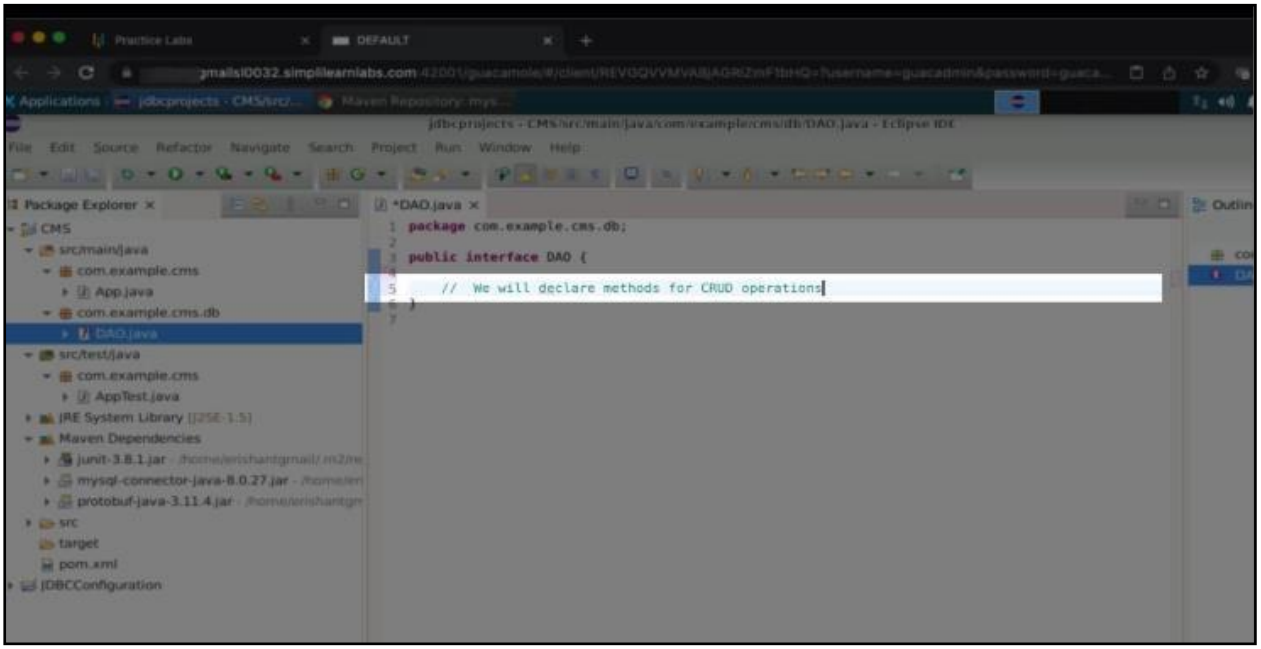
4.5 Provide a name as **DAO**



4.6 Add a package name, say **com.example.cms.db,** and click on **Finish**

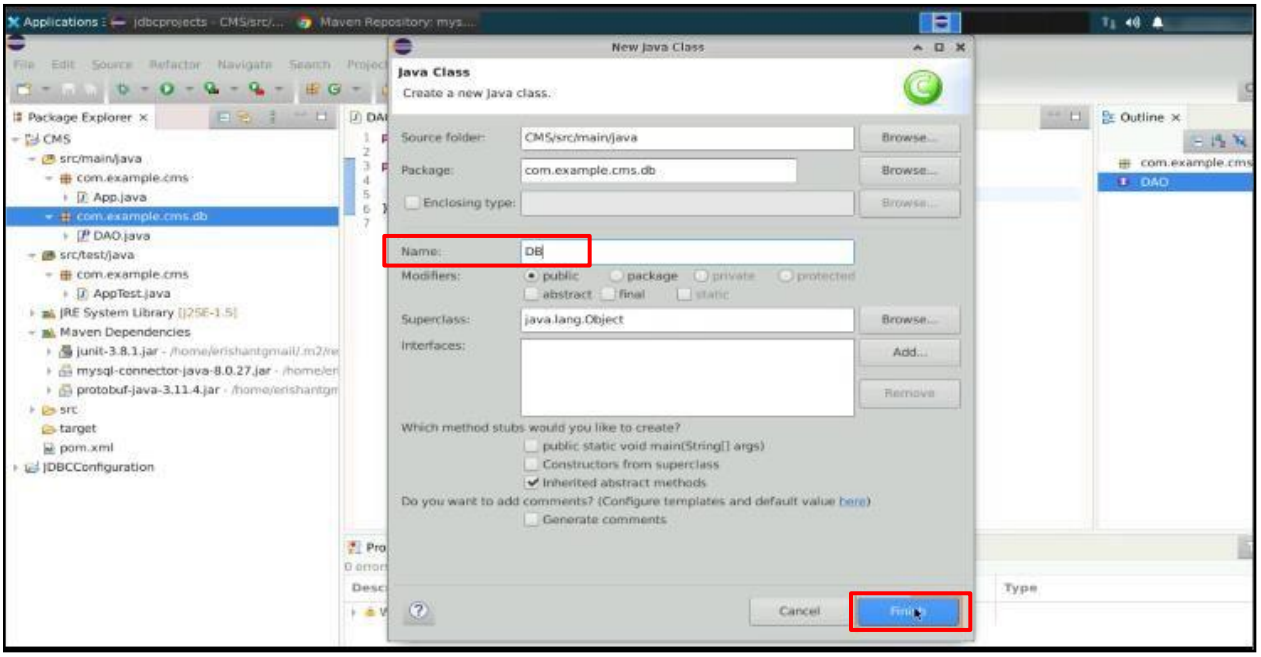


4.7 Define the methods for CRUD operations

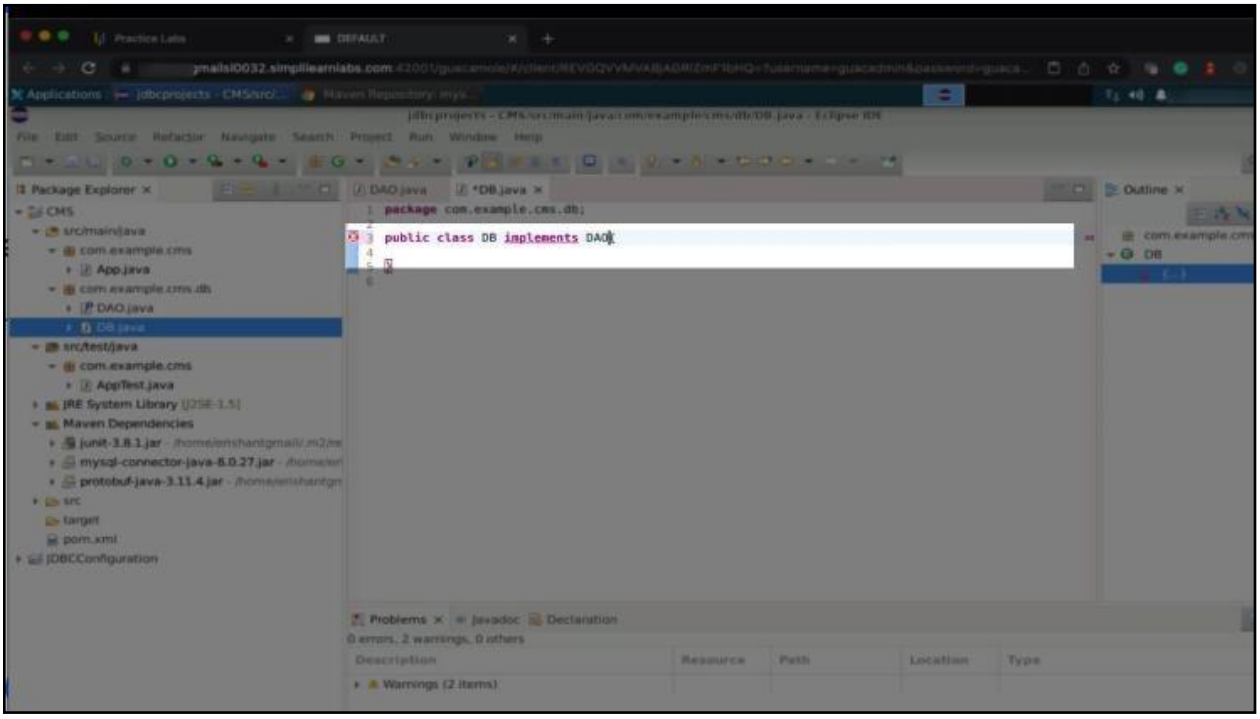


Step 5: Create the DB class

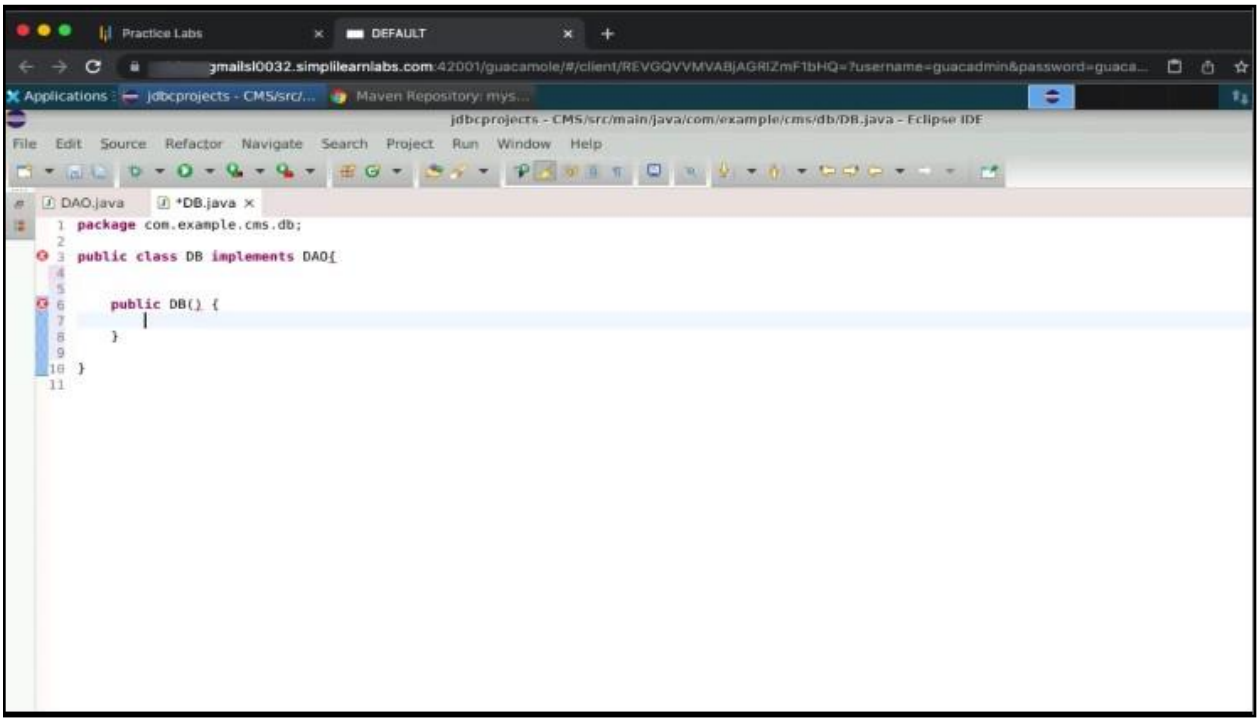
5.1 Create a new class named **DB**, and click on **Finish**



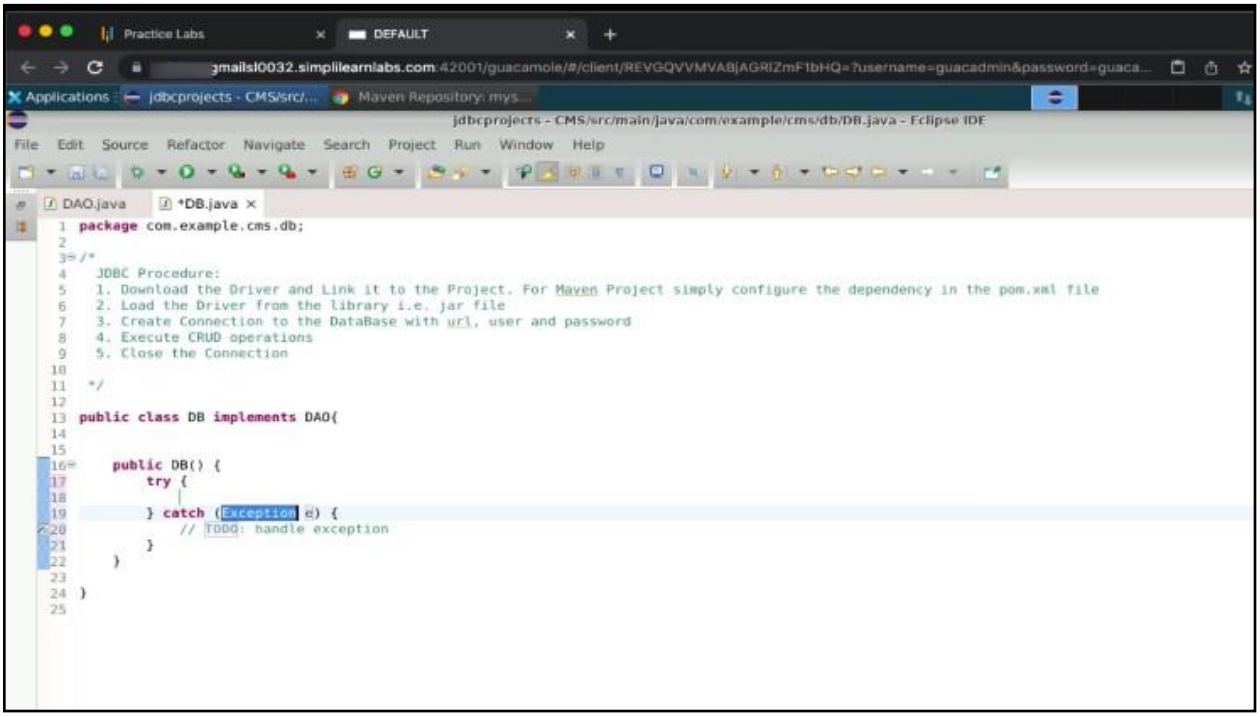
The **DB** class is the implementation file of the **DAO** interface.



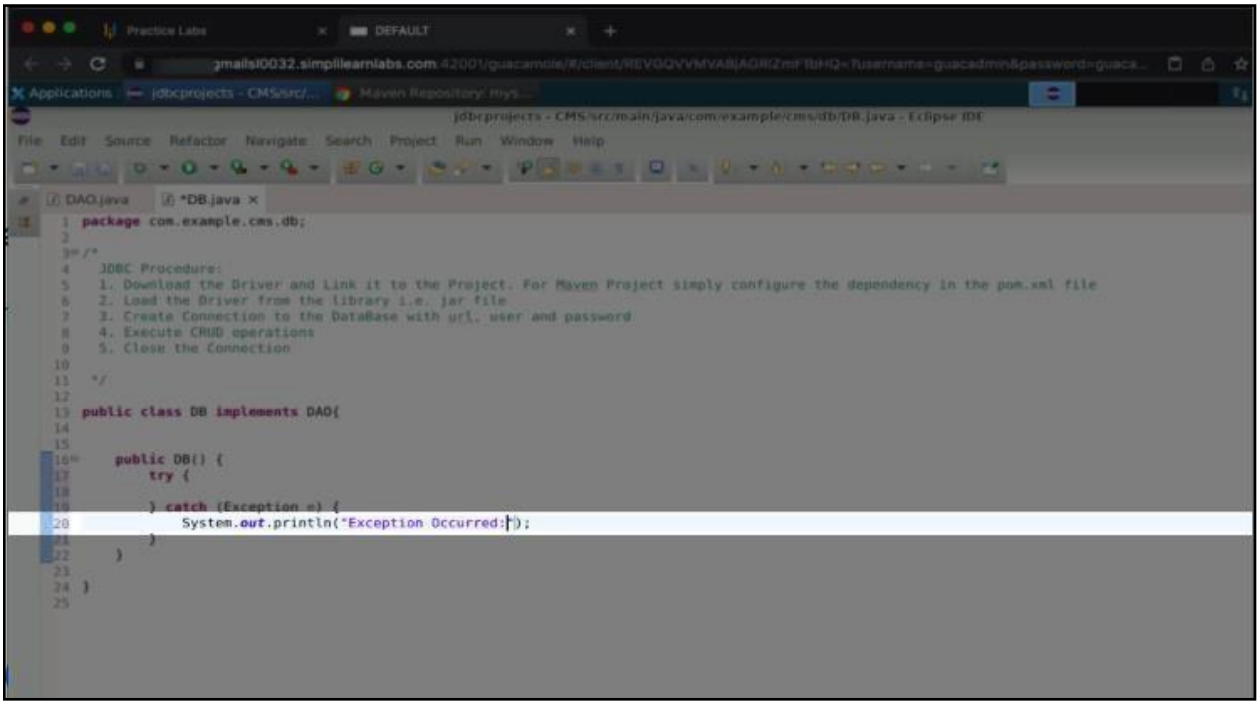
5.2 Create a constructor in the **DB** class as shown in the screenshot below:



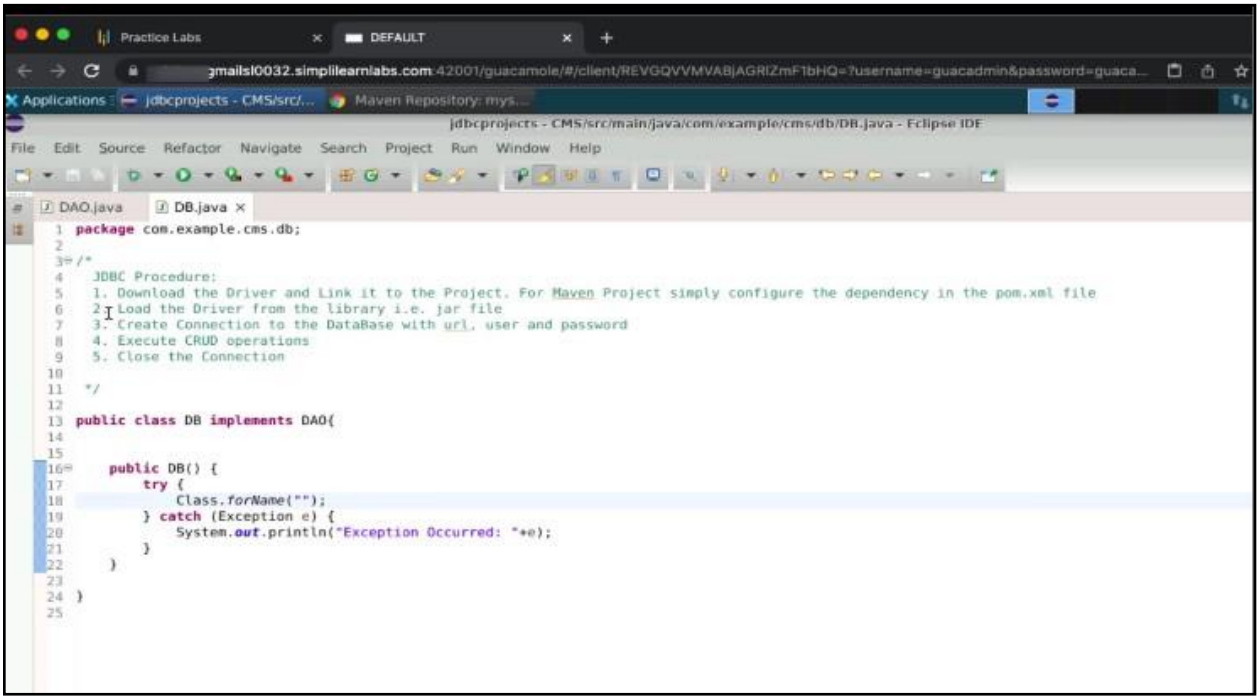
5.3 Use a **try-catch** block in the **constructor** to configure the database



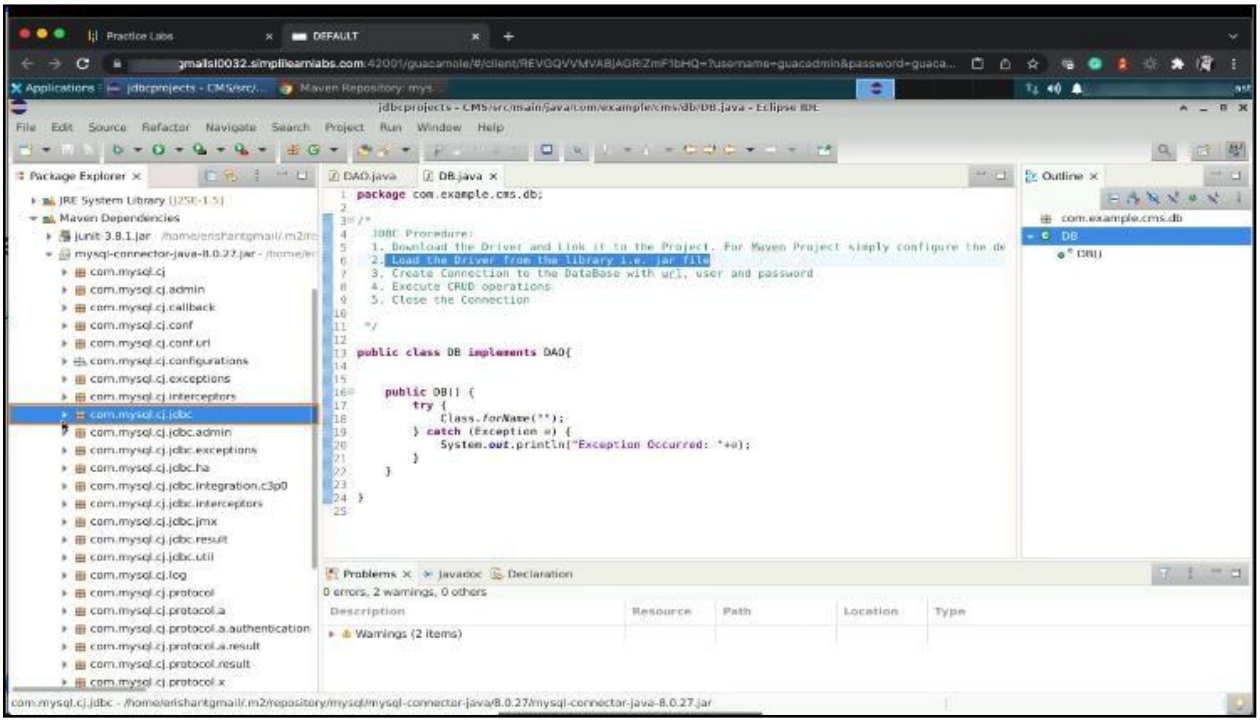
5.4 Manage the **Exception** in the catch block



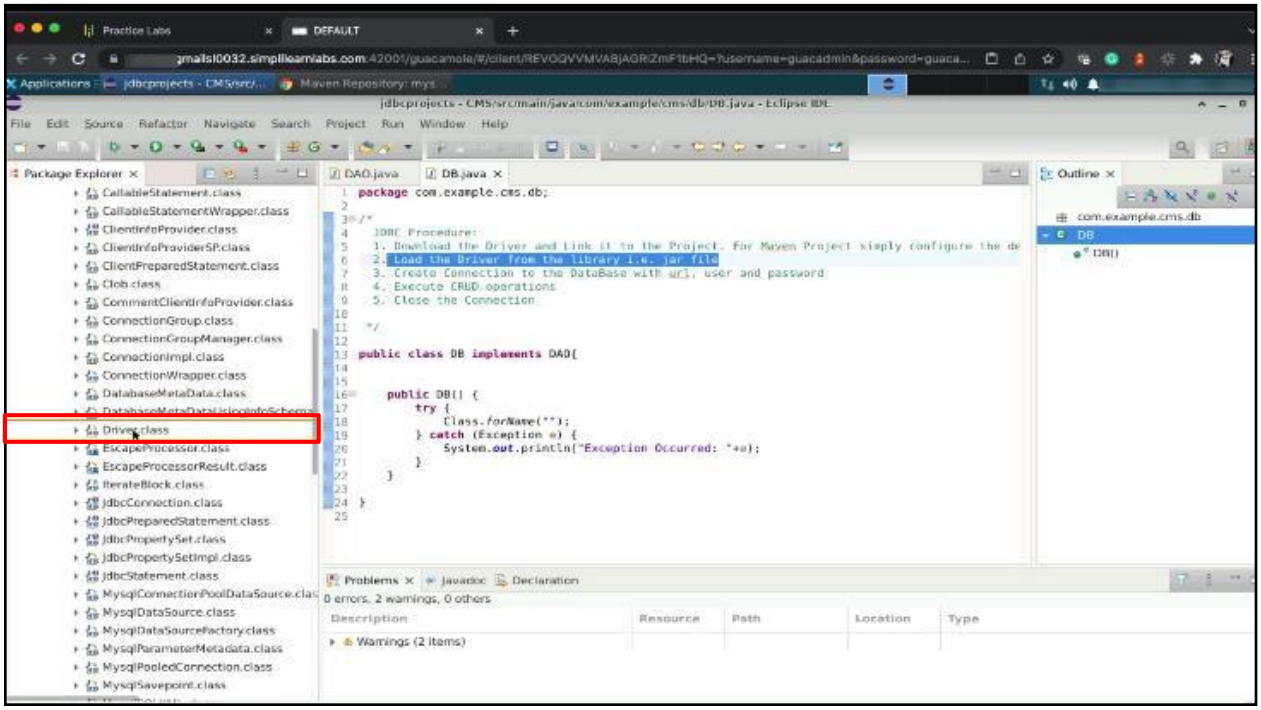
5.5 Add an API named **Class**



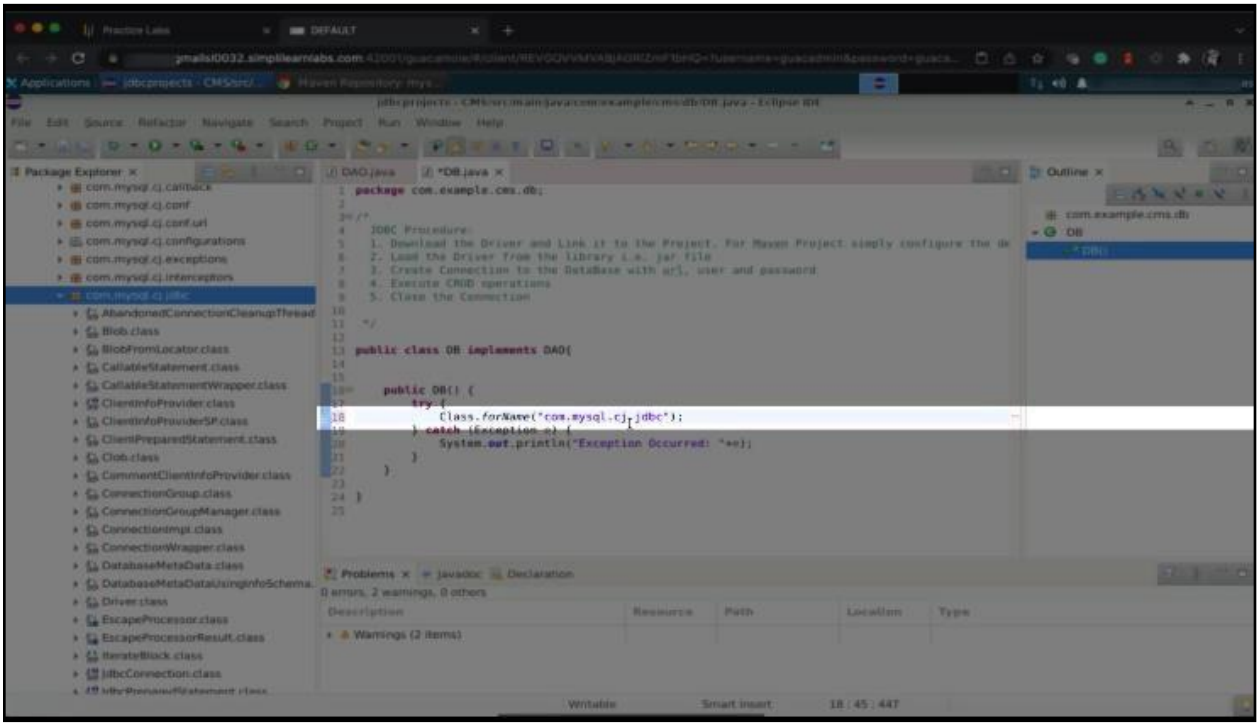
5.6 Open **mysql-connector-java-8.4.0.jar** and find the package containing the driver class



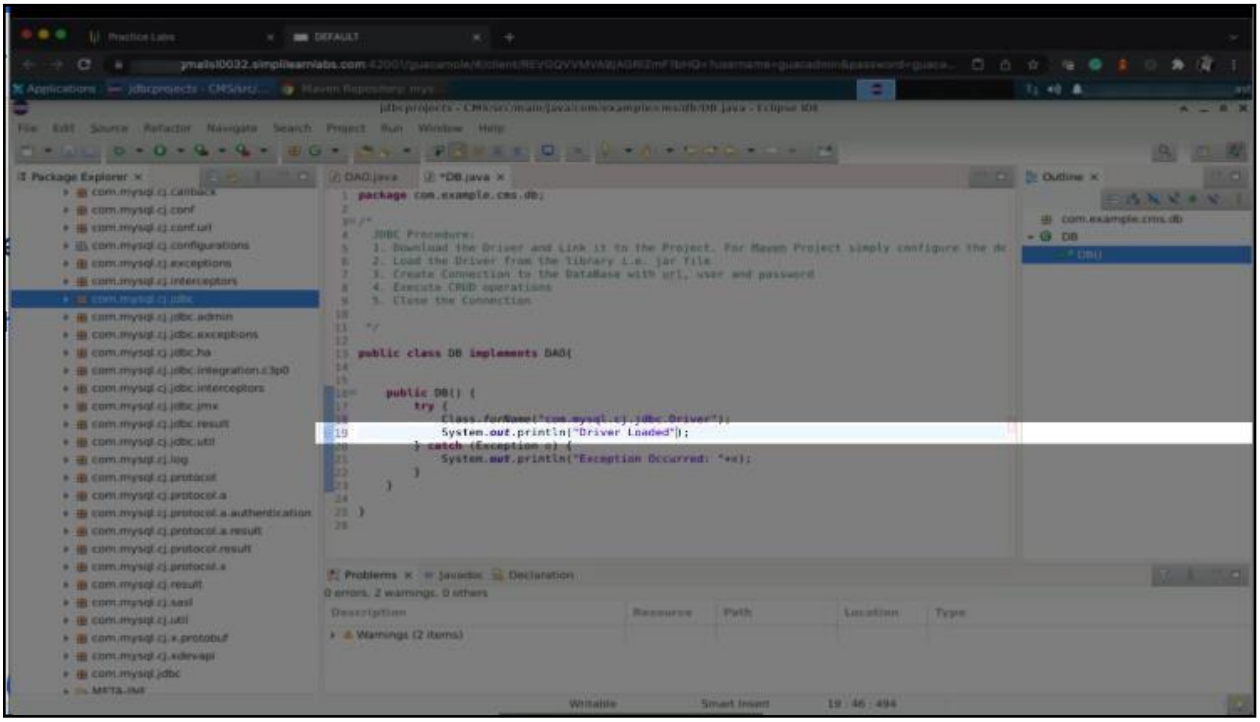
5.7 Select the **Driver** class as shown in the screenshot below:



5.8 Copy and paste the package name as shown in the screenshot below:

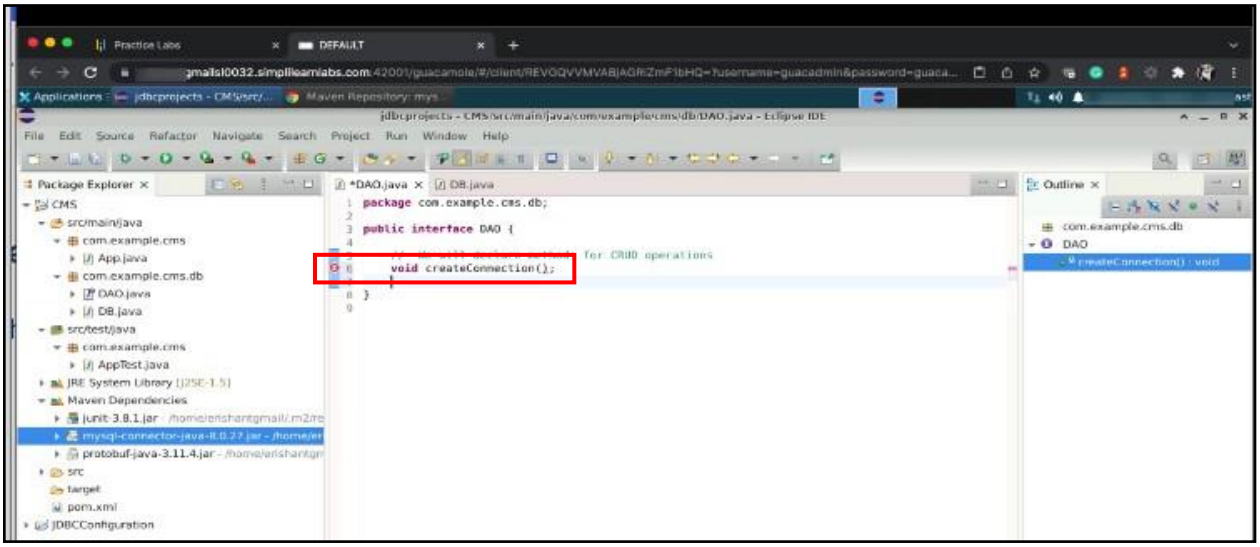


5.9 Include a print statement to load the driver as shown in the screenshot below:

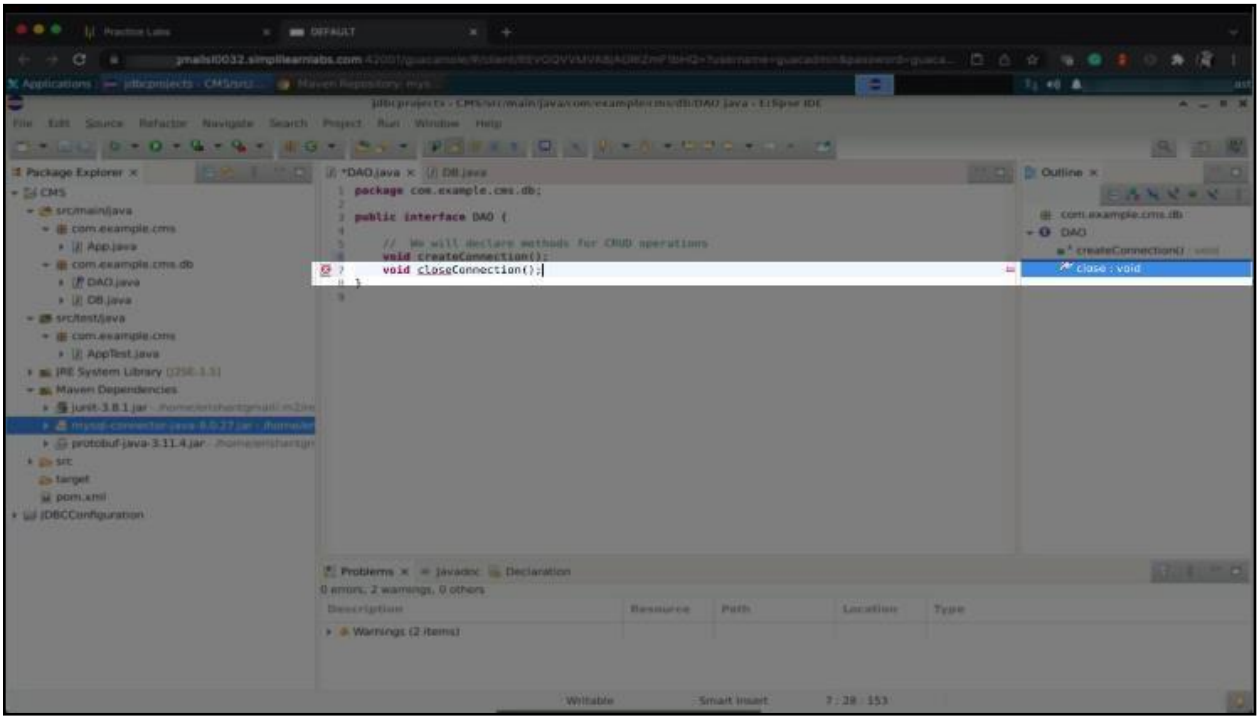


Step 6: Create a connection in the DAO.java file

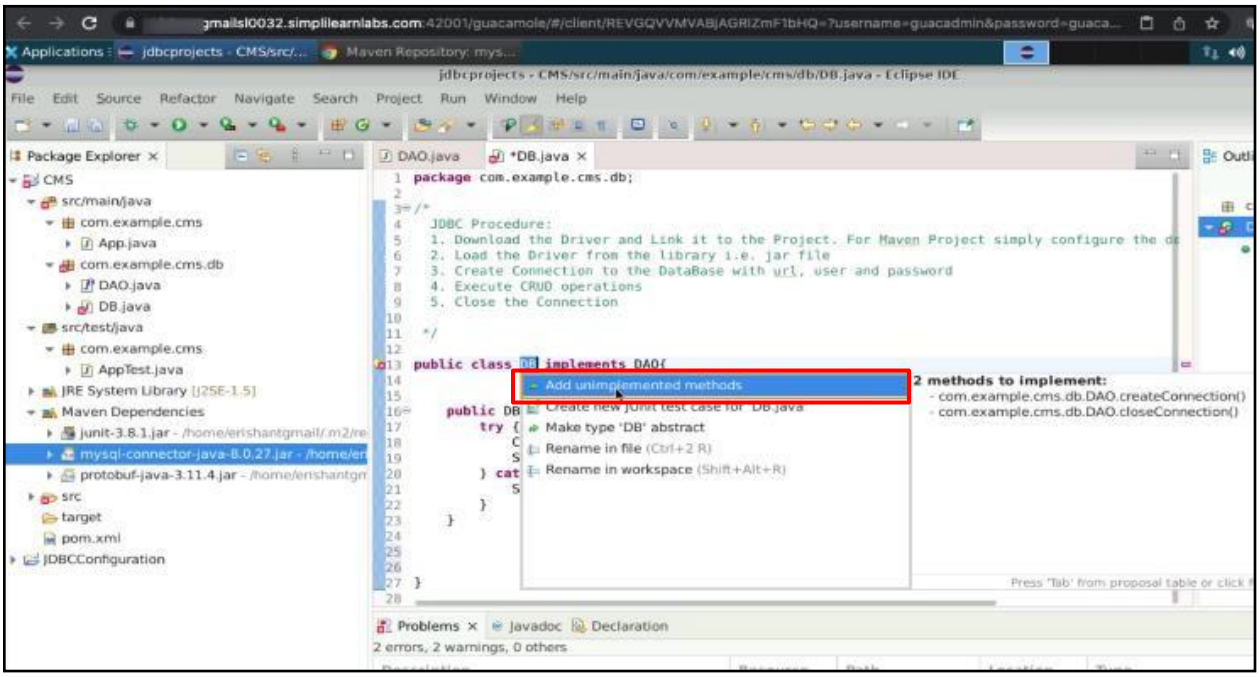
6.1 Create a connection in the **DAO.java** file as shown in the screenshot below:



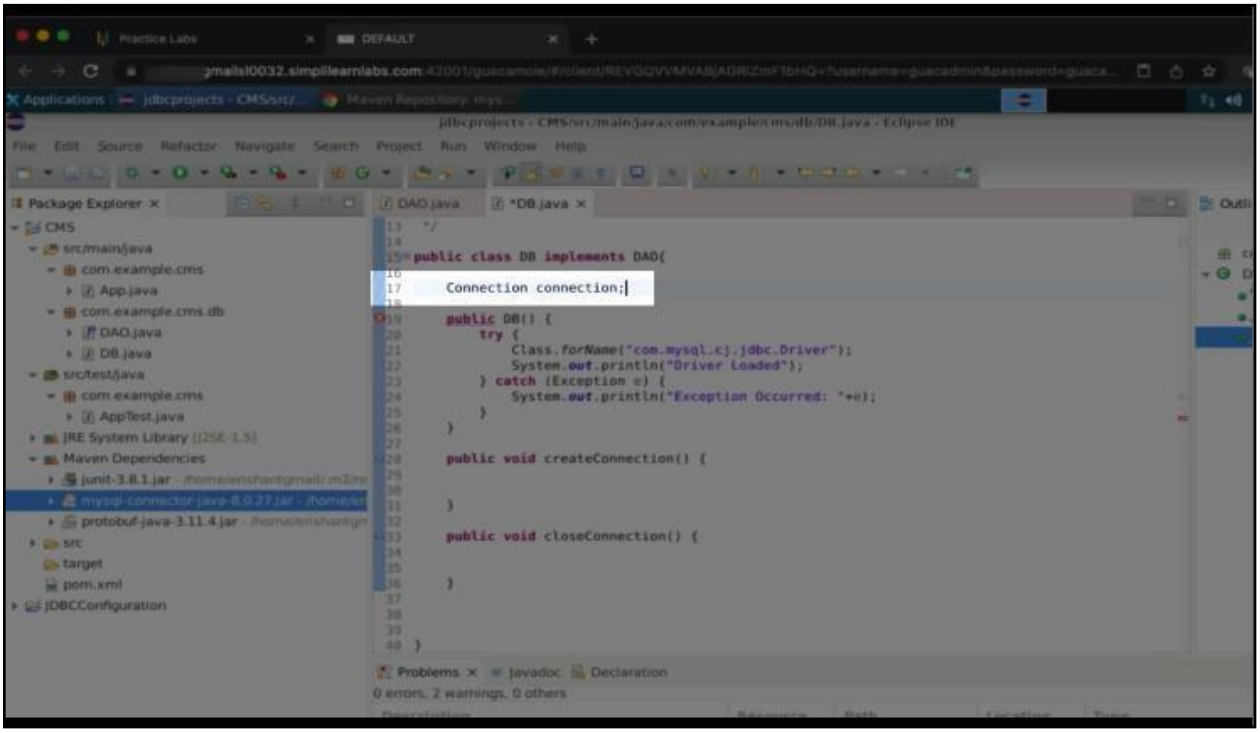
6.2 Add the **closeConnection()** method in the same file:



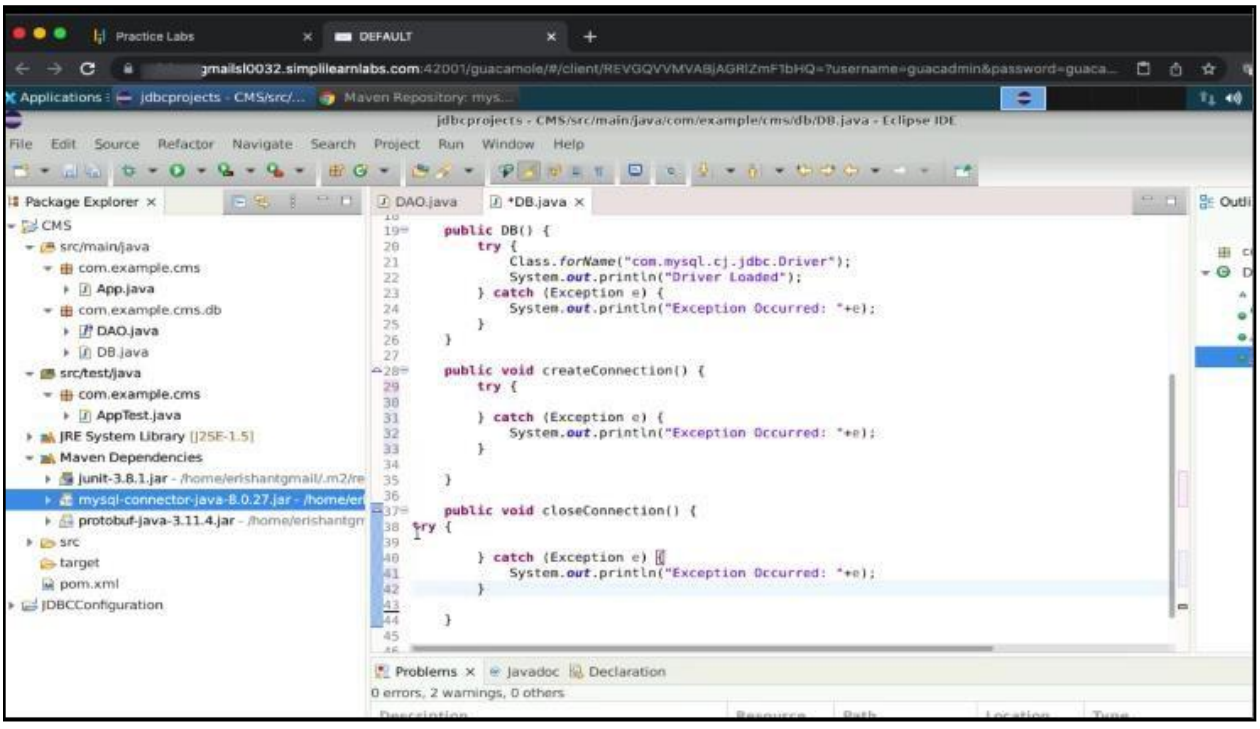
6.3 Select **Add unimplemented methods** in the **DB.java** file as shown in the screenshot below:



6.4 Add an **API** from JDBC

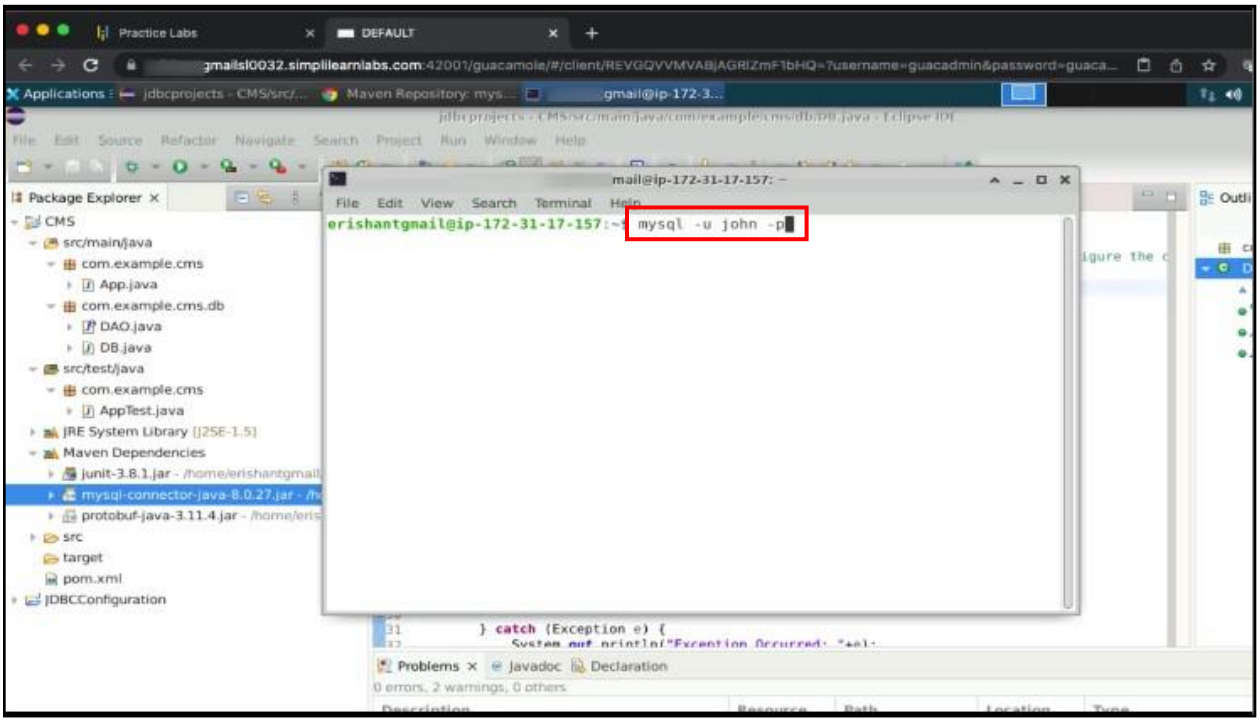


6.5 Use **try-catch** in both the methods as shown in the screenshot below:

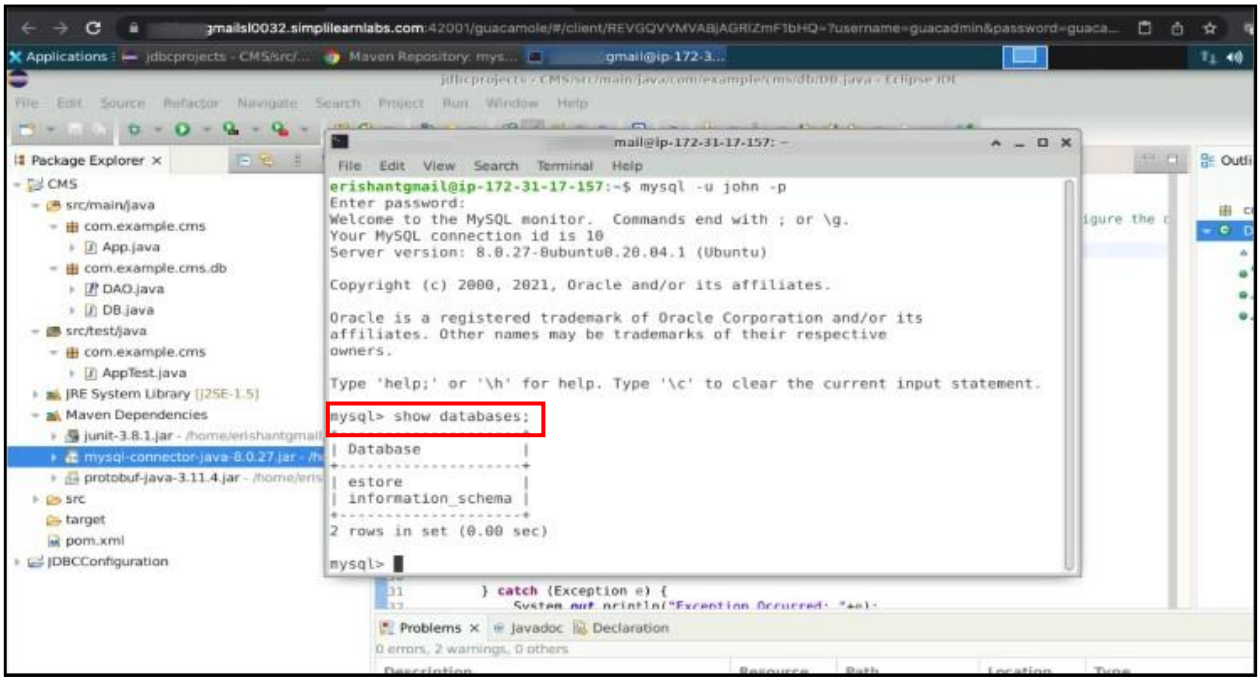


6.6 Open the terminal window and type the command:

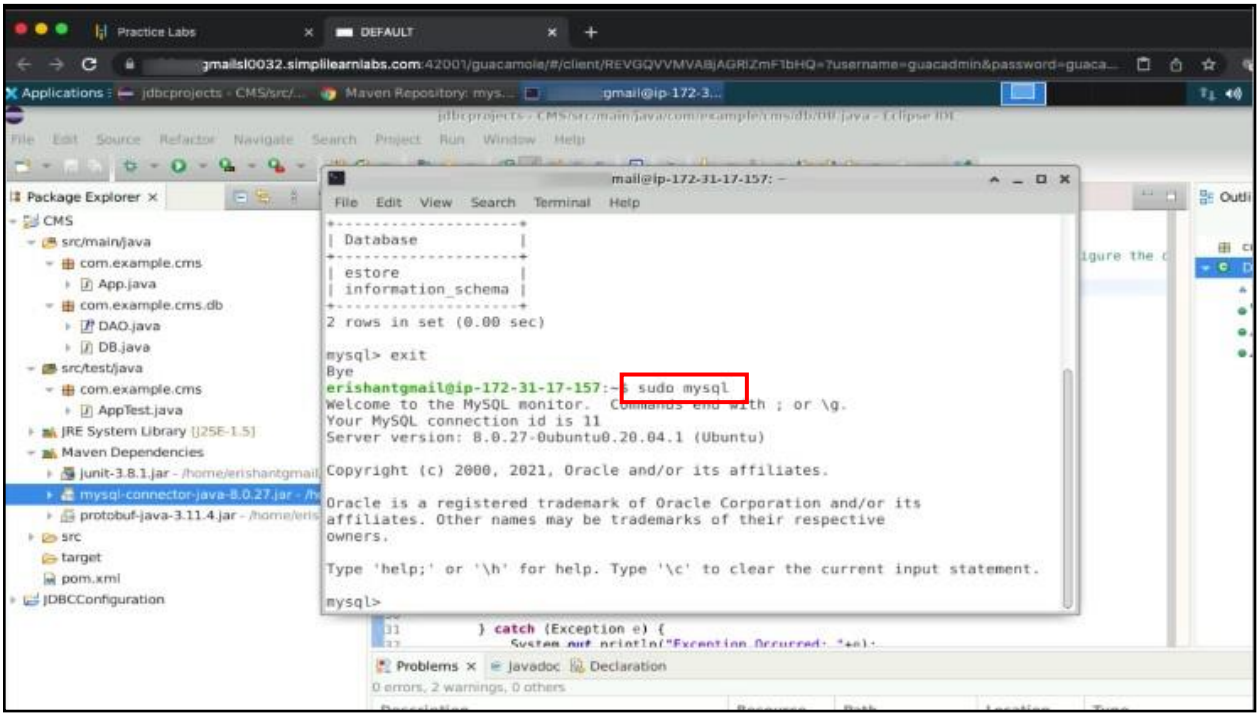
mysql -u john -p



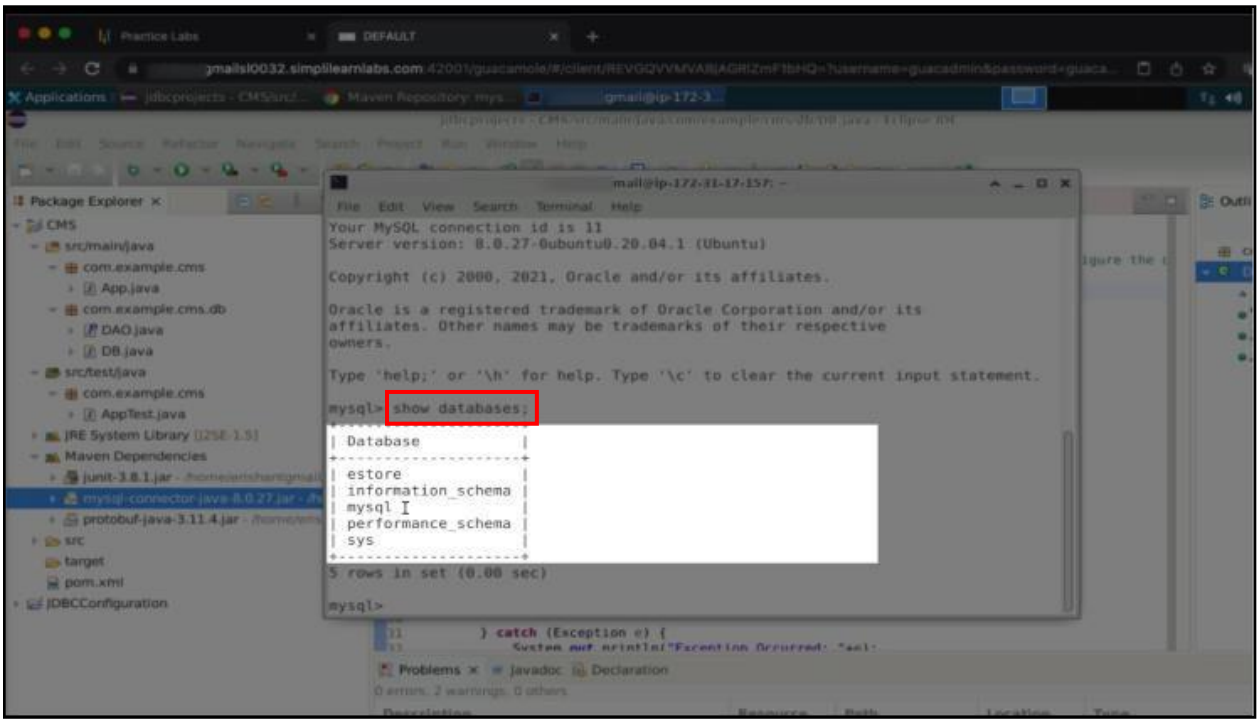
6.7 Type the **show databases;** command to see the available database names



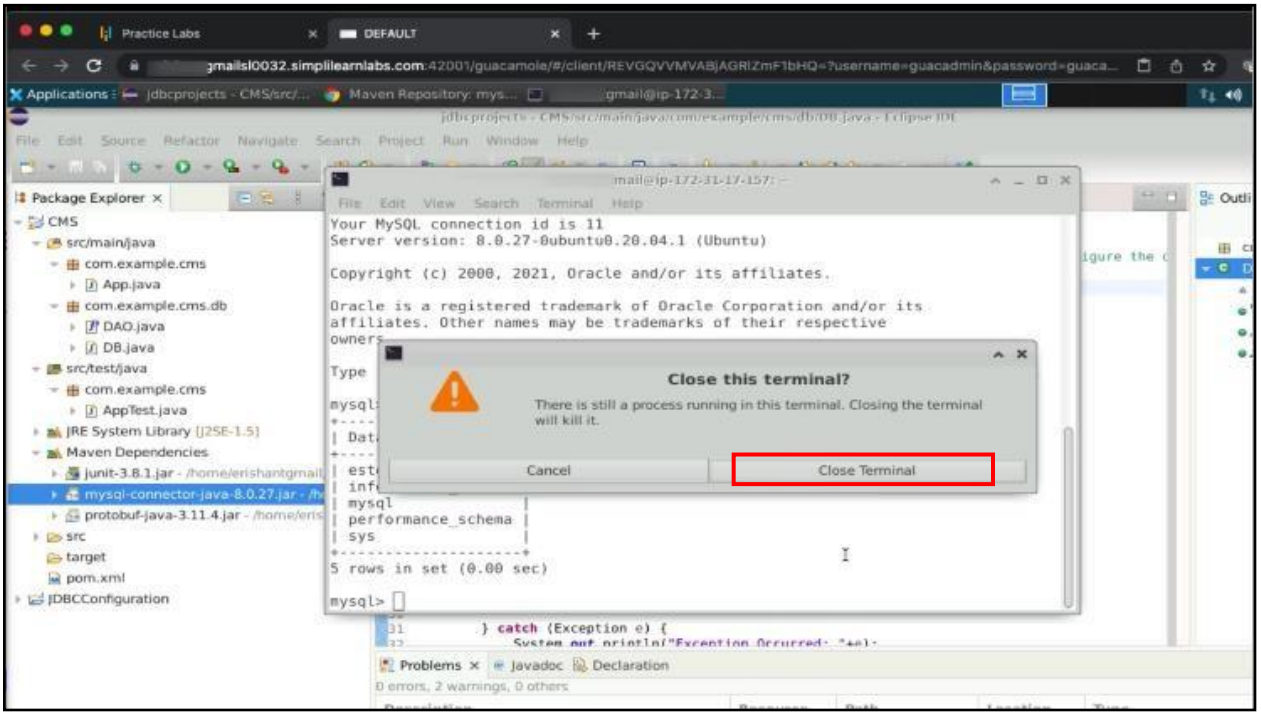
6.8 Type the **sudo mysql** command to log in as a **sudo user**



6.9 Use the **show databases;** command to view some more databases as shown in the screenshot below:

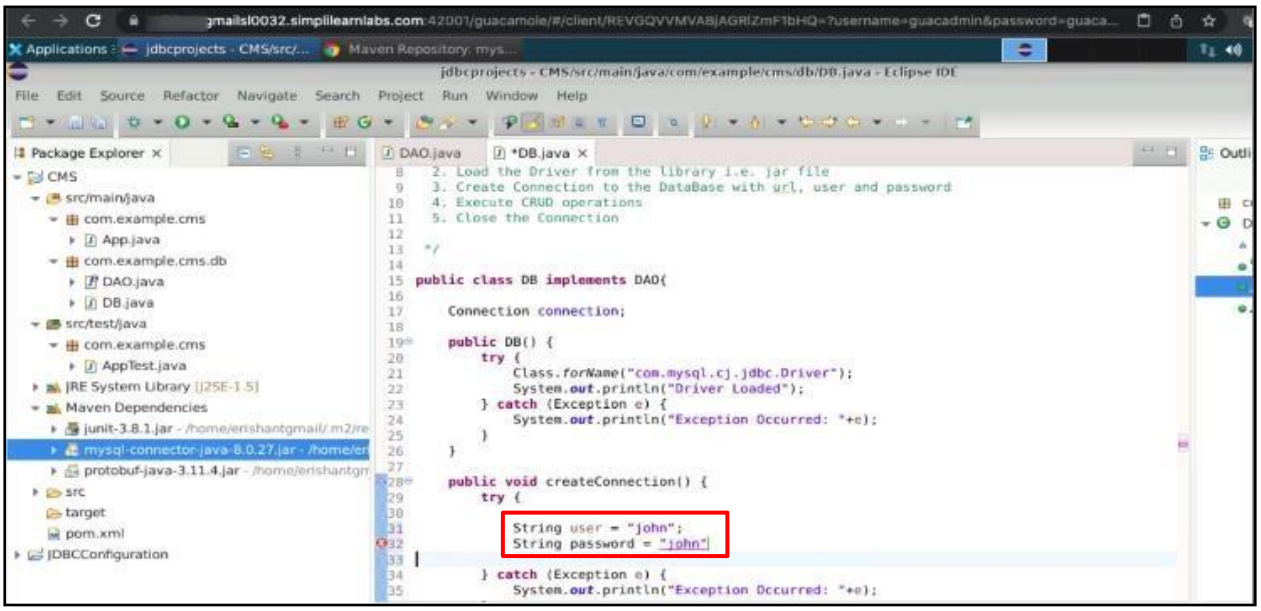


6.10 Close the **terminal**

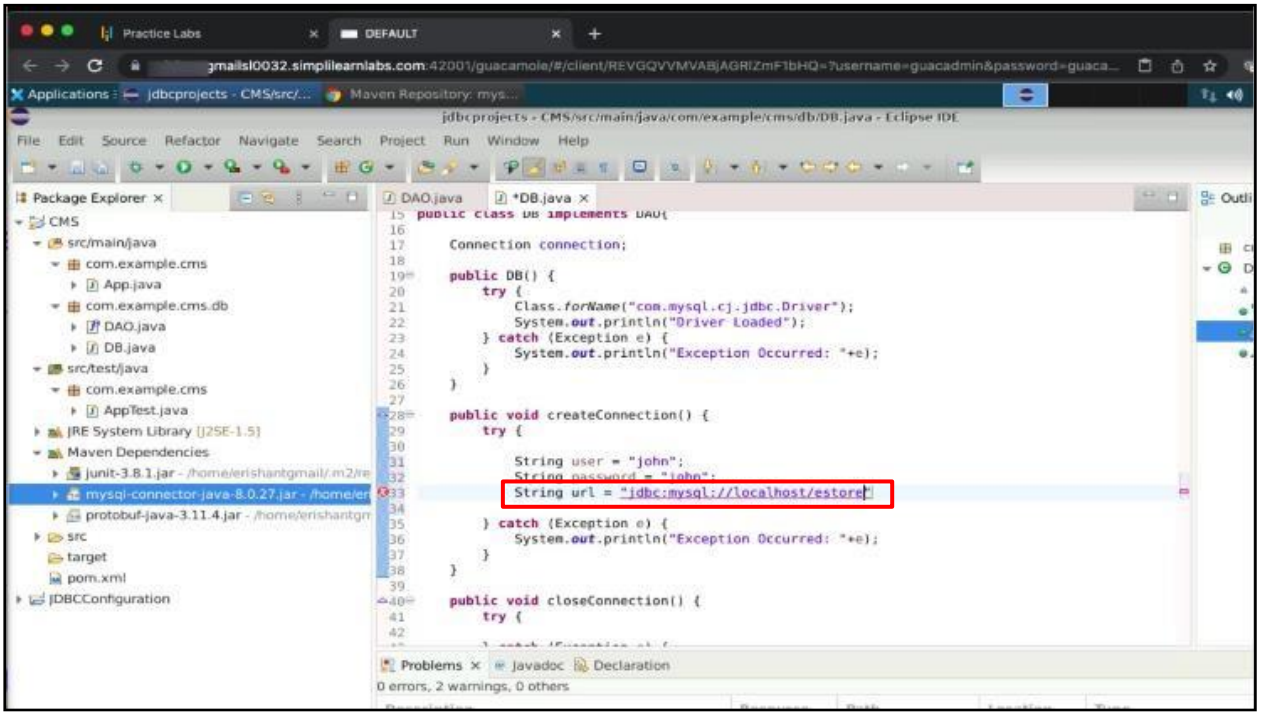


Step 7: Initialize the connection

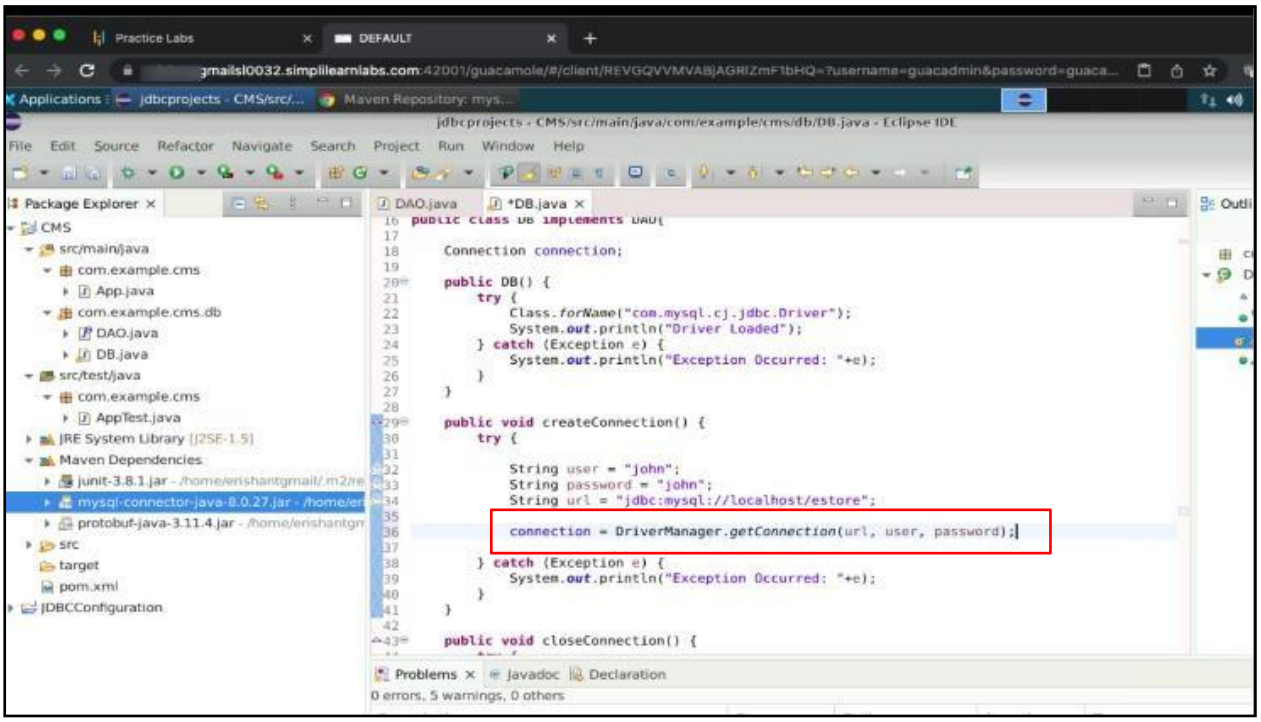
7.1 Add a **username and password** for the connection as shown in the screenshot below:



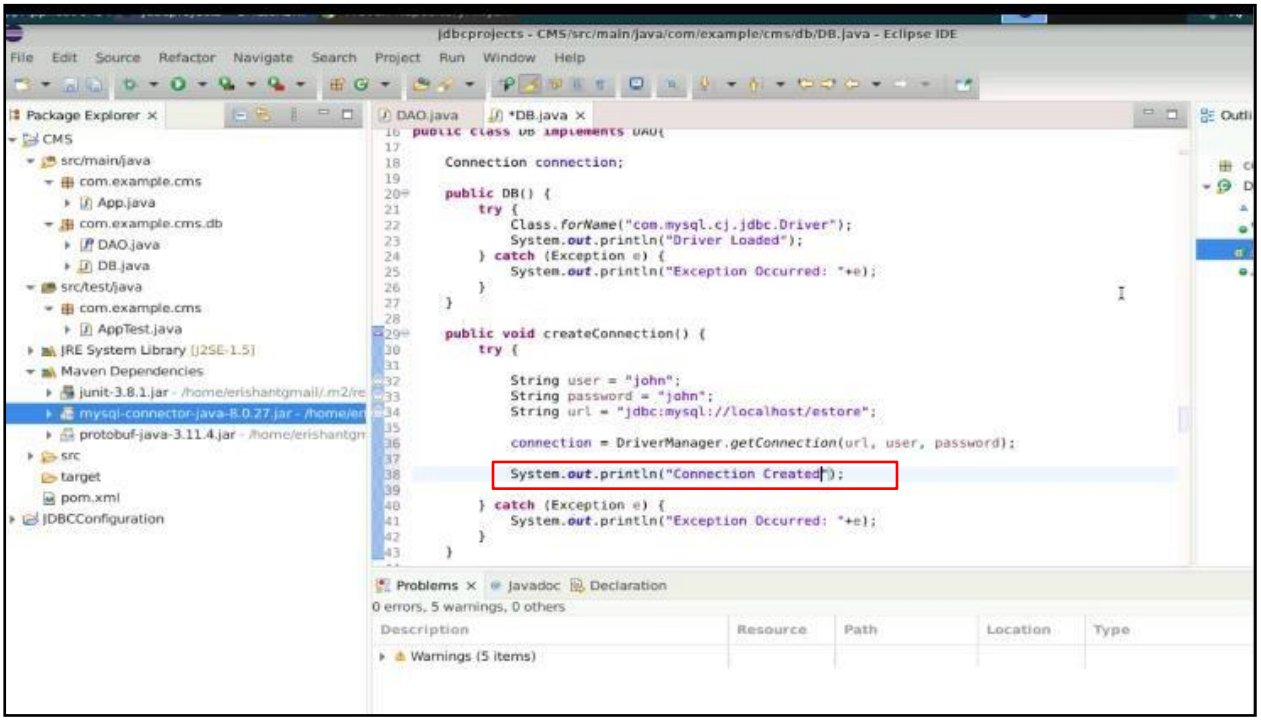
7.2 Add a **URL** to connect MySQL as shown in the screenshot below:



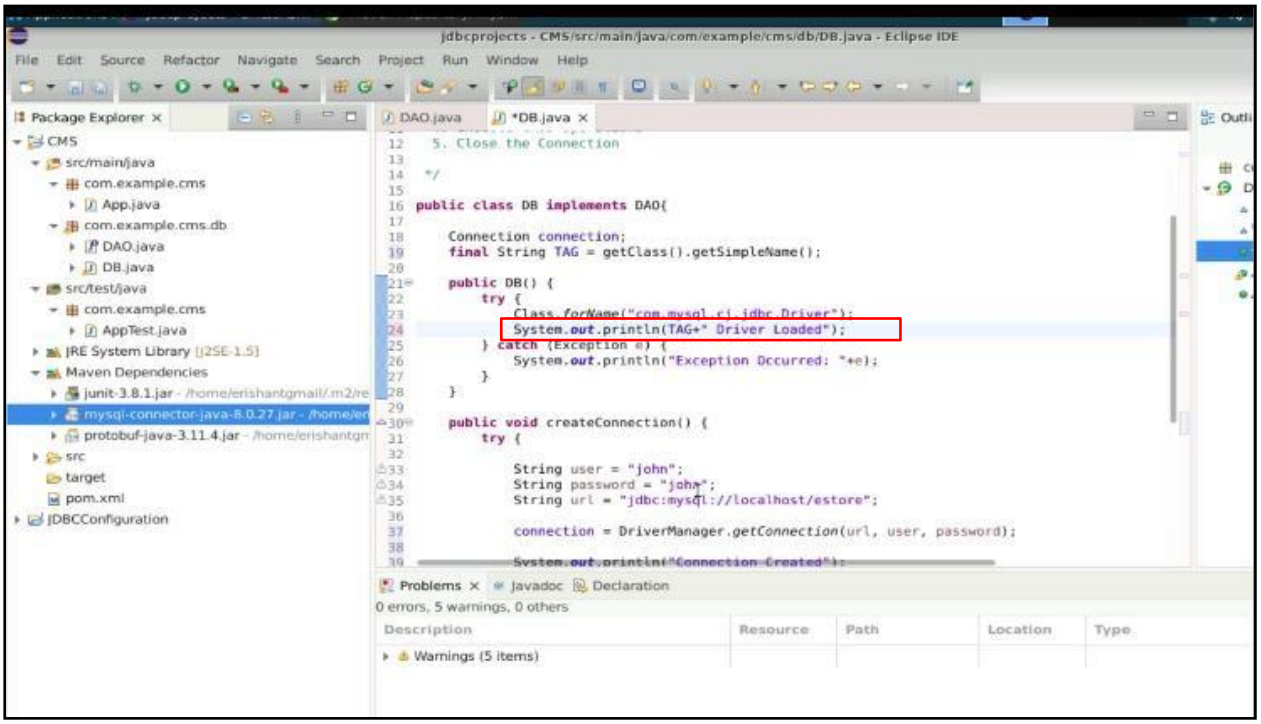
7.3 Initialize the connection as shown in the screenshot below:



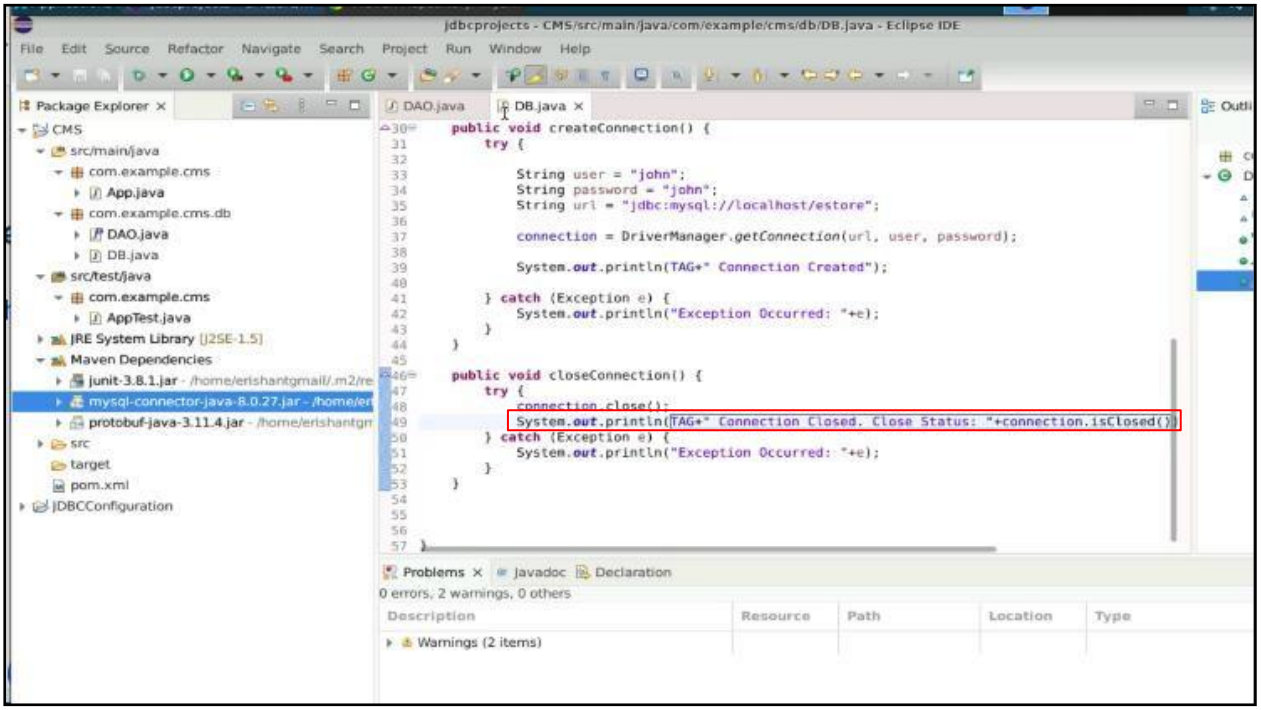
7.4 Add a print statement to show that the connection is created as shown in the screenshot below:



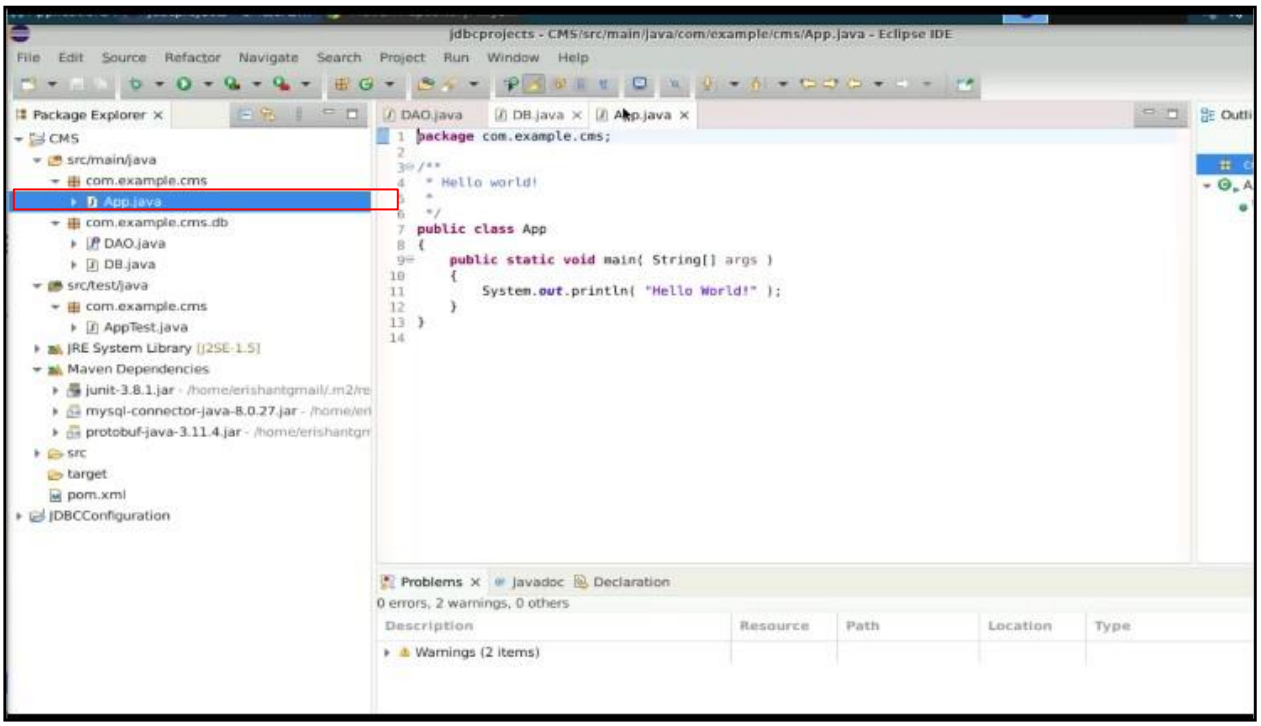
7.5 Type the final string tag as shown in the screenshot below:



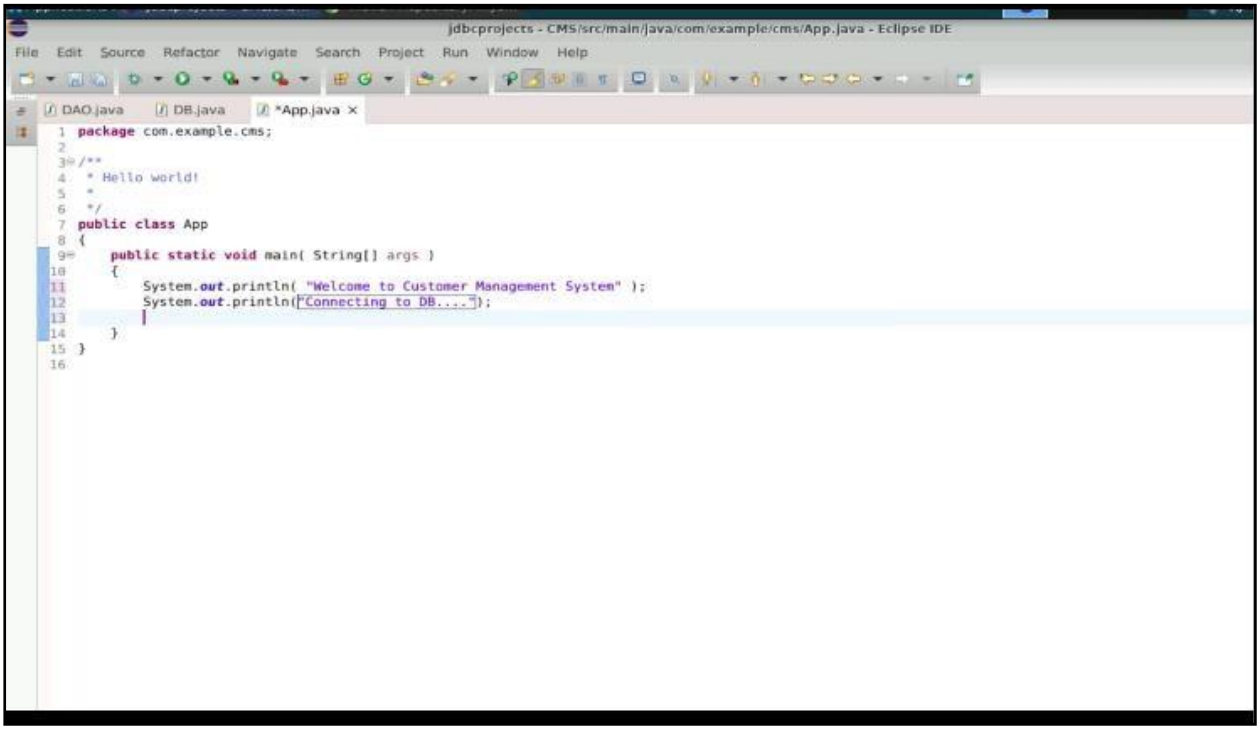
7.6 Execute the **closeConnection** method as shown in the screenshot below:



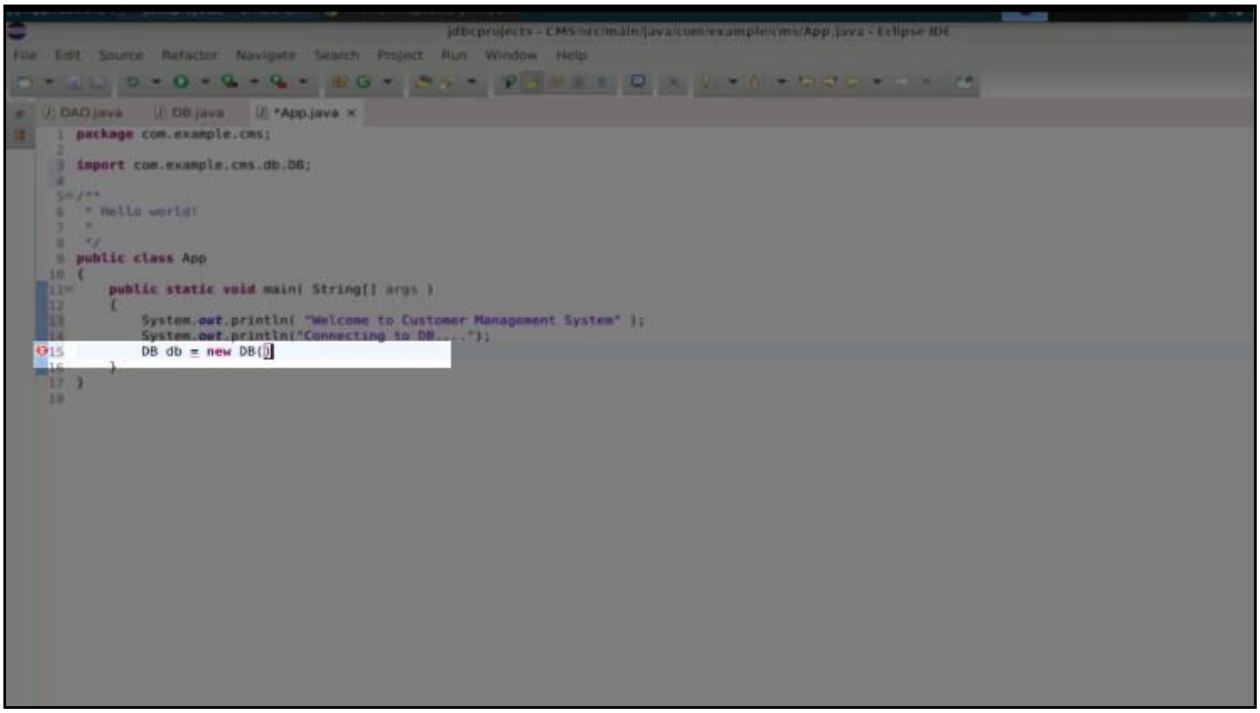
7.7 Open the **App.java** file from the left section as shown in the screenshot below:



7.8 Add a print statement in the **App.java** file as shown in the screenshot below:

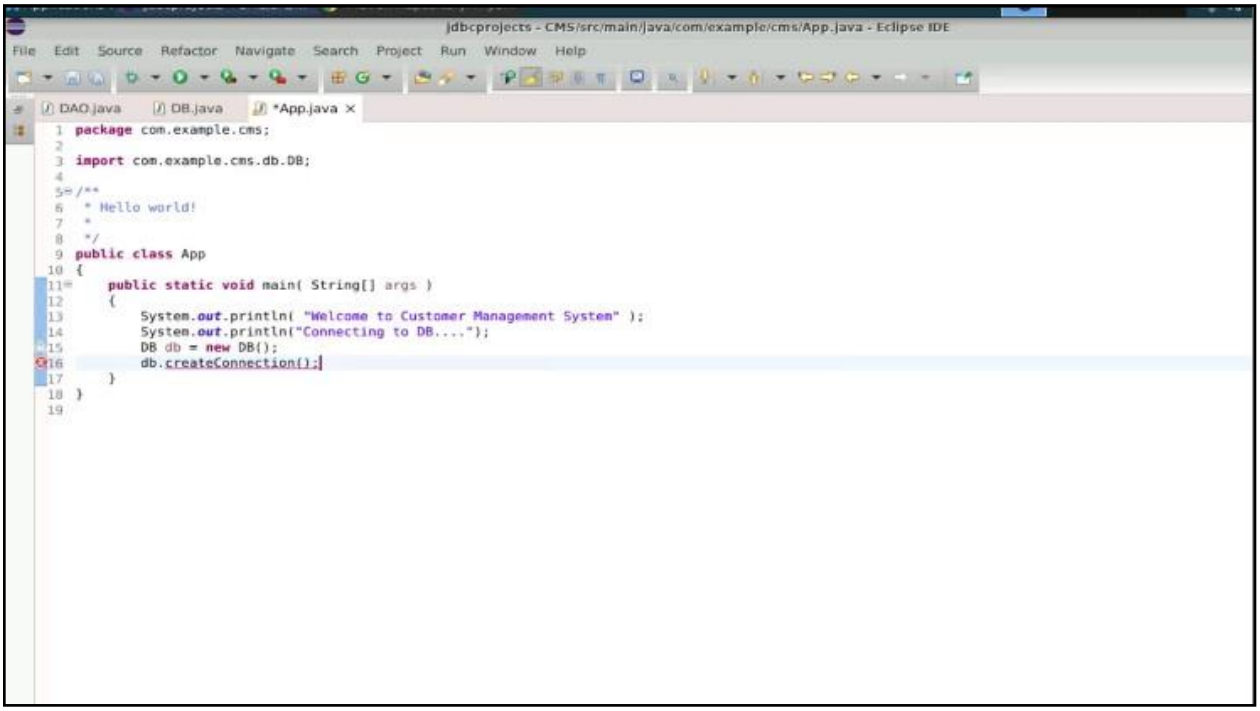


7.9 Use the **DB** database as shown in the screenshot below:

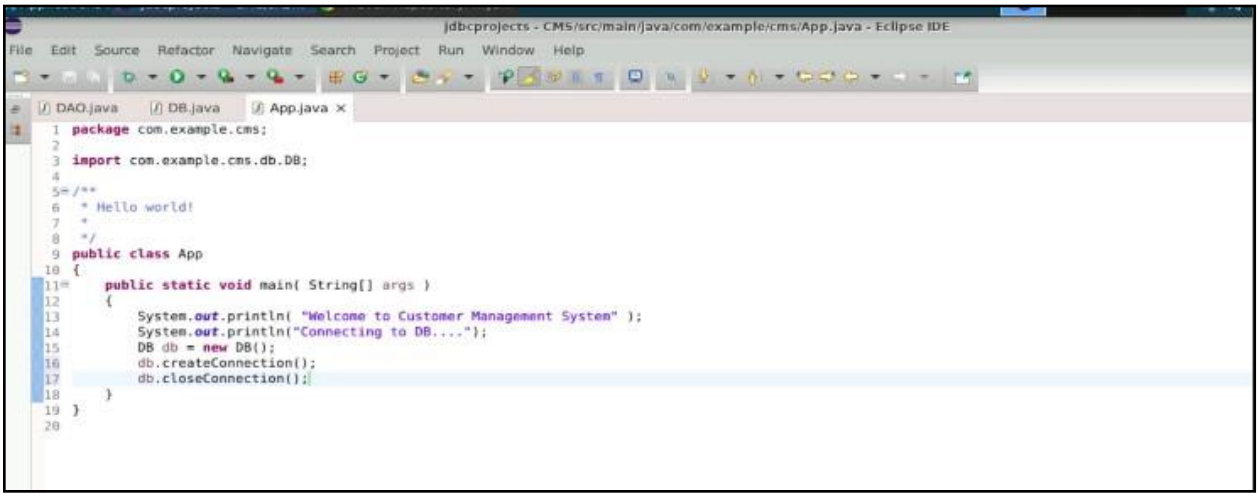


Step 8: Execute the createConnection method

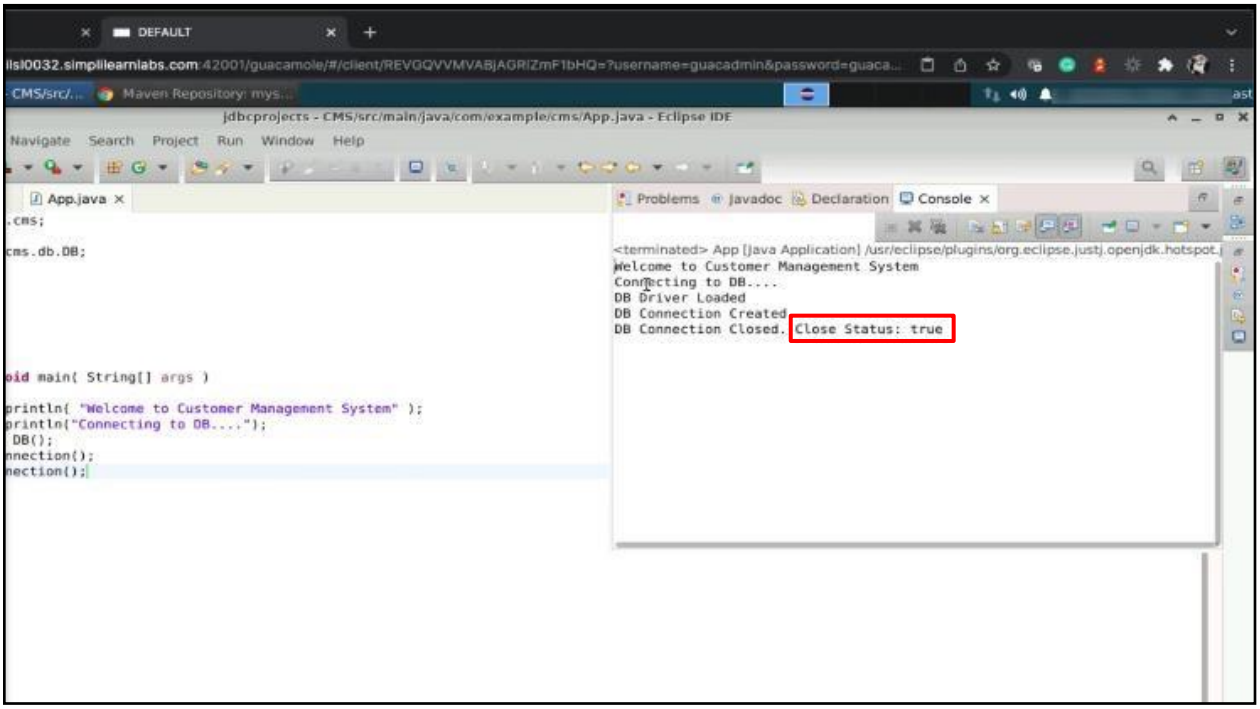
8.1 Execute the **createConnection()** method



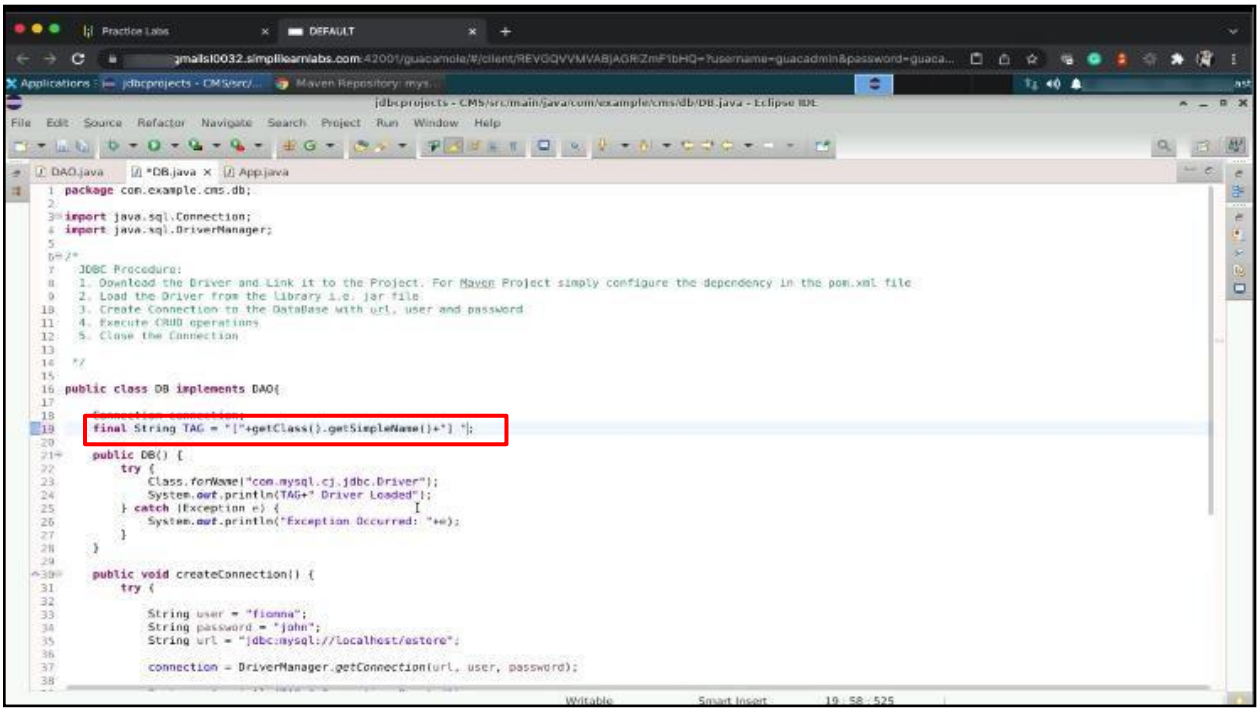
8.2 Execute **closeConnection()**



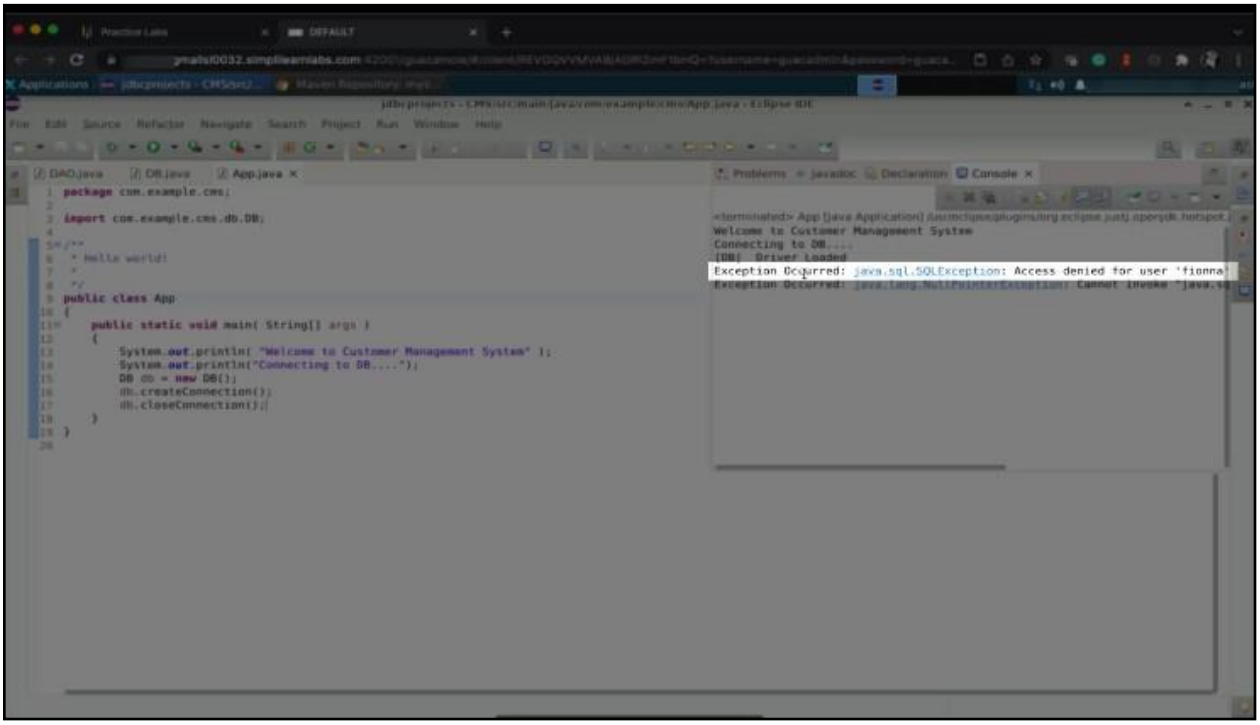
8.3 Run the code, and you should see the output as **Close Status: true** as shown in the screenshot below:



8.4 Change the username, and add a tag as shown in the screenshot below:



8.5 Re-run the code



By following these steps, you have successfully connected a Java project to a MySQL database by configuring JDBC and creating a connection using the **createConnection()** method for database operations.