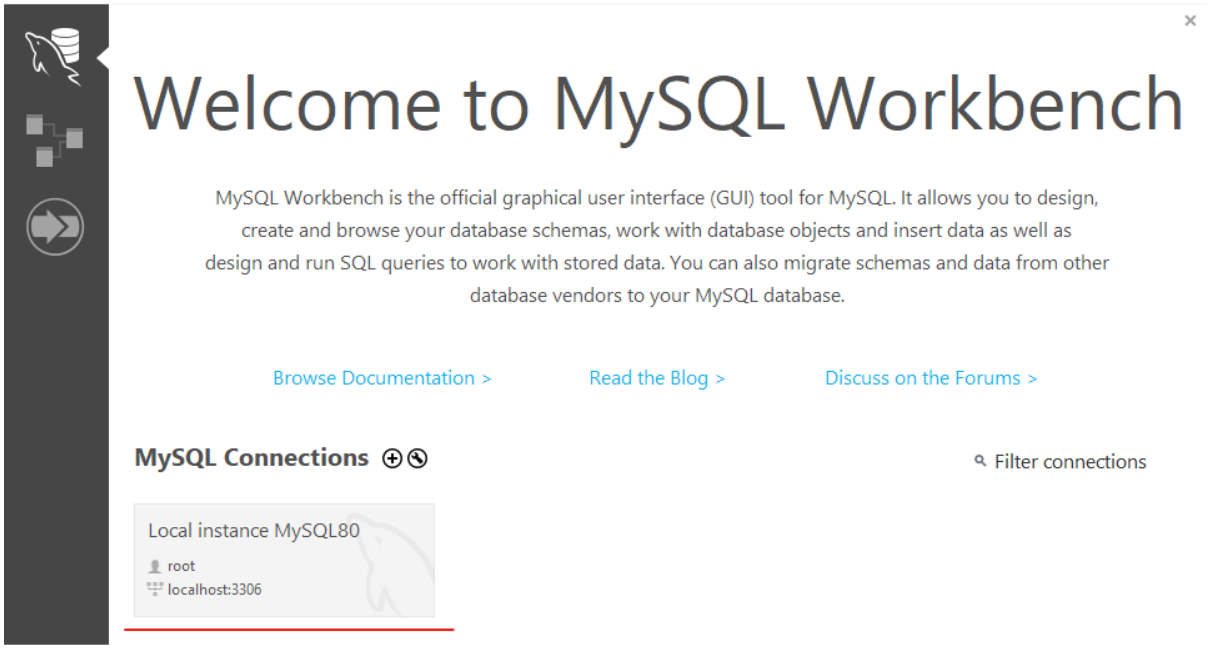
Lab 3

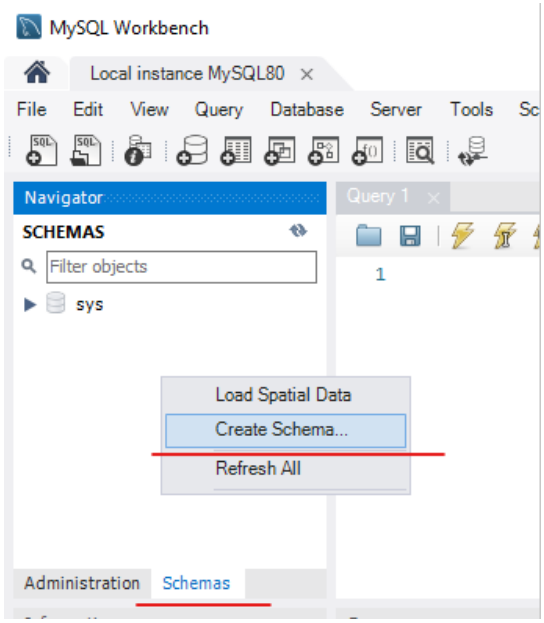
Part A: JDBC

For this exercise, we will be using the MySQL database.

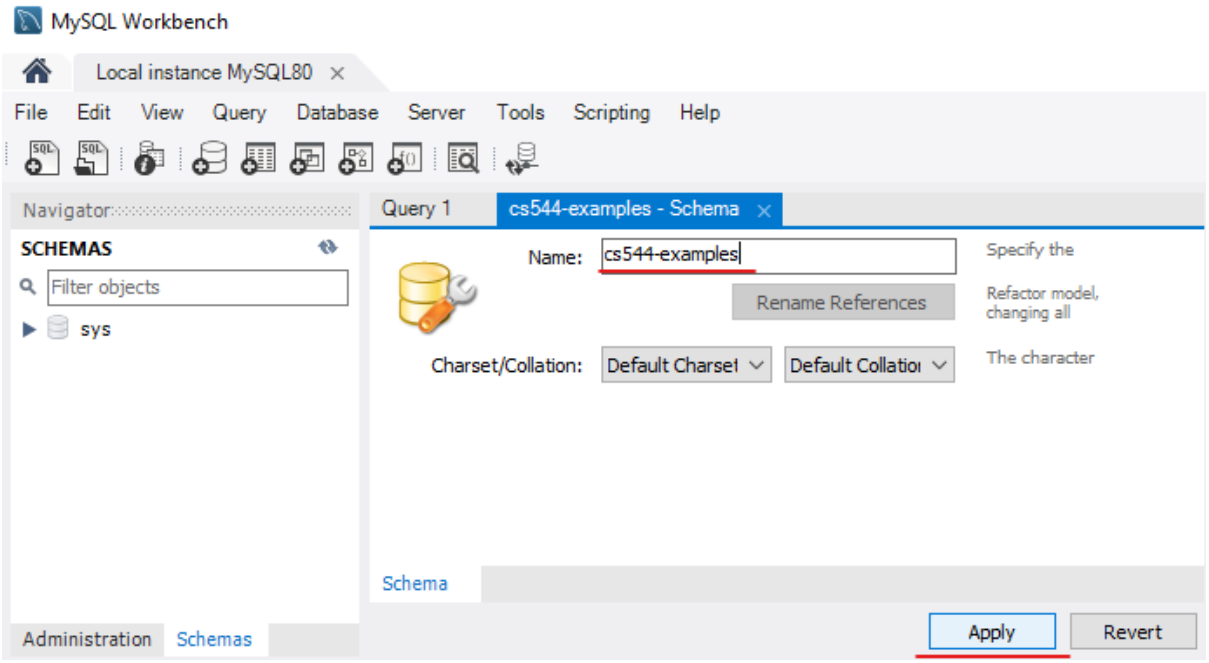
You can start the MySQL database by clicking on its Local instance:



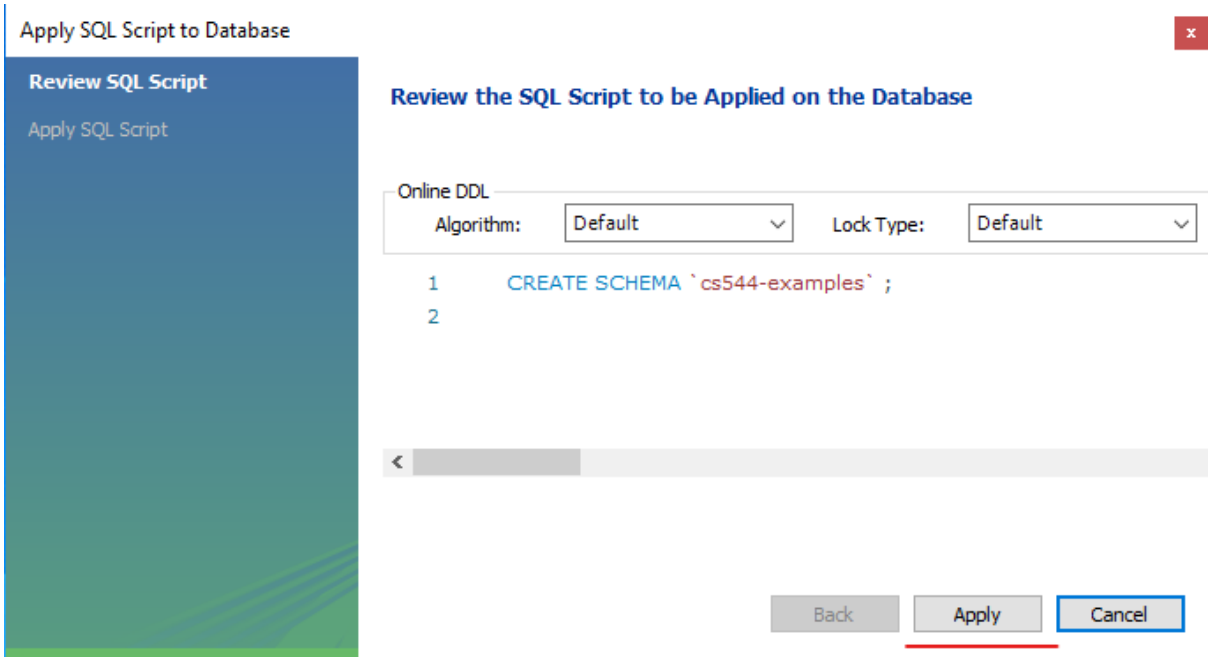
Once inside in the Local instance, click on **Schemas** and then right-click to display its menu and click on **Create Schema…**



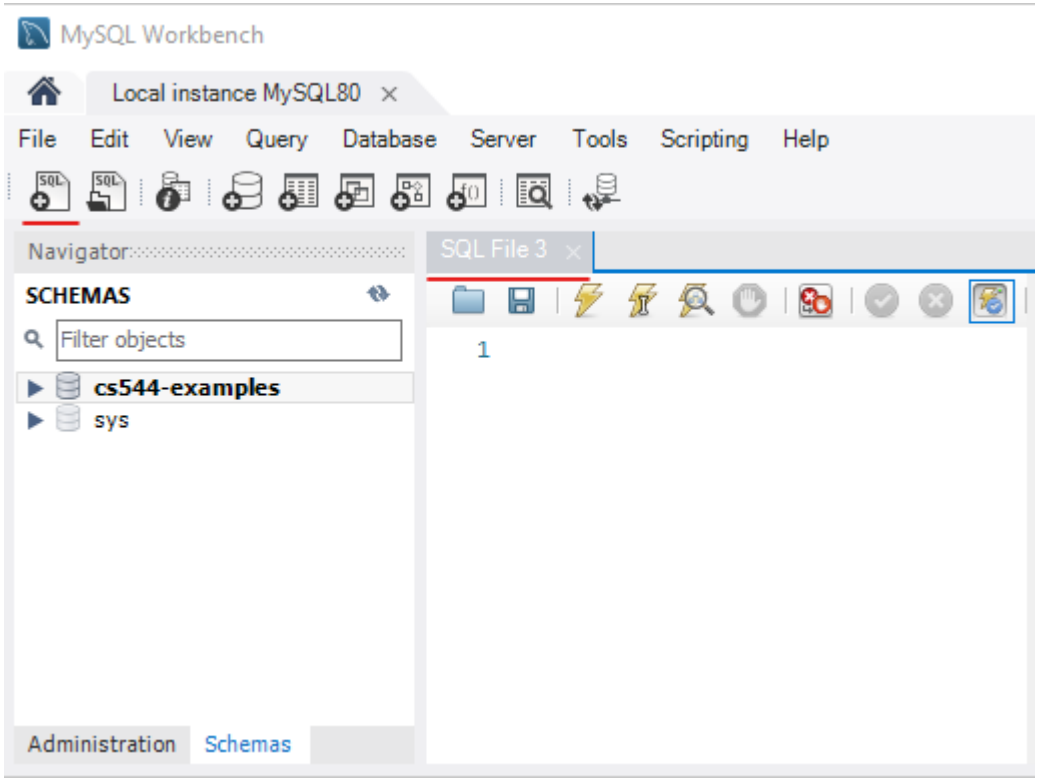
In the next window, you can now create your database schema. You can type any name. In this case, I am using the name we have for the example given in the demo source code:



On the next window, just click on **Apply** and you will create your new database schema:

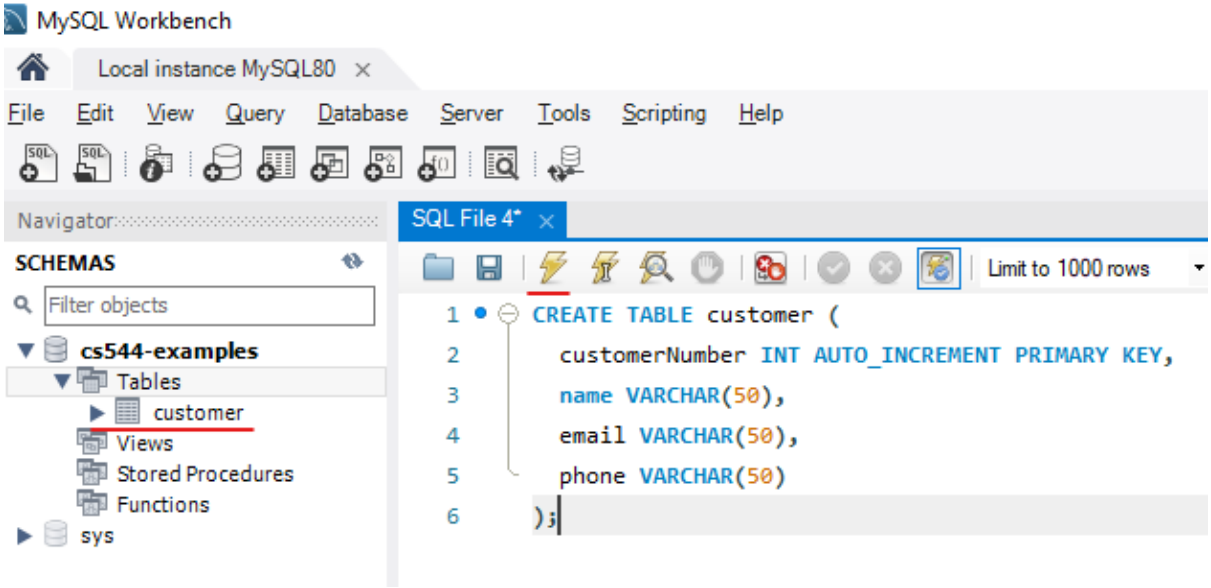


You should now be able to see your database schema in the Schemas section. Now we to open a new SQL Query clicking on its icon and a new query window will be shown:

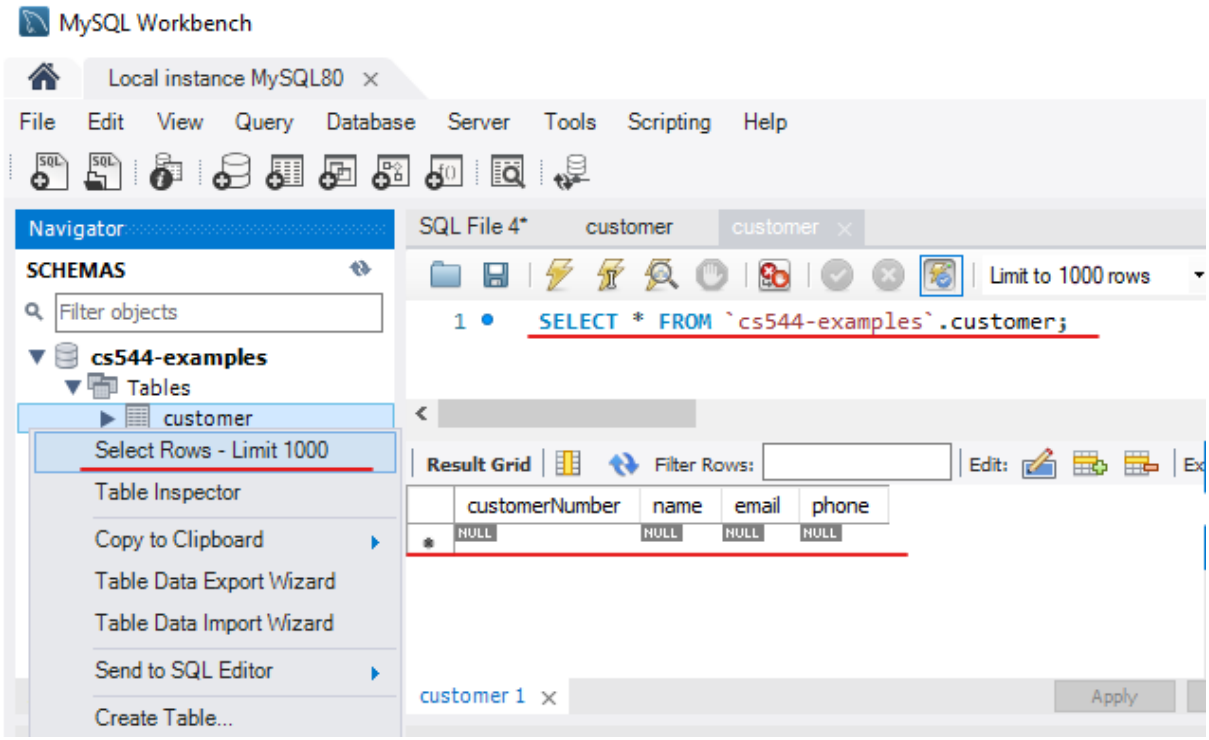


Next step is adding the SQL statement to create a new customer table:

```
CREATE TABLE customer (
  customerNumber INT AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(50),
  email VARCHAR(50),
  phone VARCHAR(50)
);
```



You should be able to see your new customer table. If you make a right-click on the customer table, you will see the option **Select Rows – Limit 1000**. Click here. You will see then all the records on your customer table. In this case is empty because there are not records yet.



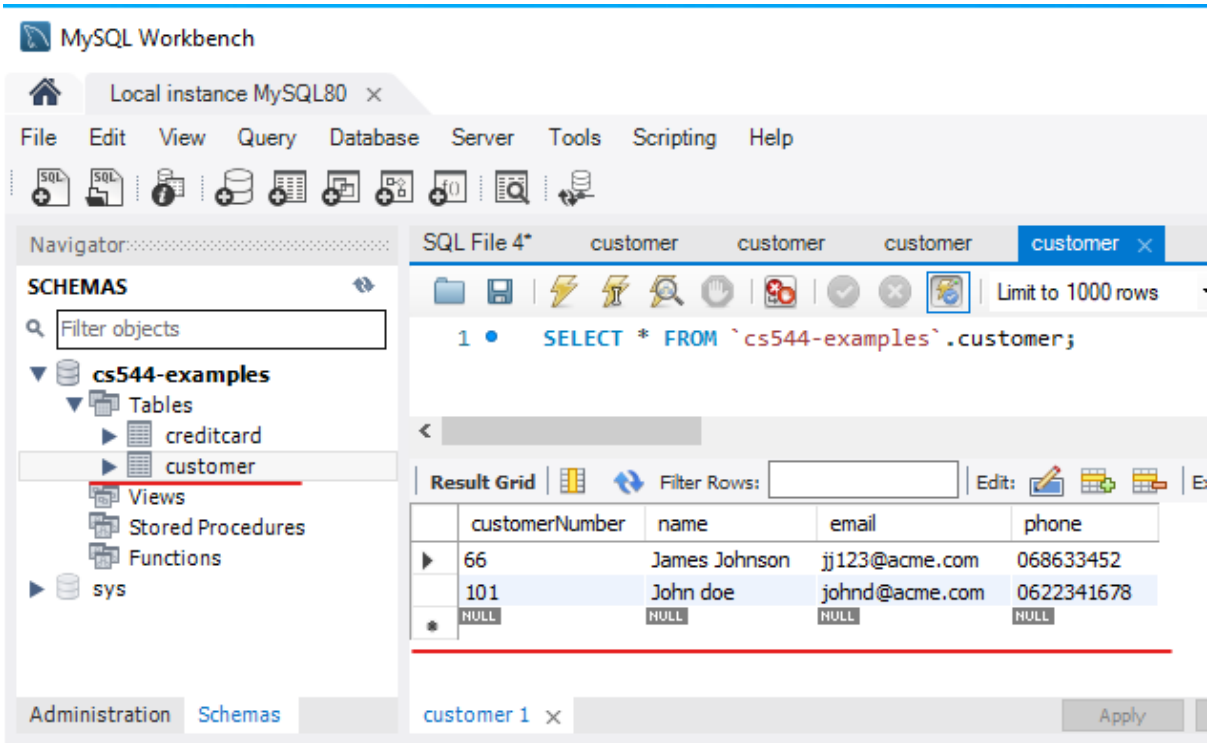
In the same way create the creditcard table:

```
CREATE TABLE creditcard (
  cardNumber VARCHAR(50) PRIMARY KEY,
  type VARCHAR(50),
  validationDate VARCHAR(50),
  customernumber INTEGER
);
```

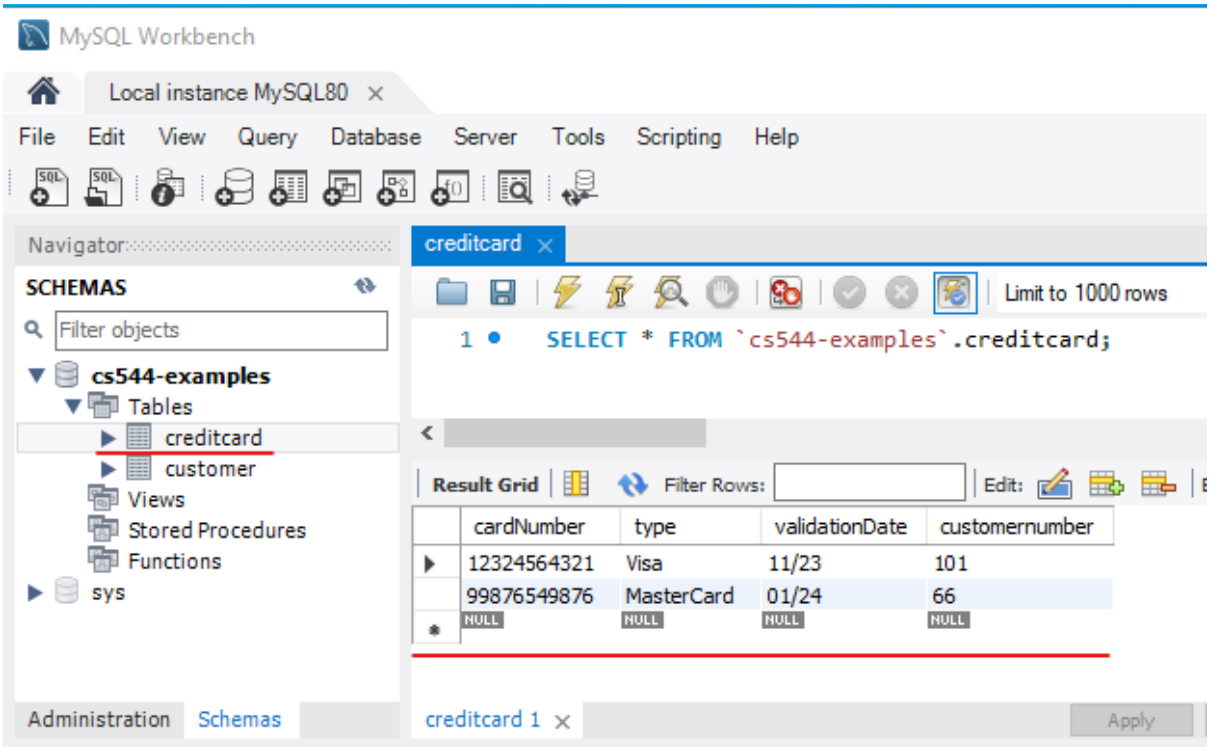
Now open the given project **lesson03-spring-boot-jdbc-customer-with-credit-card** that is given in the demo code into your favorite Java IDE and run the file Application.java.

The code inserts 2 customers with a corresponding credit card in the database.

Customer table:



Credit Card table:



Check the database.

Study the code and now write the Java code to save products in the database.

A product has the following attributes: productNumber, name, price

Make sure to create a new table in the database first.

Write a ProductDAO with the following methods:

save()	Saves a product in the database
findByProductNumber()	Return the product with this productNumber
getAllProducts()	Return the list with all products
findByProductName()	Return the list with products with this name
removeProduct	Remove the product with the given productNumber

In your application test if your methods work.

Now add a new class Supplier with the attributes name and phone. Assume a 1 to 1 relation between product and supplier.

Modify your code so that every product also has a supplier. Modify all the methods in the ProductDAO so that they work properly with the new Supplier class.

Part B: JPA

Open the given application **lesson03-spring-jpa-customer** that is given in the demo code. The application inserts a number of customers in the database, and then performs some queries on the customers in the database.

In the same project, create the following entity class

```
public class Book {
	private int id;
	private String title;
	private String ISBN;
	private String author;
	private double price;
}
```

Annotate the class with the correct JPA annotations, and write a Book repository.

Save some books, and retrieve all books from the database.

- Create 3 books, and save them in the database
- Retrieve all books from the database and display them in the console
- Update a book
- Delete a book (not the book that was just updated)
- Retrieve all books from the database again and display them in the console

What to hand in:

1. A separate zip file with the solution of part A
2. A separate zip file with the solution of part B