|  |
| --- |
| CPU with binary numbers and blueprint |
| Structured Query Language - SQL  Assignment 2 |
| |  |  |  | | --- | --- | --- | | Mohamed AHMED | 24-10-2023 | Just IT – DT Bootcamp | |

Contents

[**Task 1 - Types of Relationship in Relationship databases with examples** 2](#_Toc149098398)

[**Many-to-Many Relationship in a Database** 3](#_Toc149098399)

[**Task 2 – Normalization and its importance to database development** 4](#_Toc149098400)

[**The World Database** 6](#_Toc149098401)

[I **downloaded the world database from team >> Assignment** 6](#_Toc149098402)

[**SQL Queries** 9](#_Toc149098403)

[Task 3 – Count Statement 9](#_Toc149098404)

[Task 4 – Population & life expectancy of Argentina 9](#_Toc149098405)

[Task 5 – Highest Life expectancy by Country 10](#_Toc149098406)

[Task 6 – 25 countries starting with the letter (F) 11](#_Toc149098407)

[Task 7 – First 10 rows in the city table 11](#_Toc149098408)

[Task8 – Cities with population of more than 200000 12](#_Toc149098409)

[Task 9 – All city names with the prefix of (Be) 13](#_Toc149098410)

[Task 10 – Cities with population of between 50000 and 100000 14](#_Toc149098411)

[Task 11 – The city with the lowest population 15](#_Toc149098412)

[Task12 – Switzerland’s population and language spoken 16](#_Toc149098413)

[**EER Diagram** 18](#_Toc149098414)

[Task 13 – Creating an EER Diagram 18](#_Toc149098415)

[Task 14 – Primary Keys and Foreign Keys 20](#_Toc149098416)

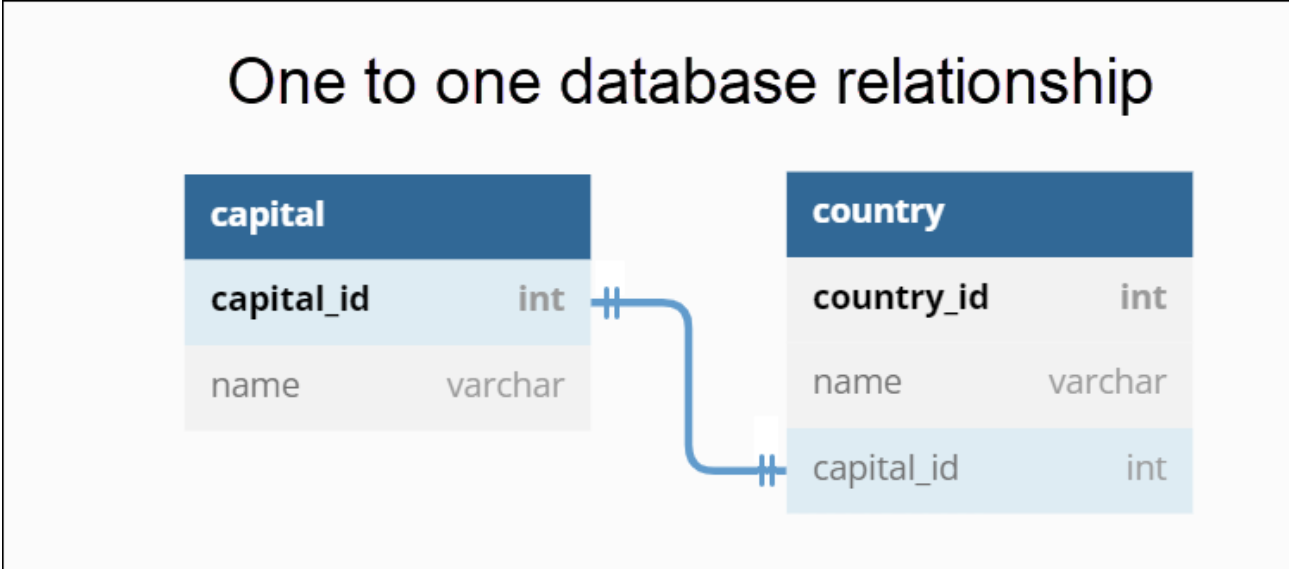
[**Reflection** 21](#_Toc149098417)

# **Task 1 - Types of Relationship in Relationship databases with examples**

**One-to-One Relationship in a Database**

A one-to-one relationship (1:1) in a database has one record on each side of the relationship. Every primary key relates to at most one entry from another table, making the foreign key unique.

A simple example to demonstrate a one-to-one relationship is with capital cities. One country has only one capital city, and one capital city belongs to only one country.



Another Example is a database for employee information where each employee is associated with one unique employee ID, and there is another table for employee contact information, where each employee has a single unique set of contact details.

**One-to-Many Relationship in a Database**

A one-to-many (1-N) relationship in a database has a single entry on one side and multiple entries on the other side of the relationship Every primary key corresponds to one or more records from another table. In this case, the foreign key is not unique.

An example of a one-to-many relationship is the connection between a driver and cars. A driver (owner) can have many cars, but every car belongs to one driver (owner) only.

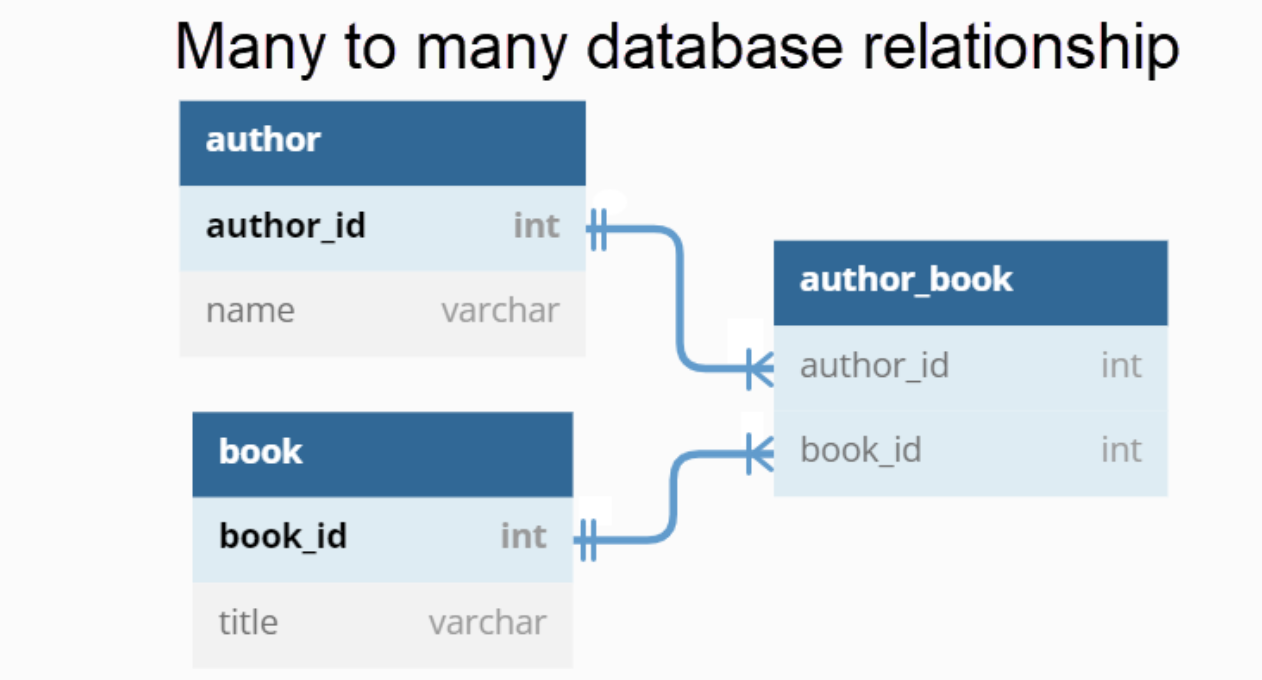
A database containing two tables with information about drivers and cars connects using a primary key. When added to the car table, the unique ID from a driver becomes a foreign key. Different cars can have the same driver (owner).

Another example is a library database, one library can have many books, 1 library can have many books, but each book is available in only one library, 1 book is belongs to 1 library.

### **Many-to-Many Relationship in a Database**

Many-to-many (N: N) relationships in a database have multiple entries on both ends of the relationship. Since numerous entries may exist on both ends, a standard solution is to create an association (junction, join) table with foreign keys from both tables.

An example of many-to-many relationship, a single book can have multiple authors. Likewise, a single author can have numerous books. If there is a table containing books and another with authors, the best way to create the relationship between the two is through a new table. The new table (a bridging table) has foreign keys from both parent tables, creating a many-to-many relationship.



Another example of many to many relationship is database for students and courses. Each student can enrol in multiple courses, and each course can have multiple students. To represent this, you might have a students table, a Courses table, and a third table Enrolment that links students to courses.

# **Task 2 – Normalization and its importance to database development**

**What is Database Normalization?**

Database normalization is a method in relational database design which helps properly organize data in tables. The process aims to create a system that represents information and relationships without data loss or redundancy.

Normalization is an **iterative process.** Commonly, normalizing a database occurs through a series of tests. Each subsequent step decomposes tables into more manageable information, making the overall database logical and easier to work with.

**The important of normalization for a database**

It eliminates attributes with multiple values, doubled or repeated attributes, non-descriptive attributes, attributes with redundant information and attributes created from other features.

Although total database normalization is not necessary, it provides a well-functioning information environment. The normalization of a database ensures that a database structure is suitable for generalized queries, it also minimised data redundancy, increasing memory efficiency on a database server. In addition, normalization maximises data integrity through the reduced insert, update, and delete anomalies. It also maximises data integrity through the reduced insert, update, and delete anomalies. Database normalization transforms overall database consistency, providing an efficient environment.

**Database Normalization Concepts**

The elementary concepts used in database normalization are:

Keys: Column attributes that identify a database record uniquely.

Functional Dependencies: Constraints between two attributes in a relation.

Normal Forms: Steps to accomplish a certain quality of a database.

**Database Normal Forms**

Normalizing a database is achieved through a set of rules known as normal forms. The main concept is to help creating a database that meets the desired quality of a relational database.

Database normalization is typically achieved by dividing data into separate tables, identifying the relationships between these tables, and ensuring that each table is in a specific normal form (e.g., First Normal Form, Second Normal Form, Third Normal Form). The process can involve decomposing larger tables into smaller, related tables and using foreign keys to establish relationships.

**First Normal Form 1NF**

To rework the database table into the 1NF, values within a single field must be atomic. All complex entities in the table divide into new rows or columns.

**Second Normal Form 2NF**

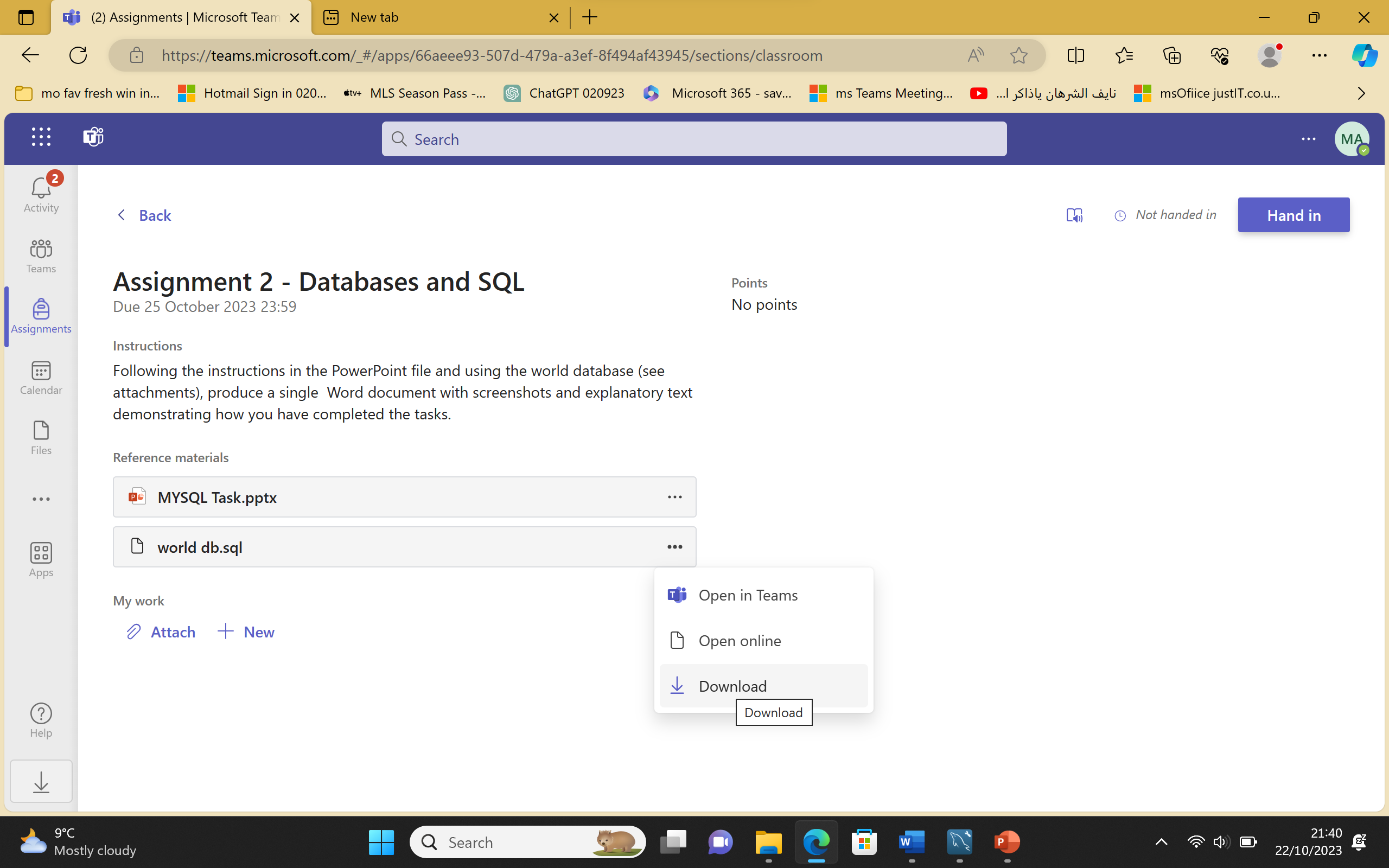
The second normal form in database normalization states that each row in the database table must depend on the primary key.

**Third Normal Form 3NF**

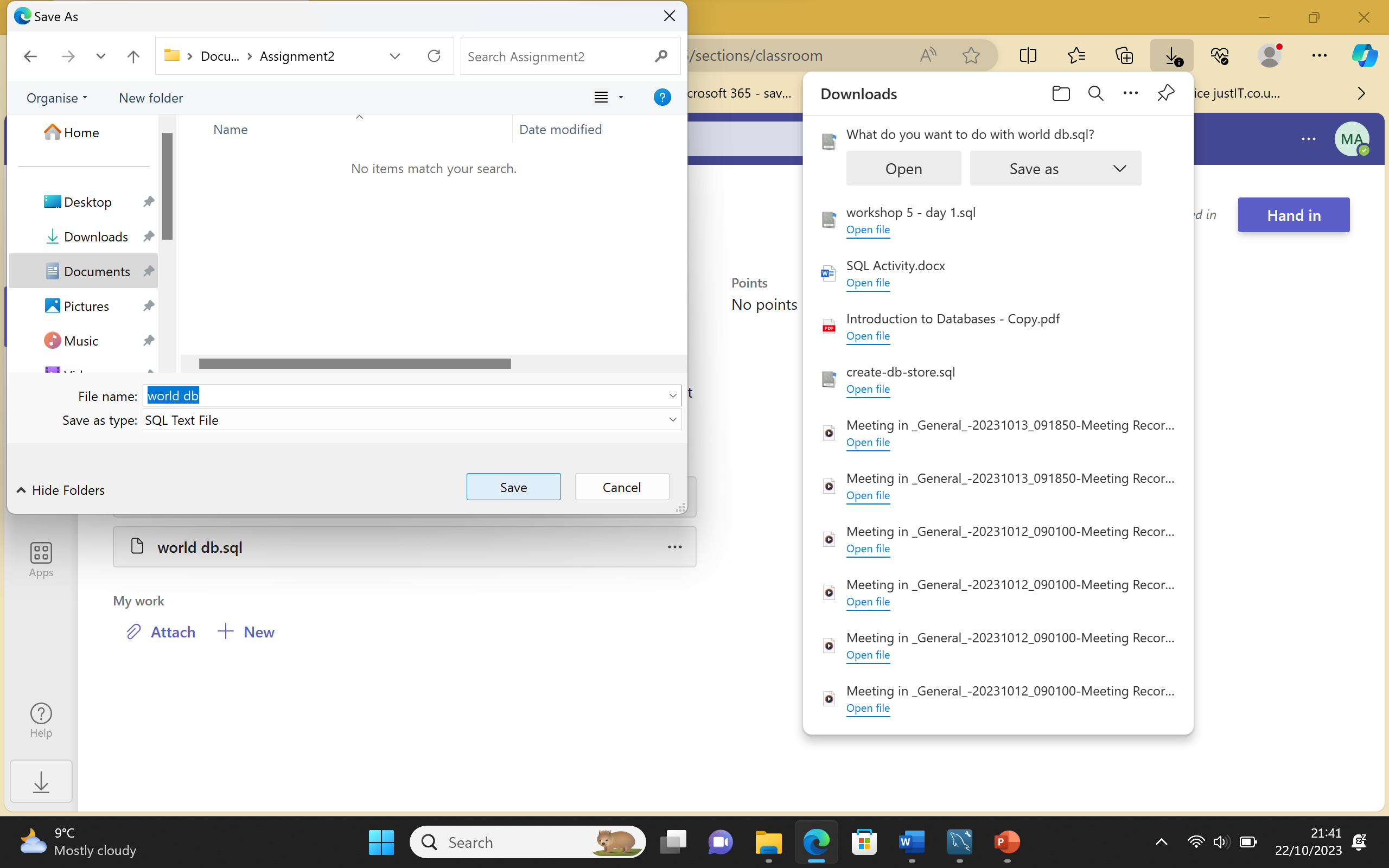
The third normal form decomposes any transitive functional dependencies. A database is normalized when it fulfils the **third normal form**.

# **The World Database**

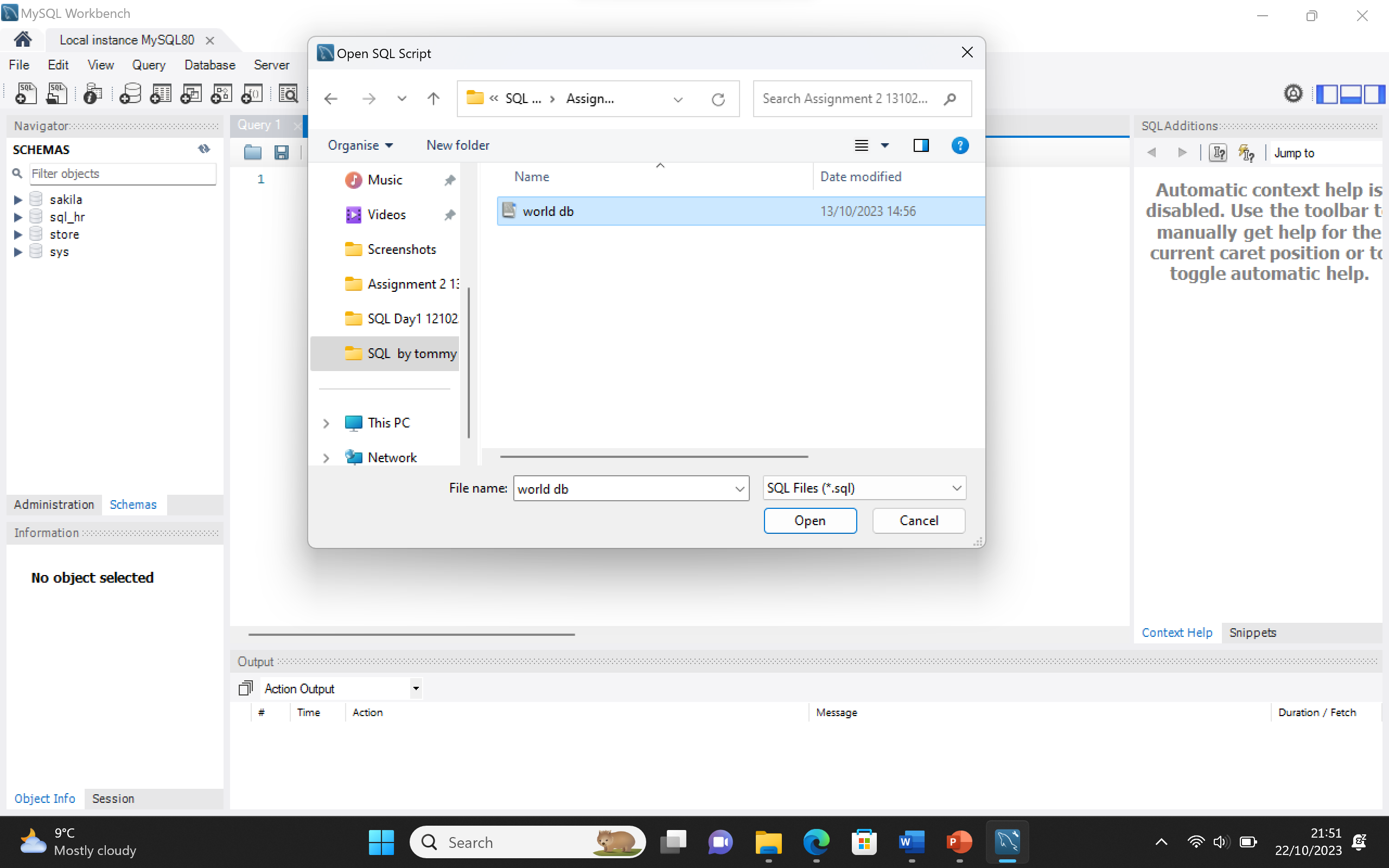
## I **downloaded the world database from team >> Assignment**



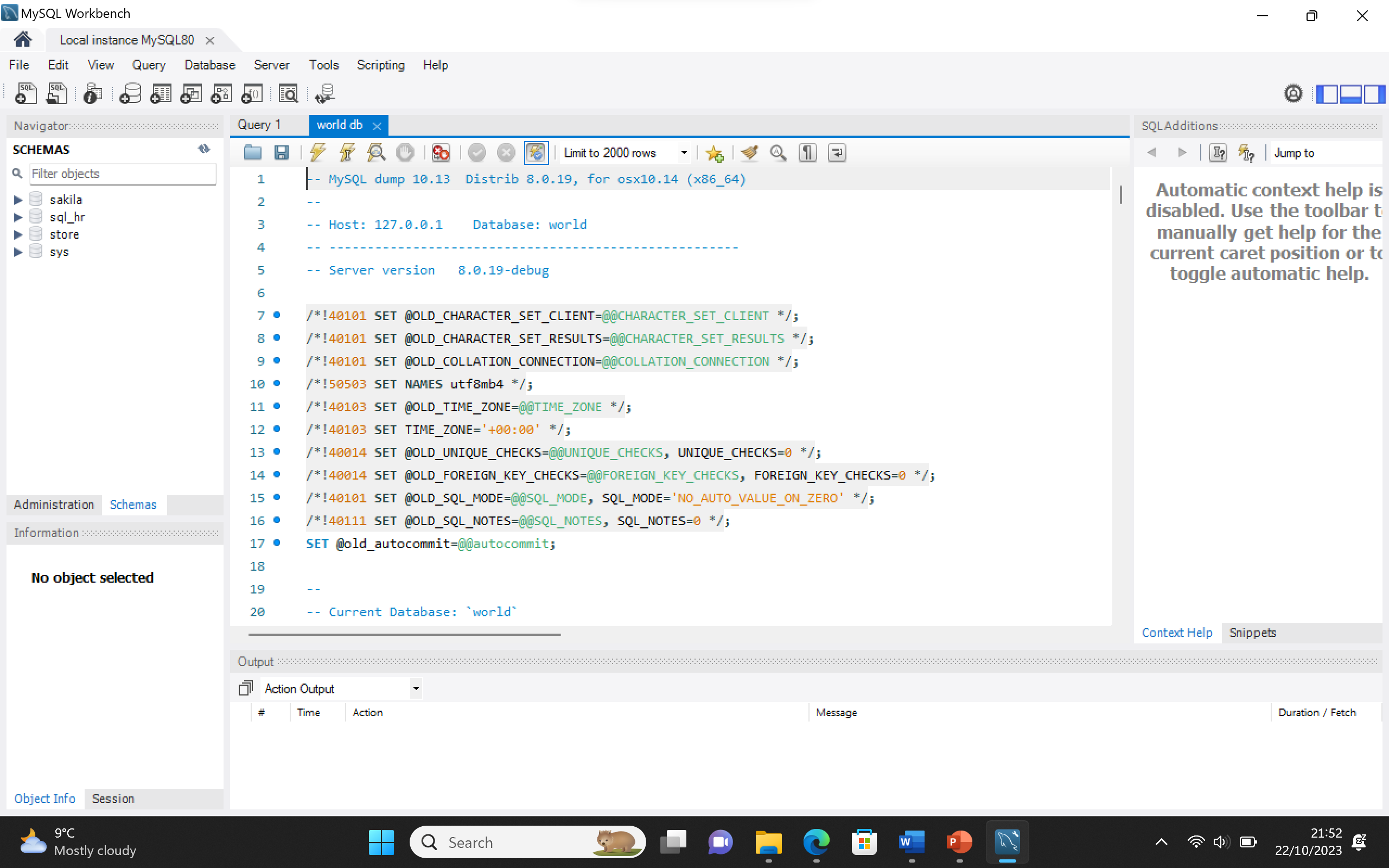
**I saved the world database on my computer**



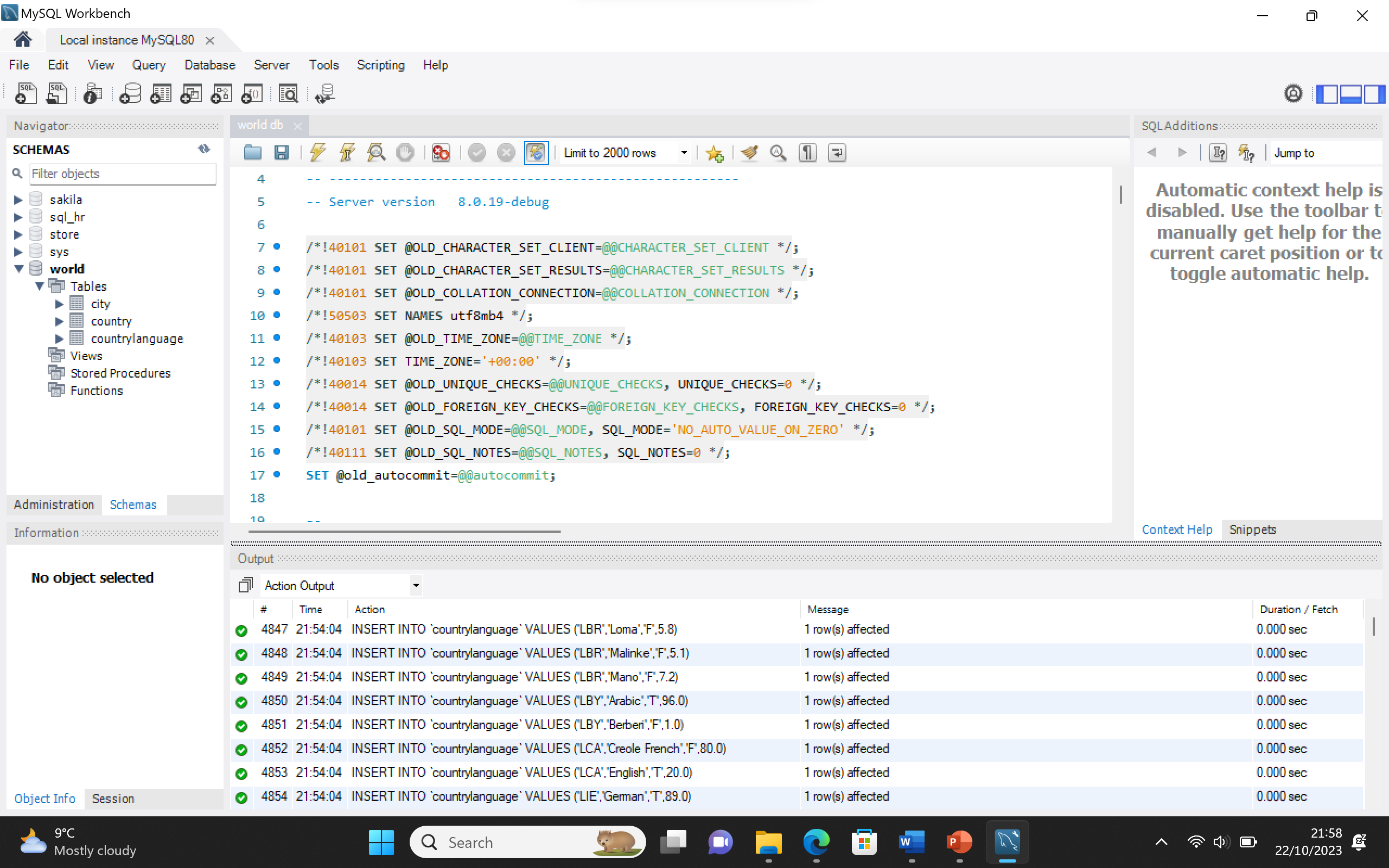
**I opened the world database in MYSQL workbench**



**I run the SQL script to create the world database**

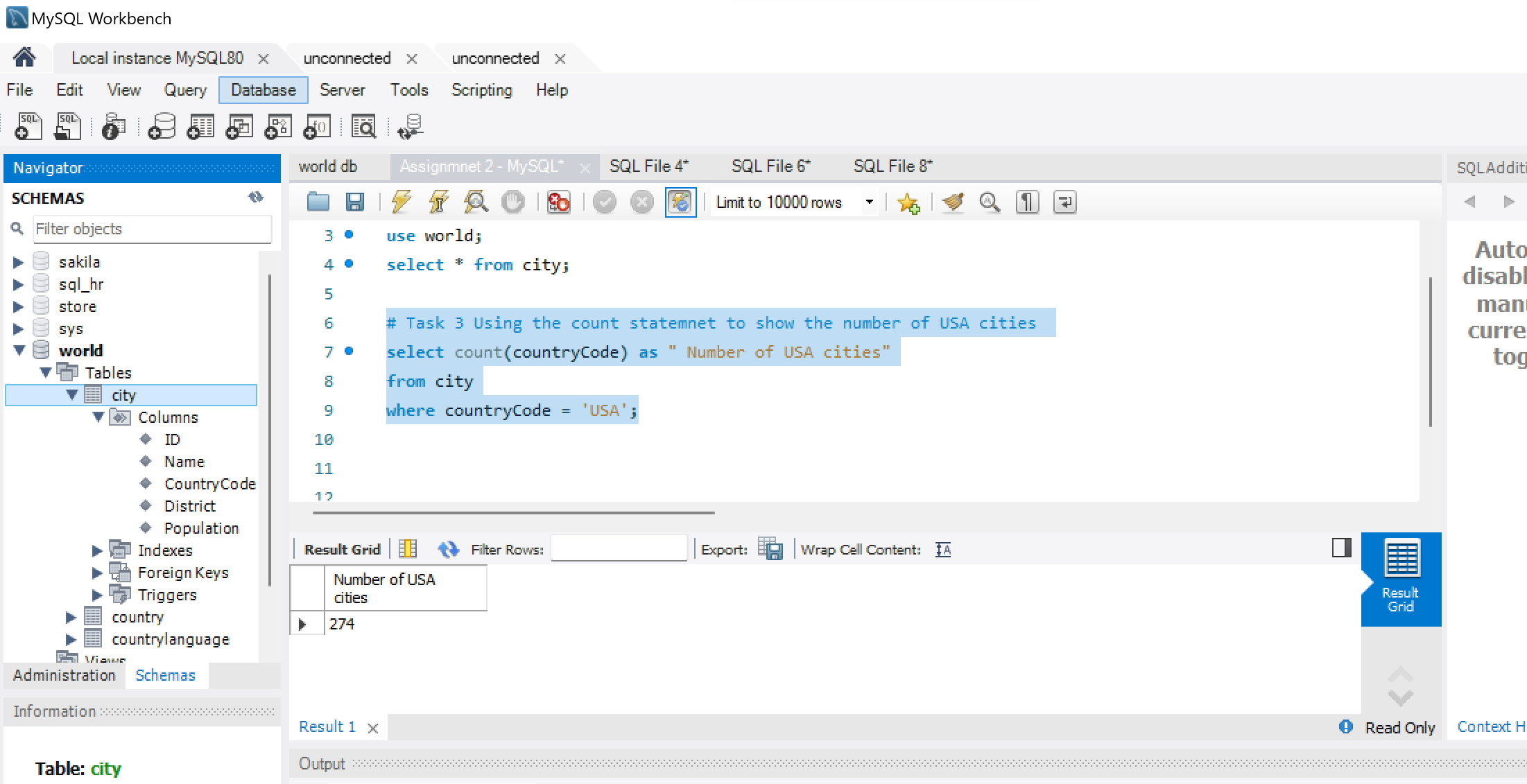


**As shown below the world database has been created and ready to create queries**

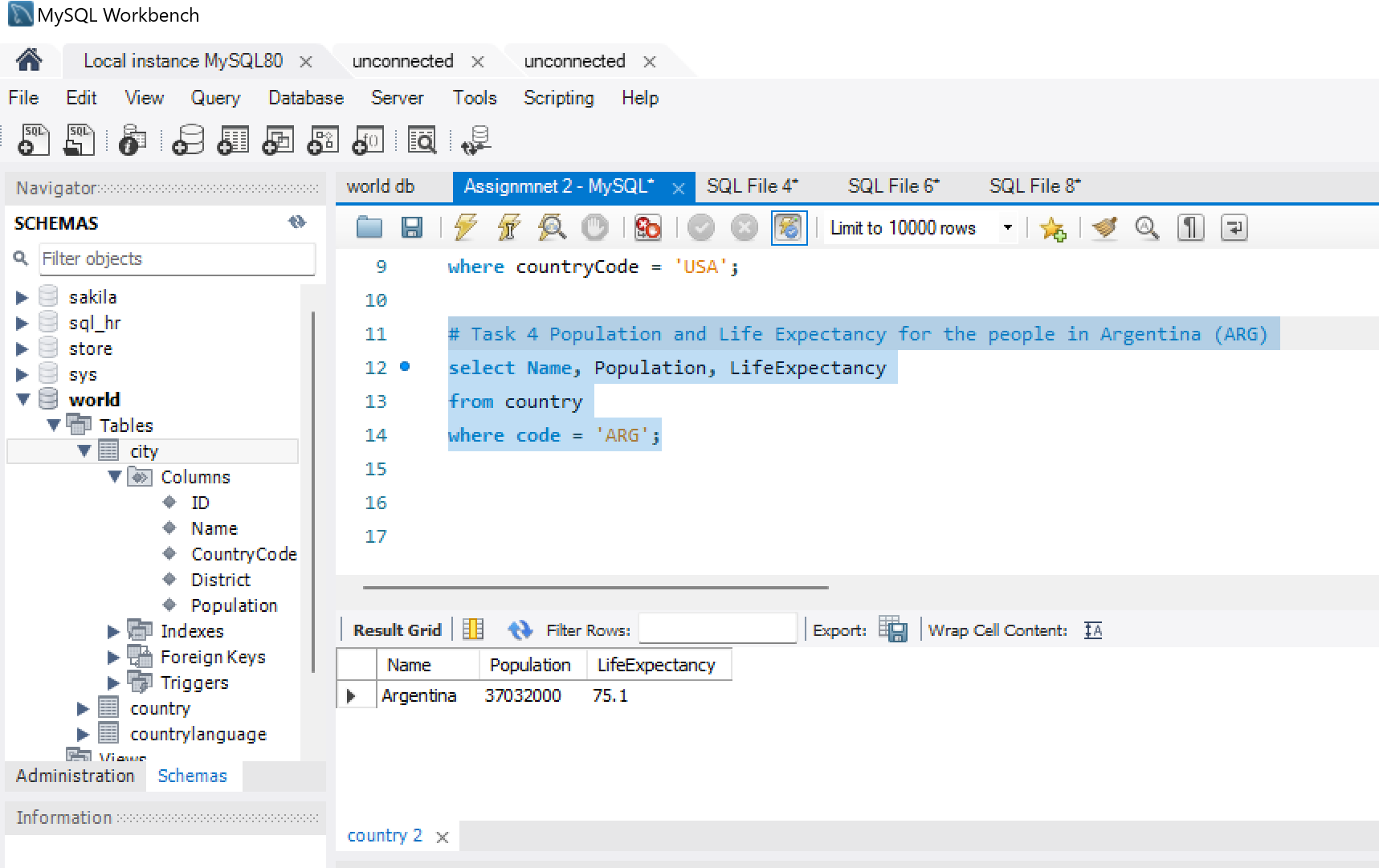


# **SQL Queries**

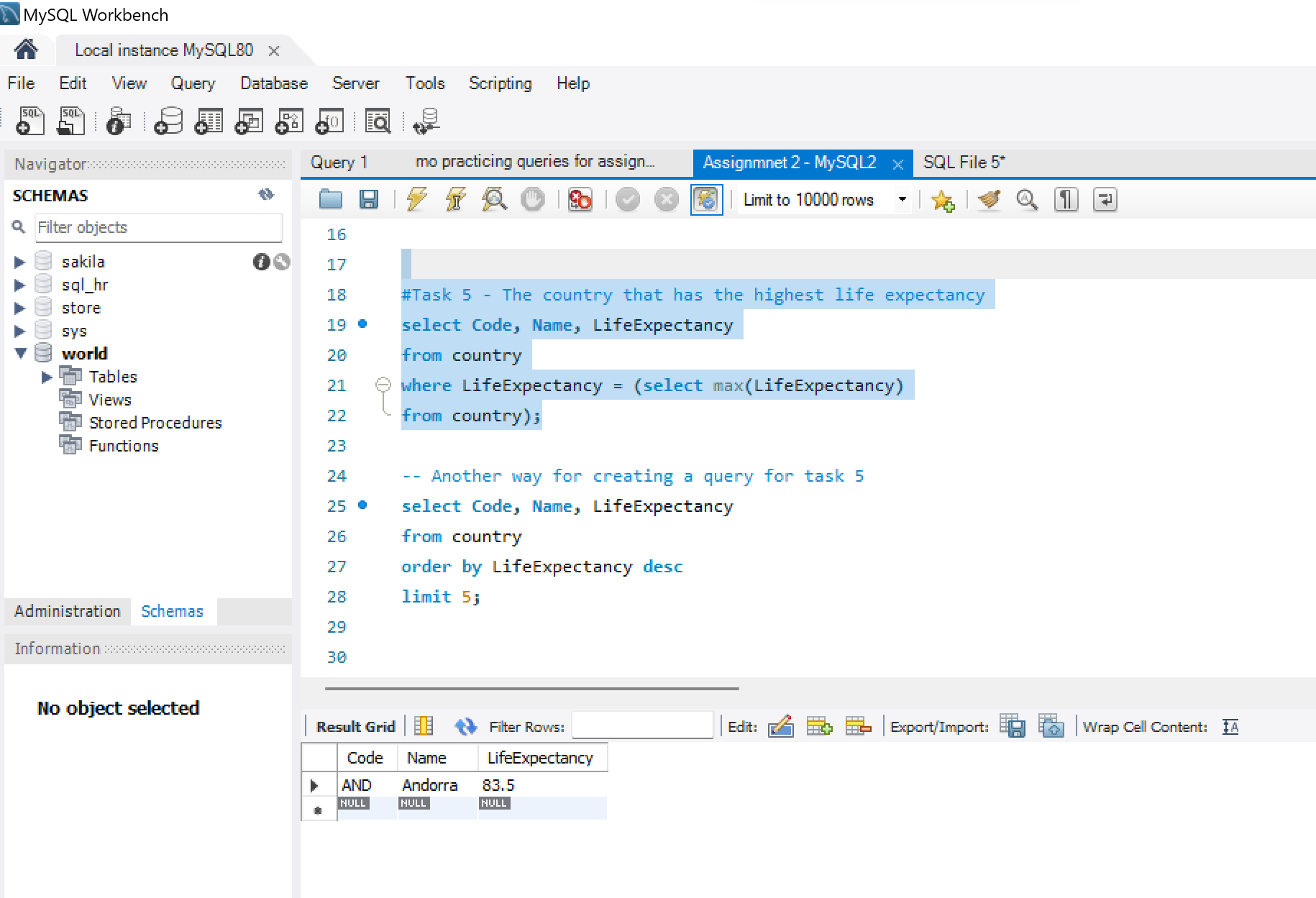
## Task 3 – Count Statement



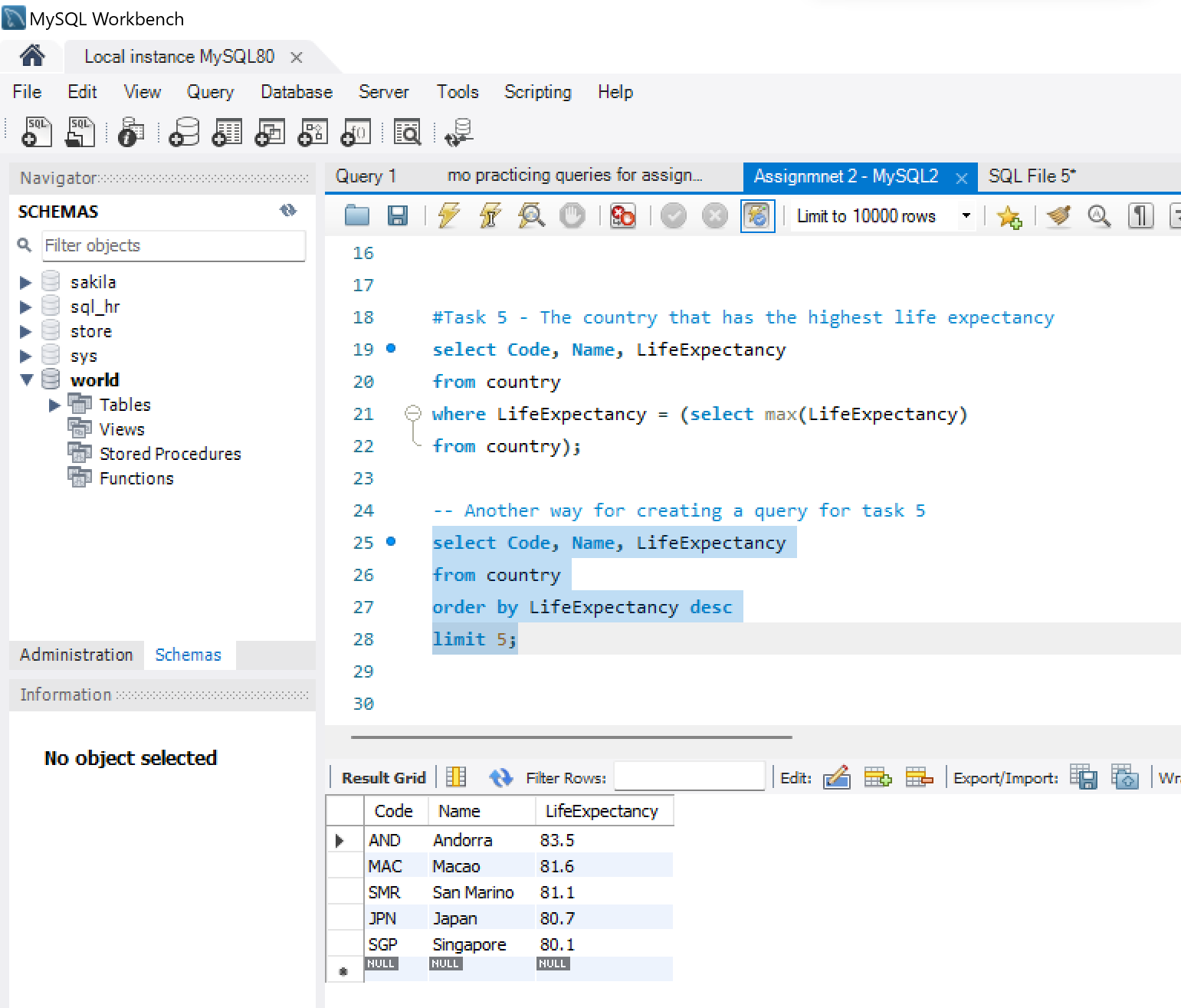
## Task 4 – Population & life expectancy of Argentina



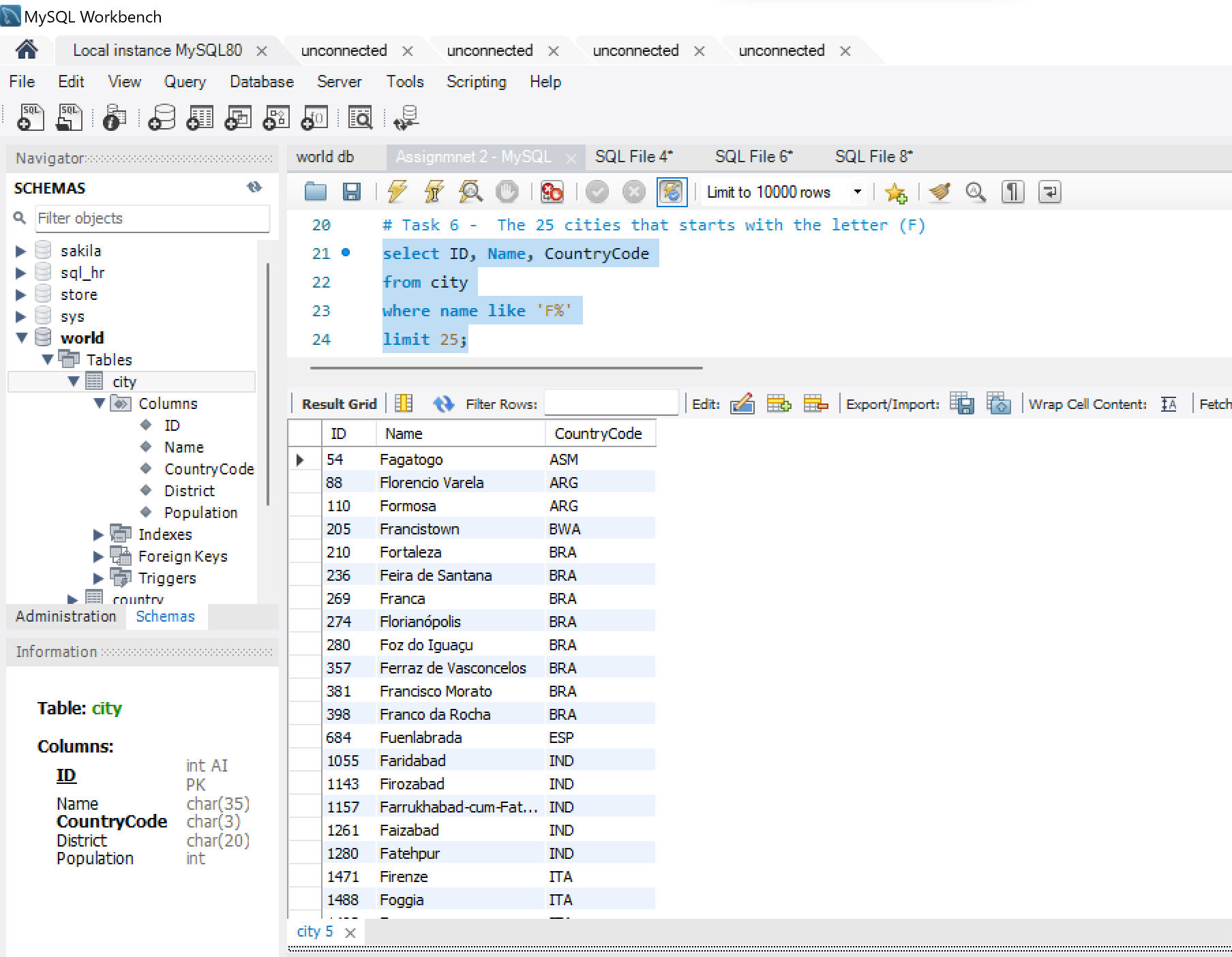
## Task 5 – Highest Life expectancy by Country



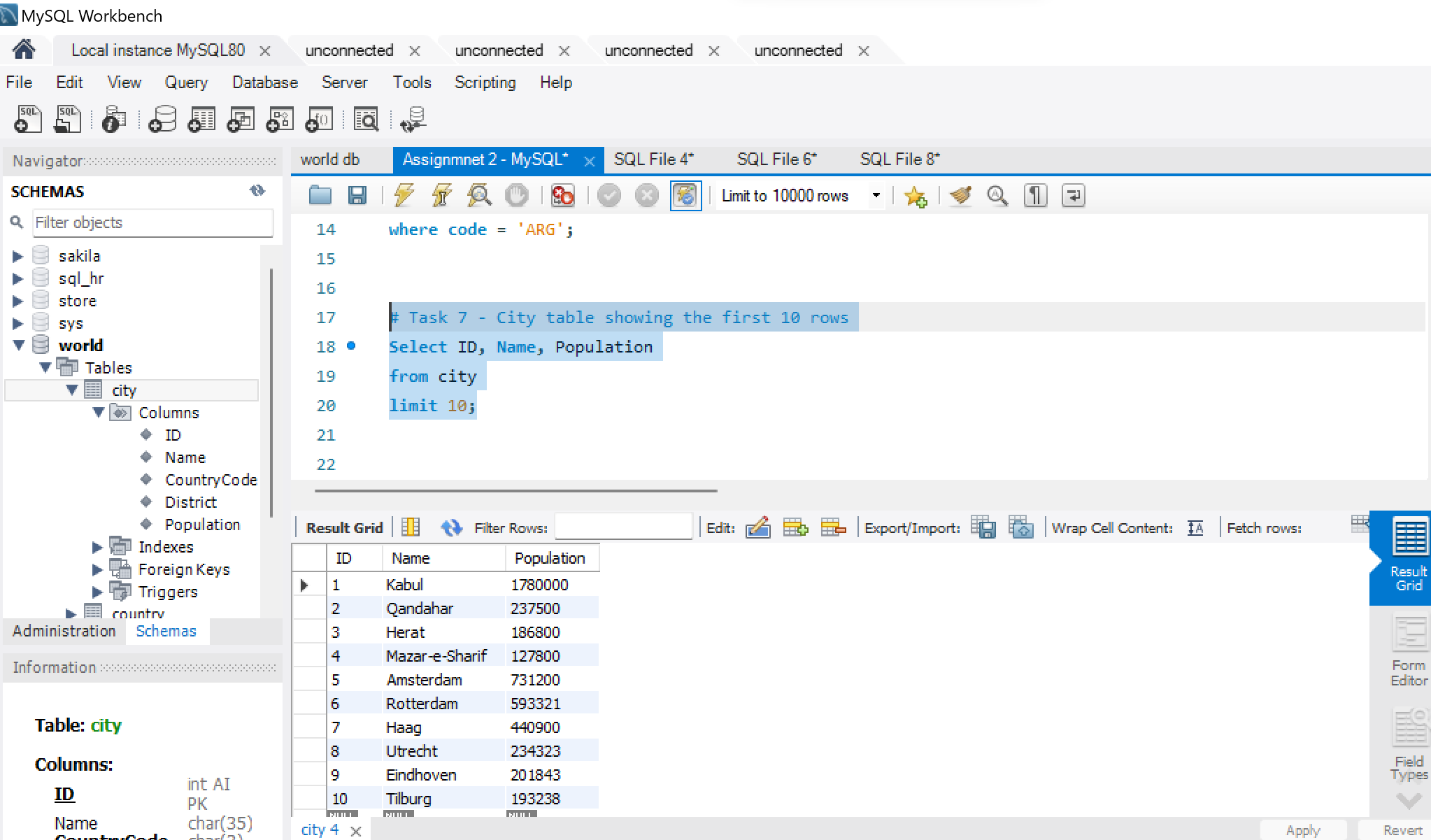
**Below is another way for creating a query for task 5**



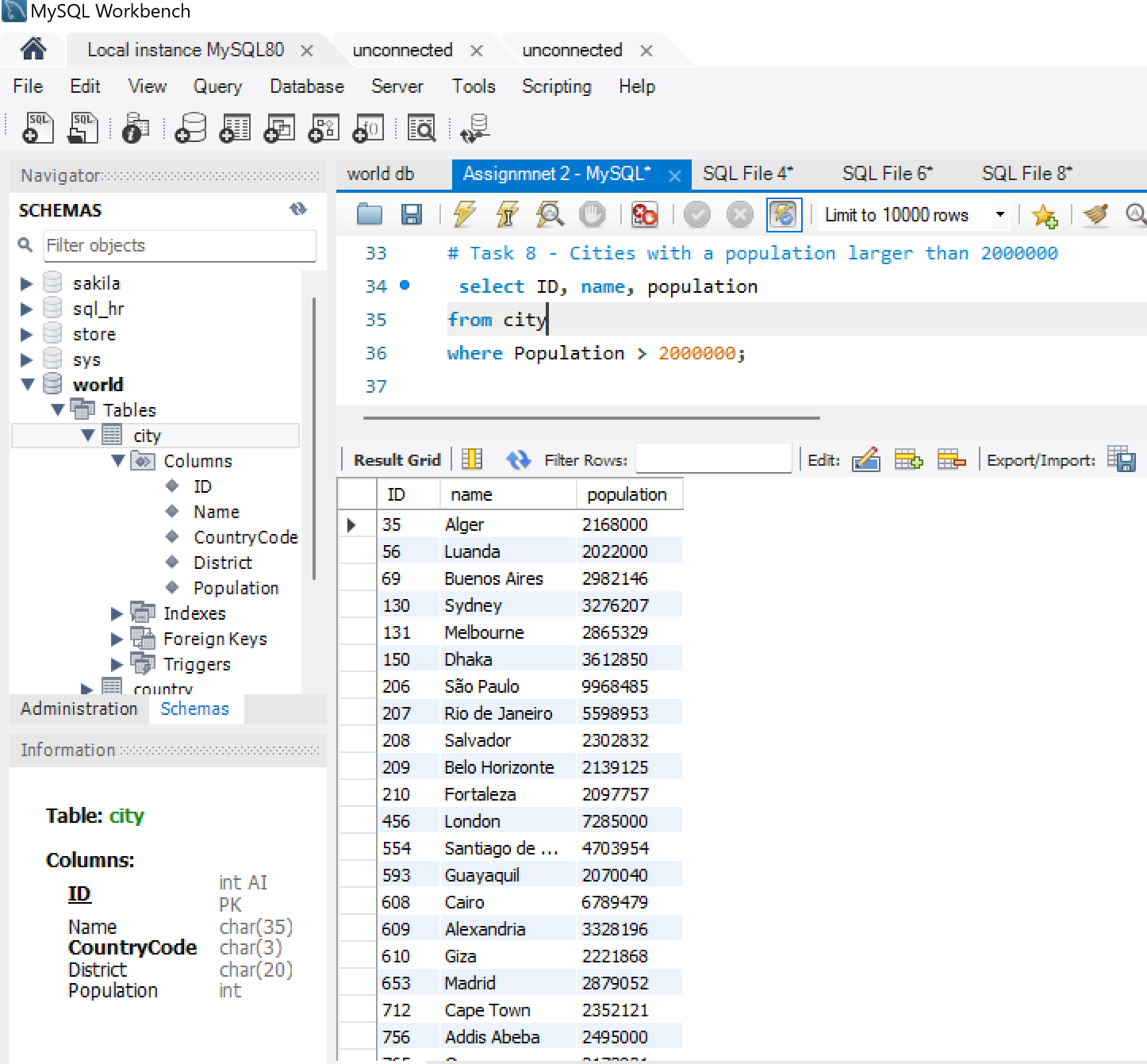
## Task 6 – 25 countries starting with the letter (F)



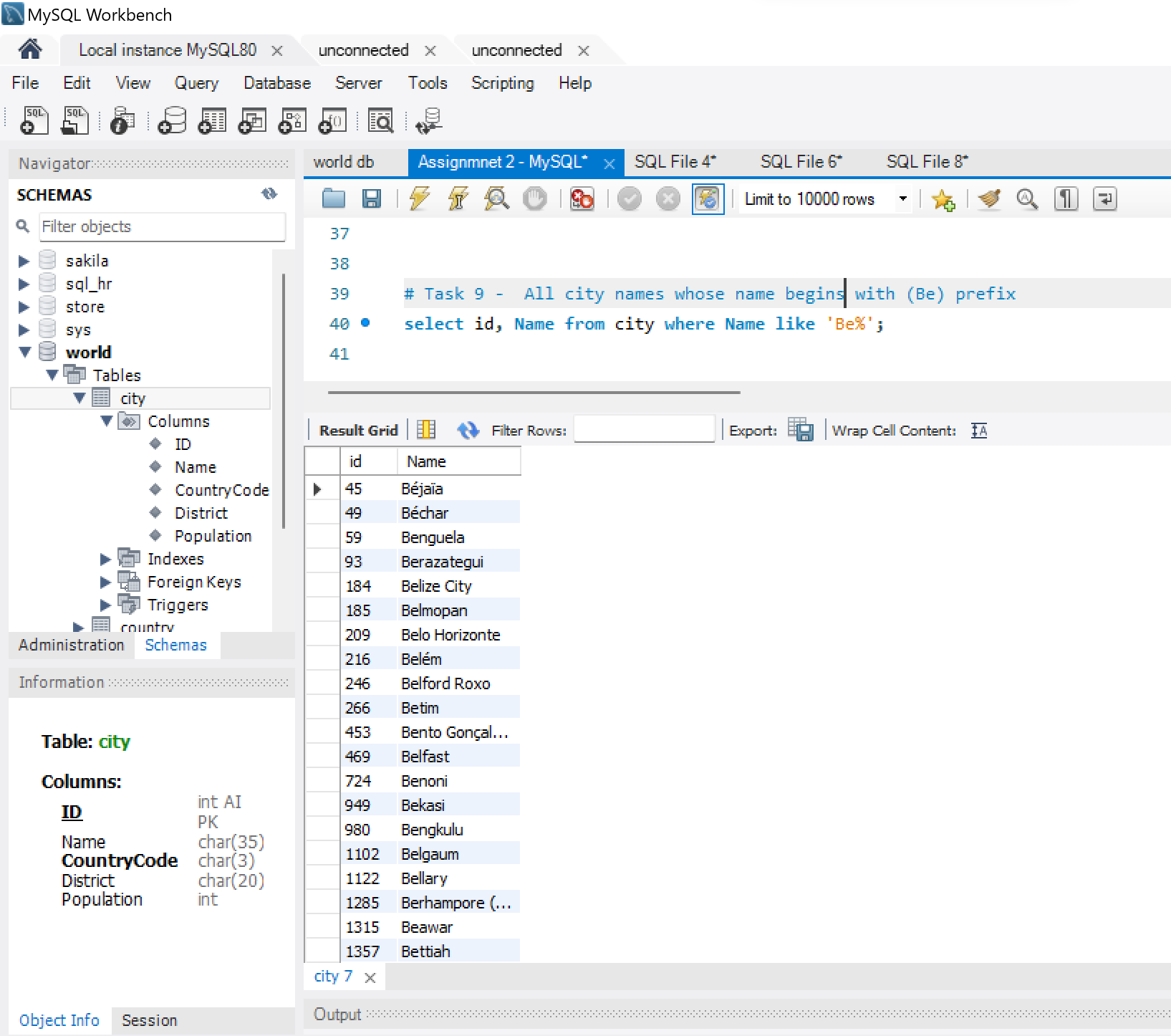
## Task 7 – First 10 rows in the city table



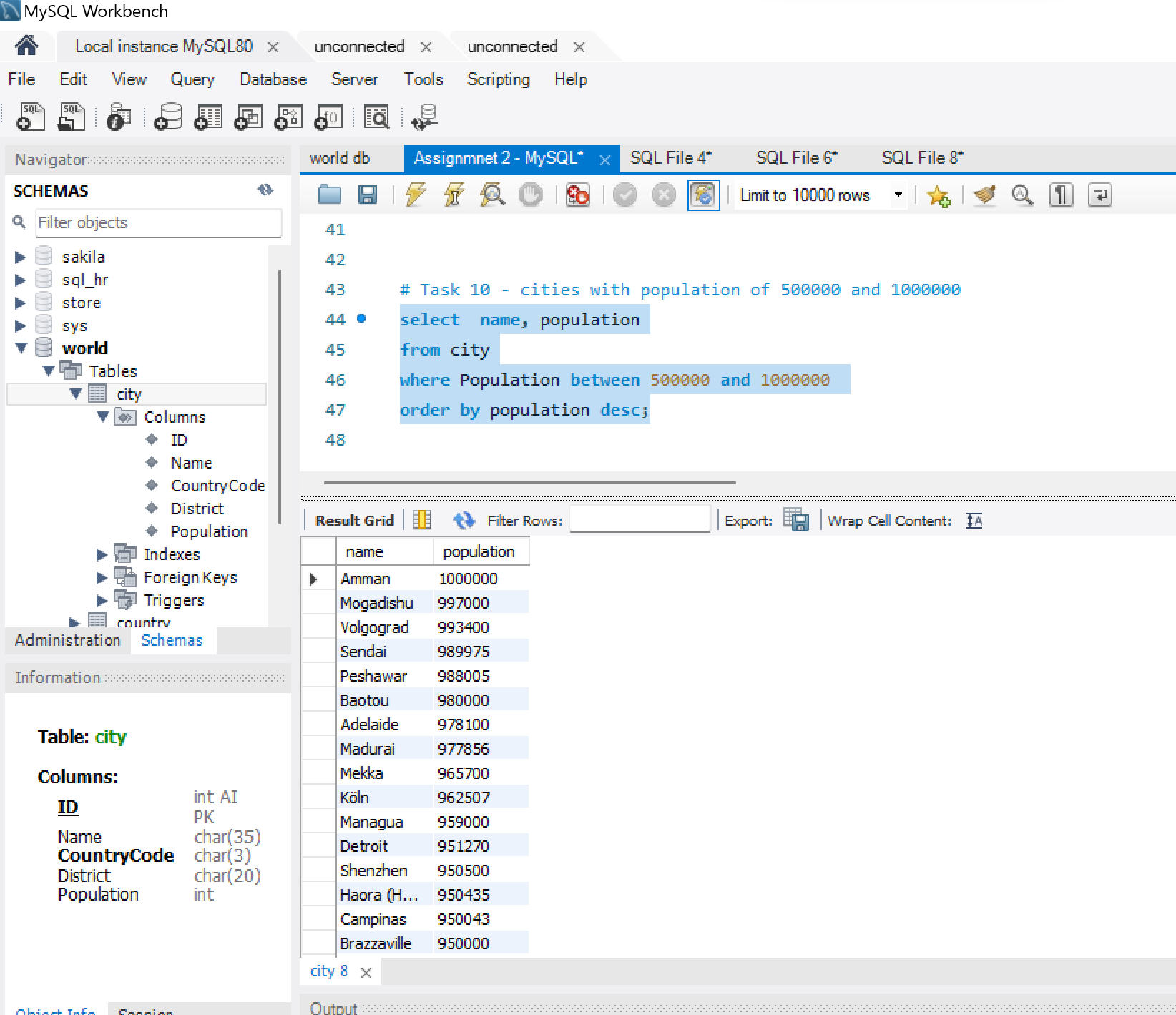
## Task8 – Cities with population of more than 200000



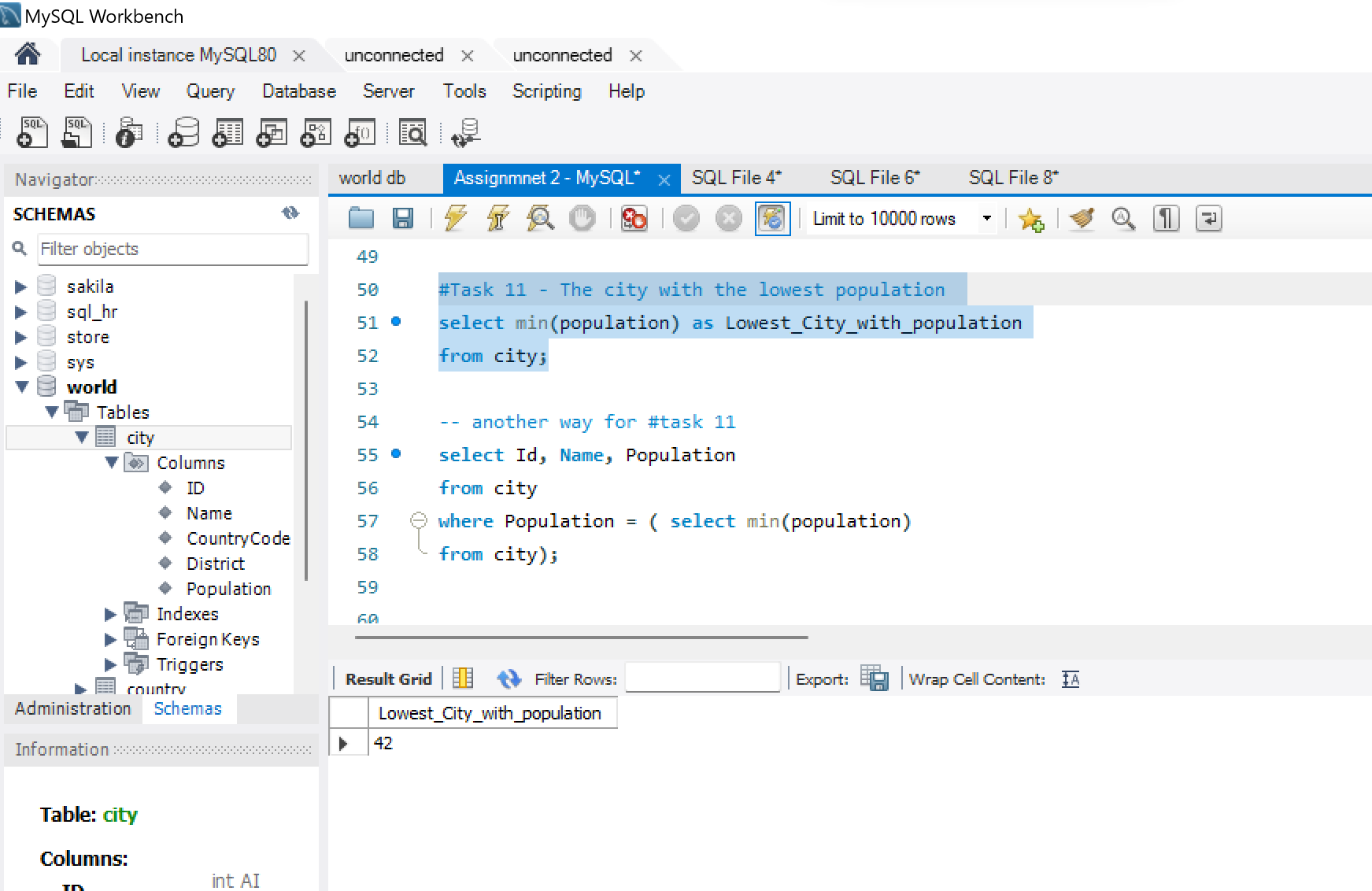
## Task 9 – All city names with the prefix of (Be)



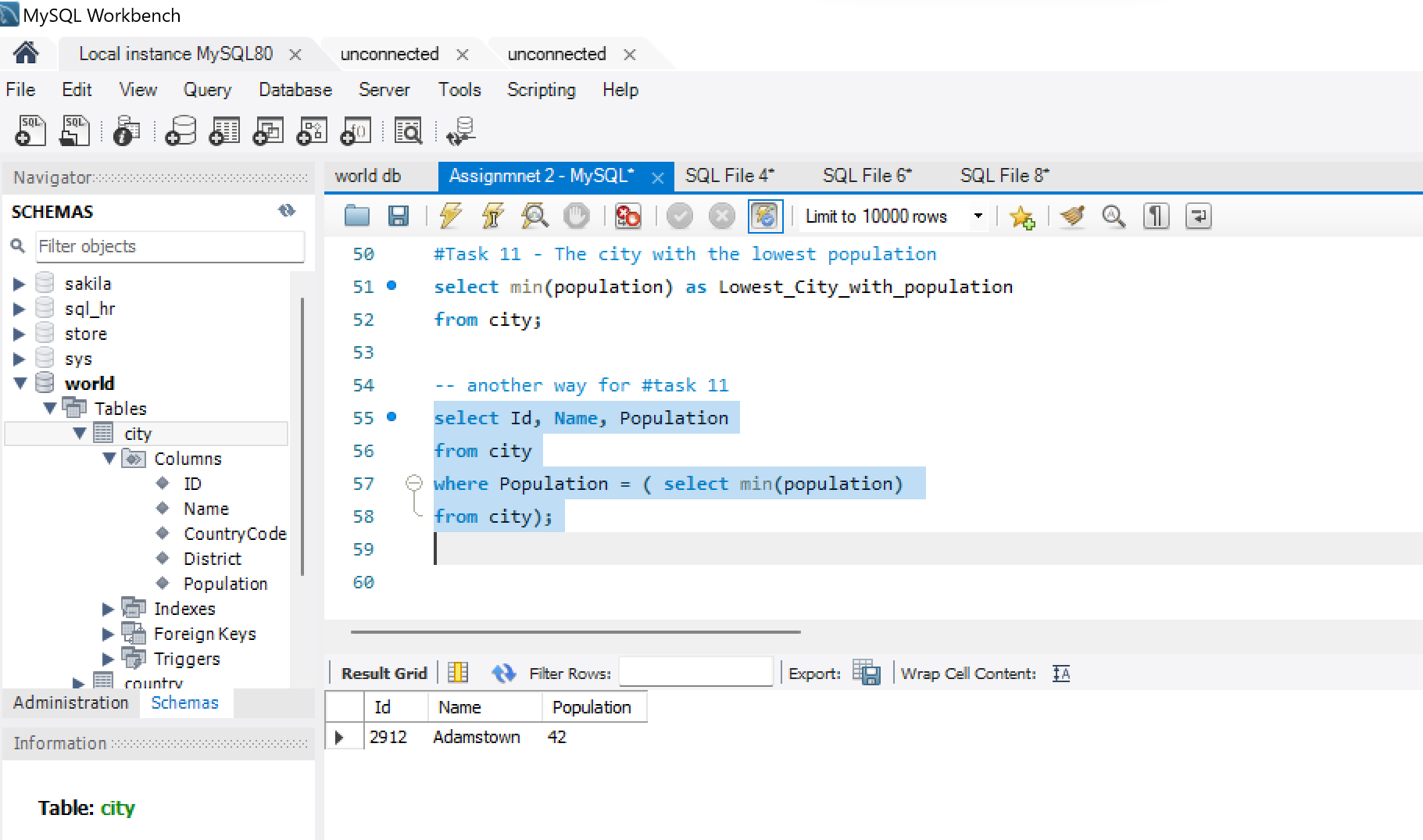
## Task 10 – Cities with population of between 50000 and 100000



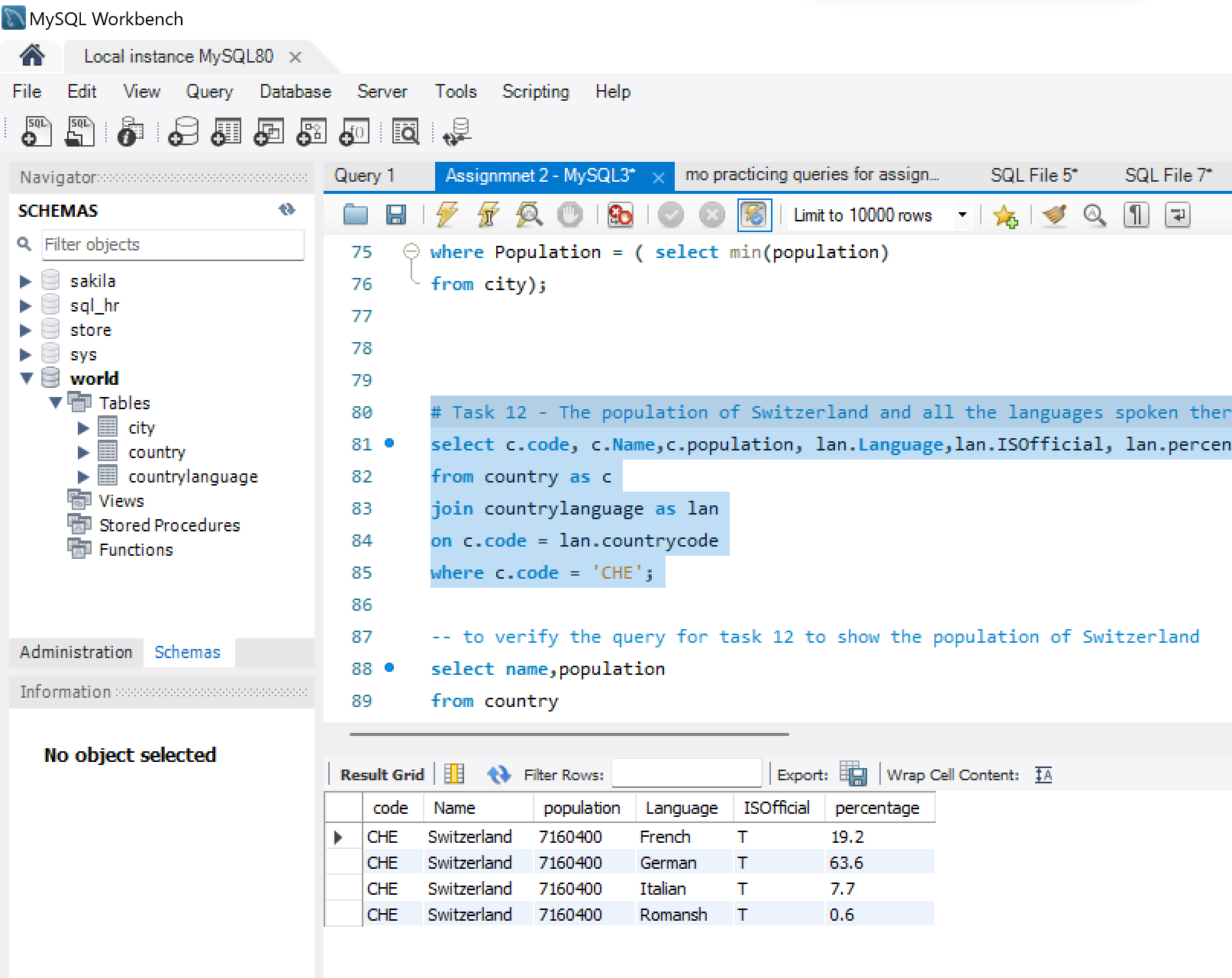
## Task 11 – The city with the lowest population



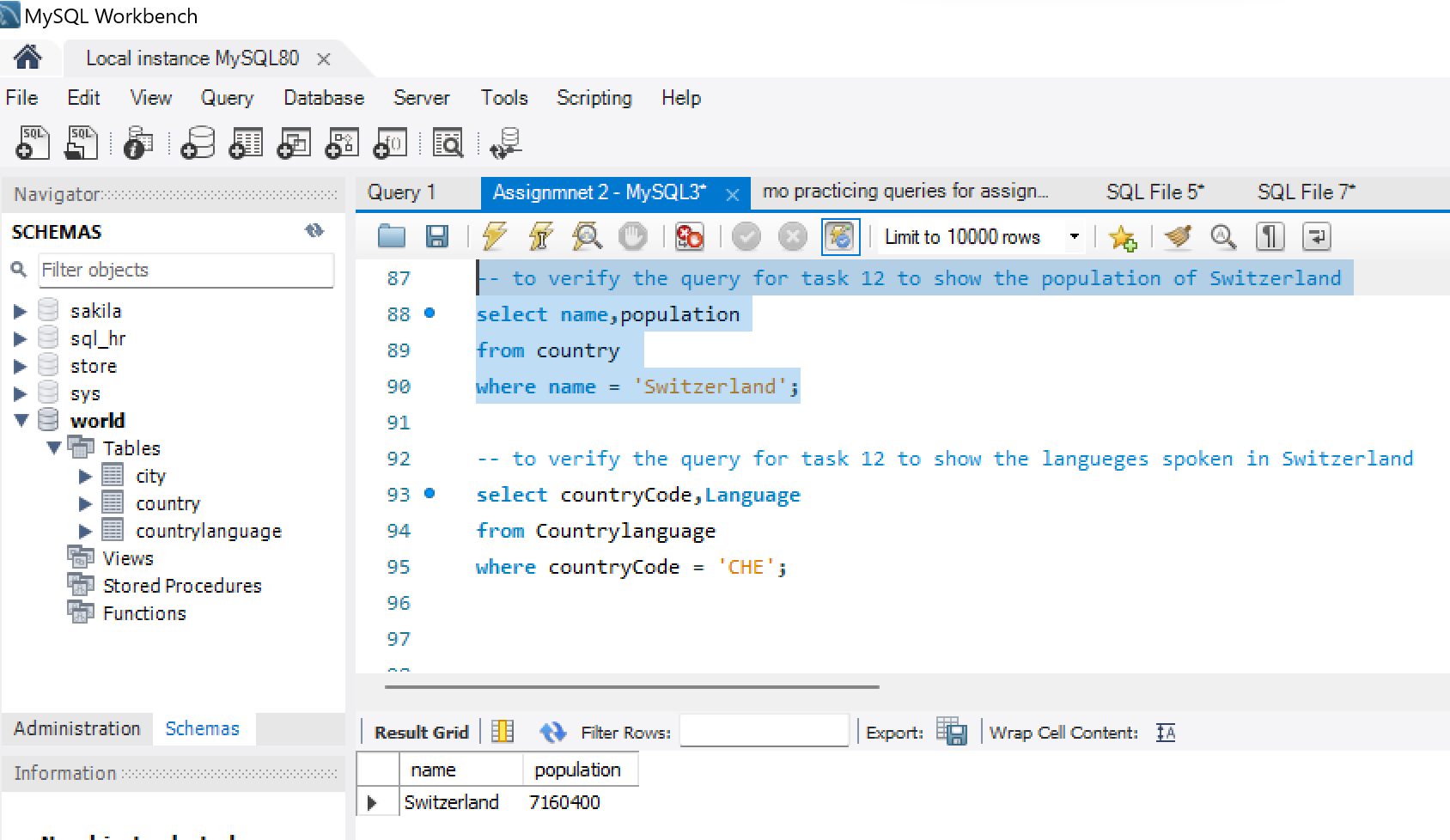
**Below is another way for creating a query for task 11**



## Task12 – Switzerland’s population and language spoken



**To verify the query for task 12 as shown below the population for Switzerland**



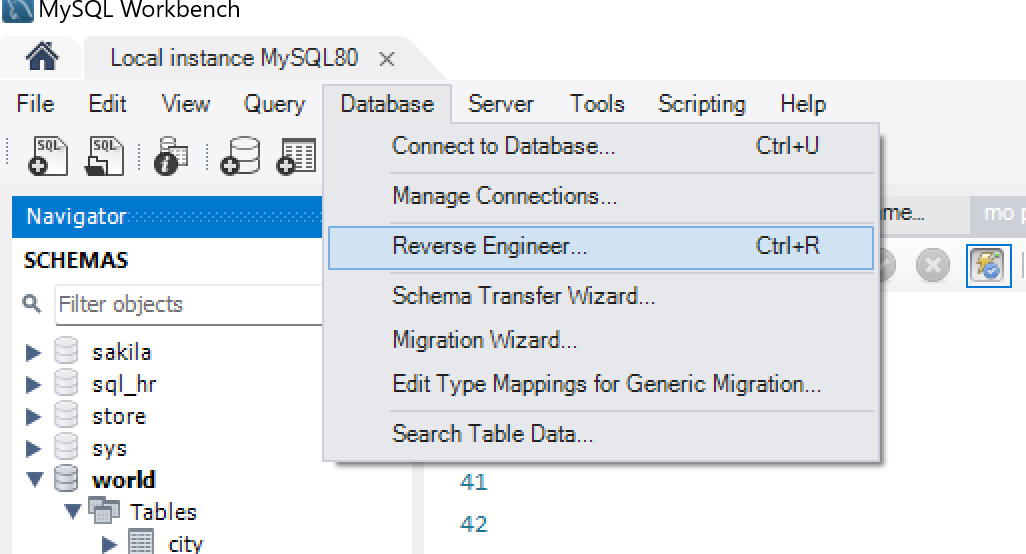
**To verify the query for task 12 as shown below all the languages spoken in Switzerland**

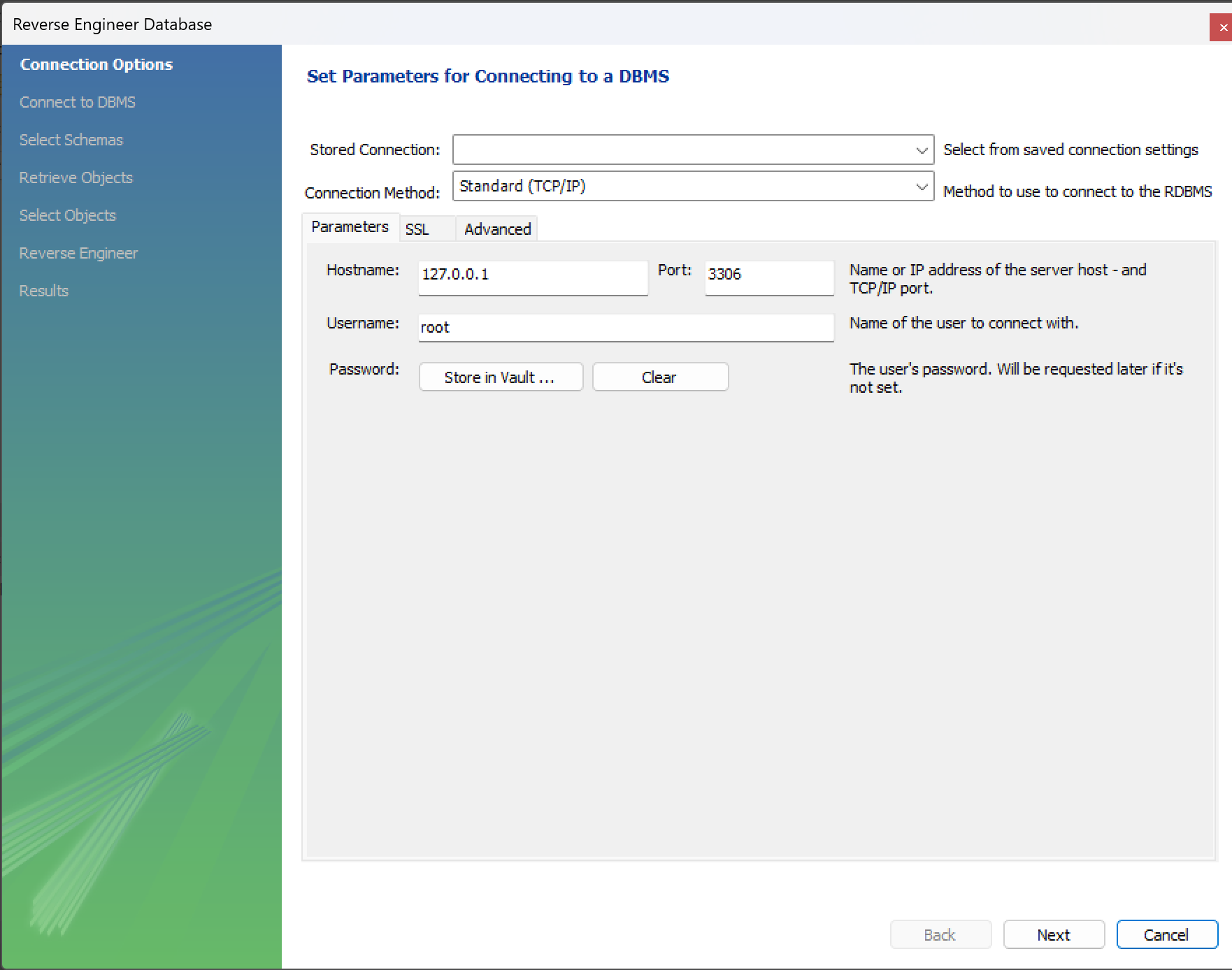


# **EER Diagram**

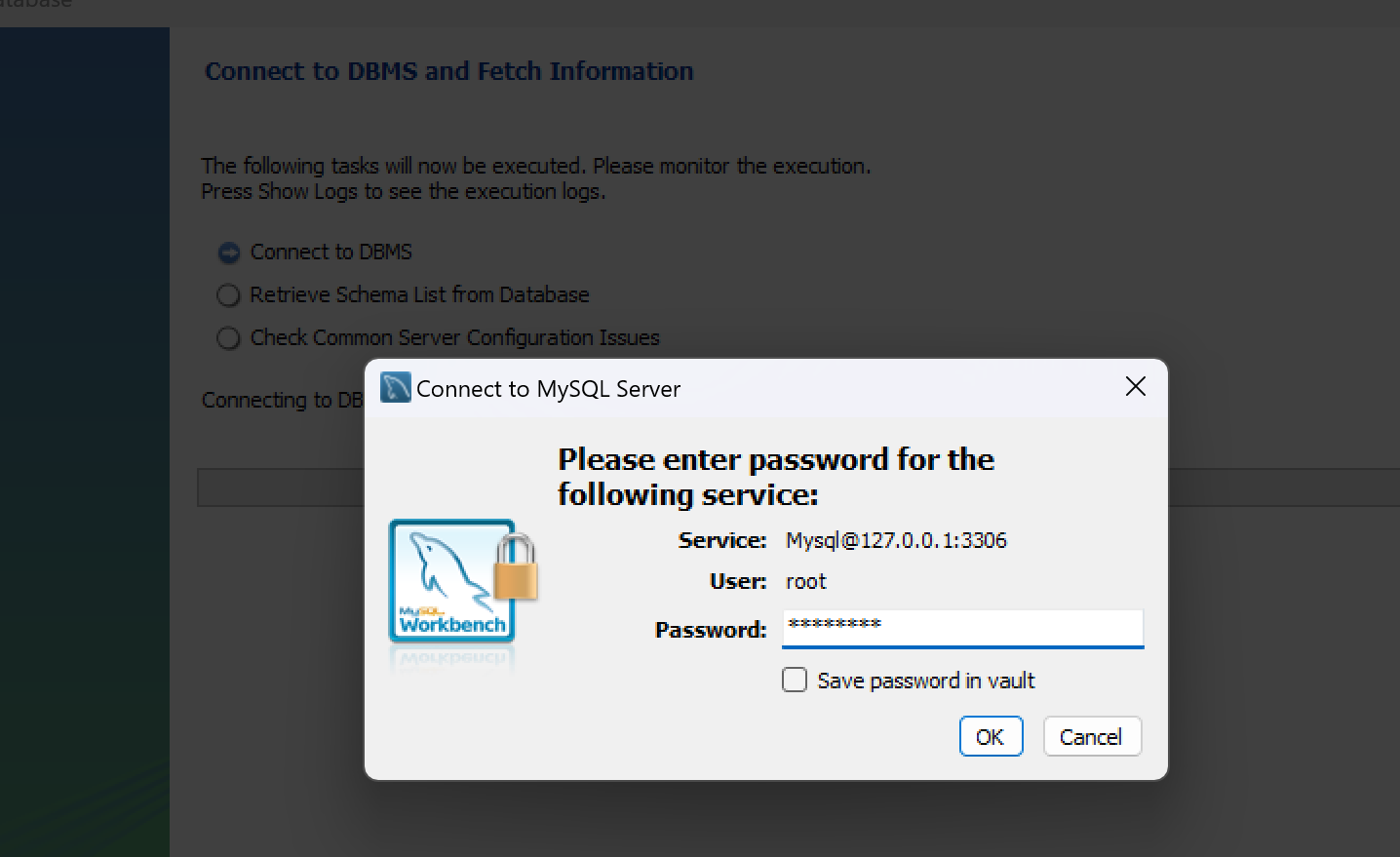
## Task 13 – Creating an EER Diagram

**As shown below I have created the EER diagram**

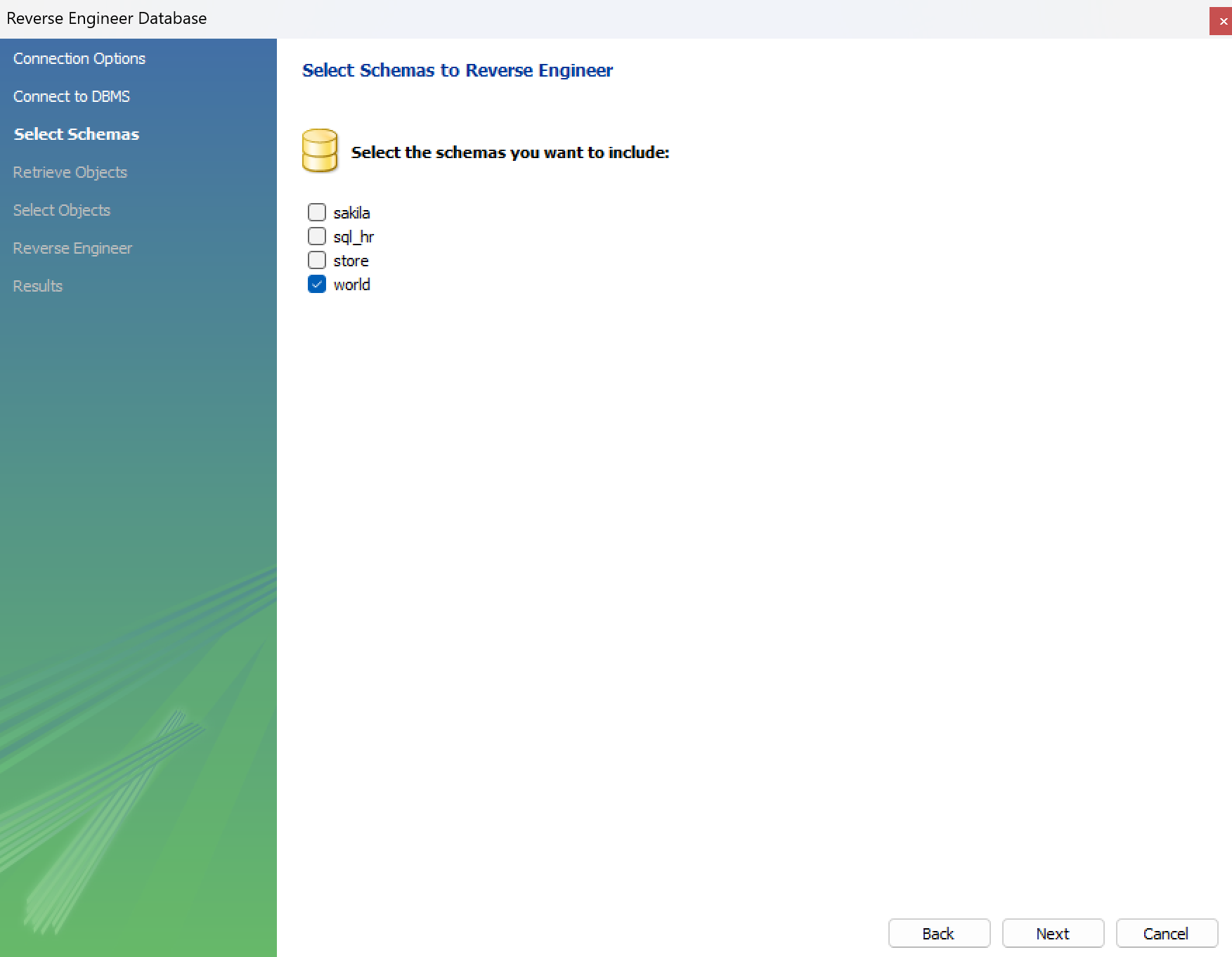


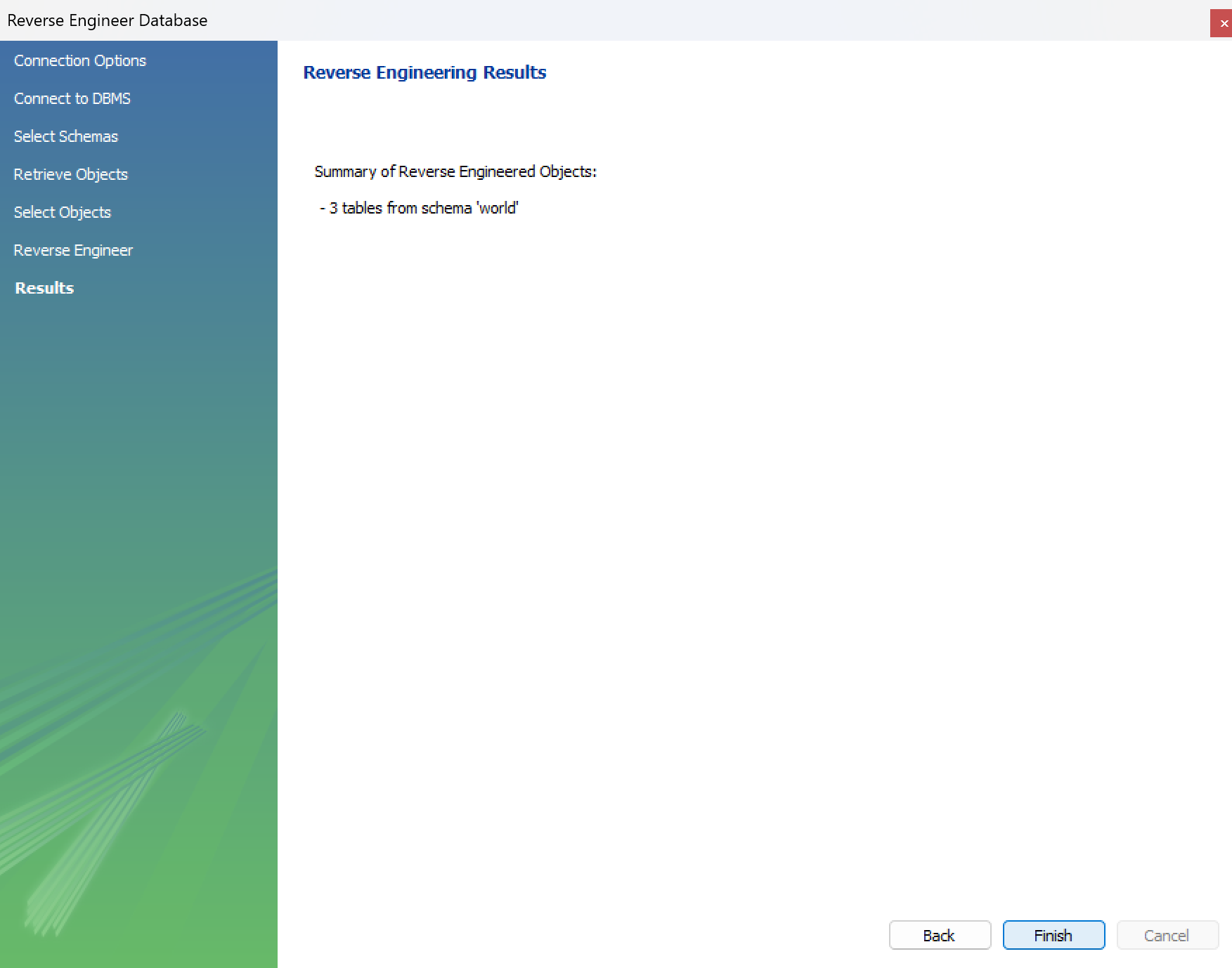


**This screenshot below shows that I have entered my password to connect to the mysql server**



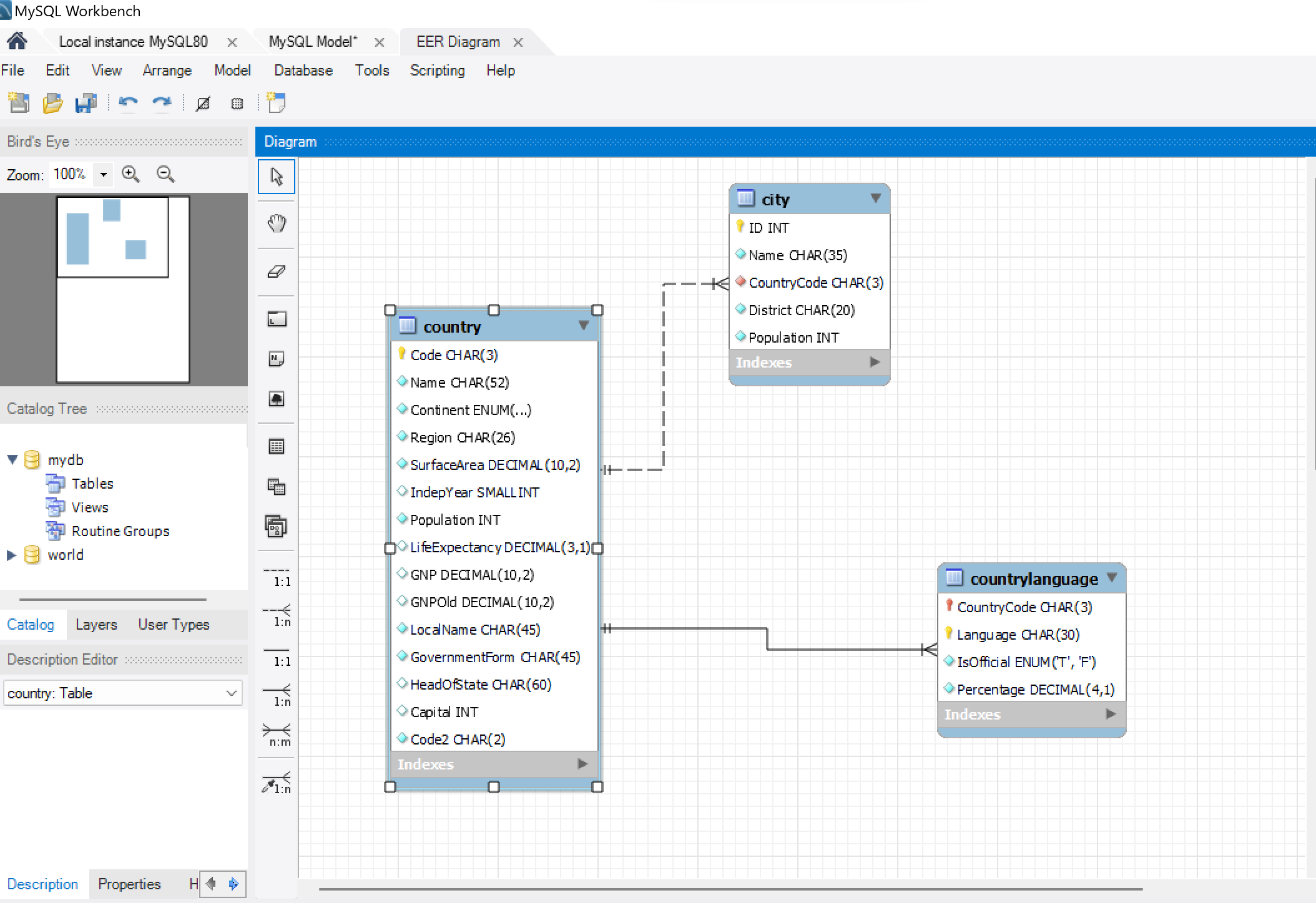
**As shown below I selected the world database to create the EER diagram**





## Task 14 – Primary Keys and Foreign Keys

**As shown below the EER has been created and it has three tables the country table , city table and the Countrylanguage table.**



**The primary key** in the country table is **code.**

**The primary key** in the city table is **ID** and it has a **foreign key** which is the **CountryCode.**

**The primary key** in the Countrylanguage table consists of two primary keys which form a **composite key** and these are **Countrycode** and **language**

# **Reflection**

I found the assignment very helpful for practicing what I have learned in MySQL. I have acquired new skills and enhanced my existing ones in MySQL.

I encountered some difficulties with task 5 because I had to figure out how to use the "MAX" keyword to retrieve the country with the highest life expectancy. After revisiting the SQL session's notes and recordings, I managed to obtain the desired result for the query.

I also faced challenges with task 12. While I successfully displayed all the languages spoken in Switzerland, I noticed that the population was repeated for each row. I believe there might be a more efficient way to display the population without it being repeated for each language spoken in Switzerland.

Overall, this assignment provided me with the opportunity to test my knowledge and practice further. In conclusion, I recognize the need to practice more with MySQL to become more confident in using it comfortably. Additionally, I aim to learn more advanced SQL statements to elevate my understanding of MySQL to the next level.