SQL Assignment Part 2

using the 'World database' script

Tamima Tarofdar

Data Technician Skills Bootcamp - Just IT (April 2024)

Task 1

- - - - X

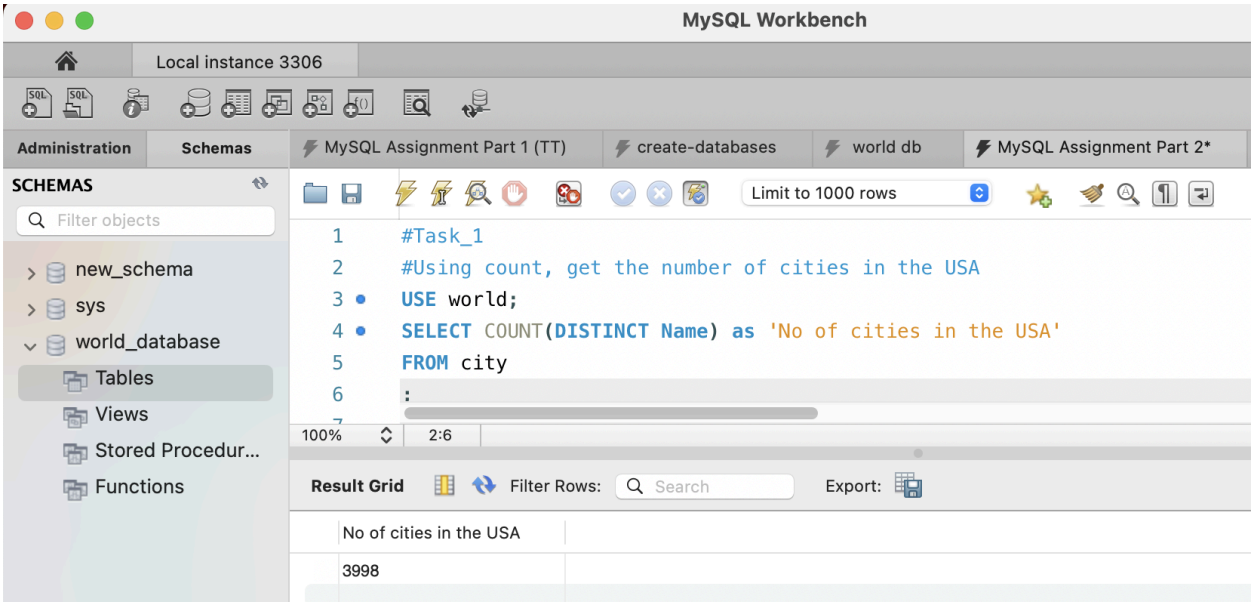
Using count, get the number of cities in the USA

Answer: 3998

Within this code, the ‘USE world’ line specifies that I would like to use the ‘world’ database. This means that all queries from now on will be executed within this database.

The ‘SELECT’ statement is used to retrieve data from the database and the ‘COUNT(DISTINCT Name)’ function counts the number of unique values in the 'Name' column of the 'city' table.

The DISTINCT keyword ensures that each city name is counted only once and ‘as 'No of cities in the USA’’ operation is renaming the result of the count function to 'No of cities in the USA' so the viewer can read and understand the results better.



Task 2

- - - - X

Find out what the population and average life expectancy for people in Argentina (ARG) is.

Answers:

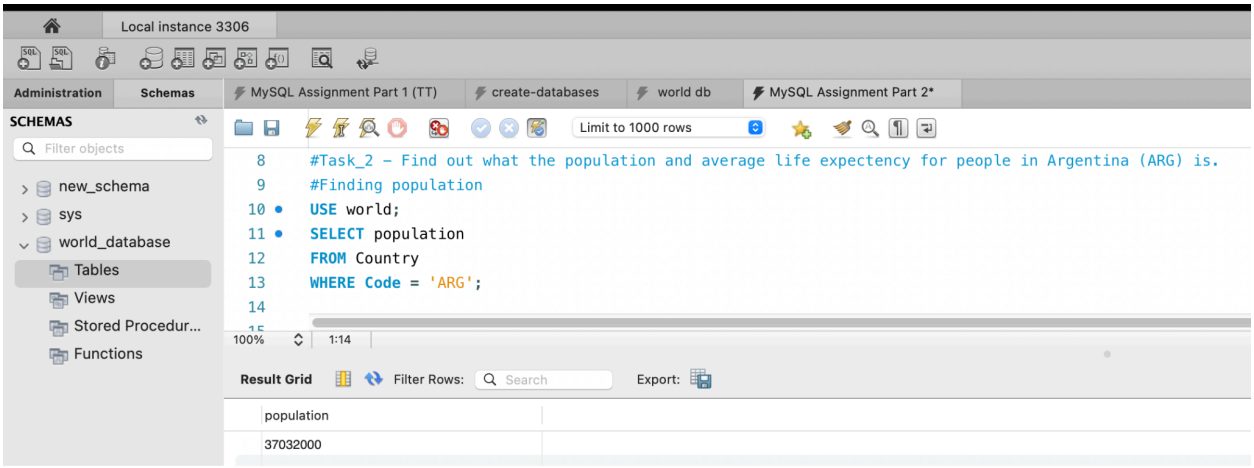
Population = 37032000

Life Expectancy = 75.1

Part 1 - Population

To tackle this task I decided to split it into two parts. In part 1 I used the 'SELECT' statement to select the 'population' column and the 'FROM' statement to specify that this data should be retrieved from the 'Country' table.

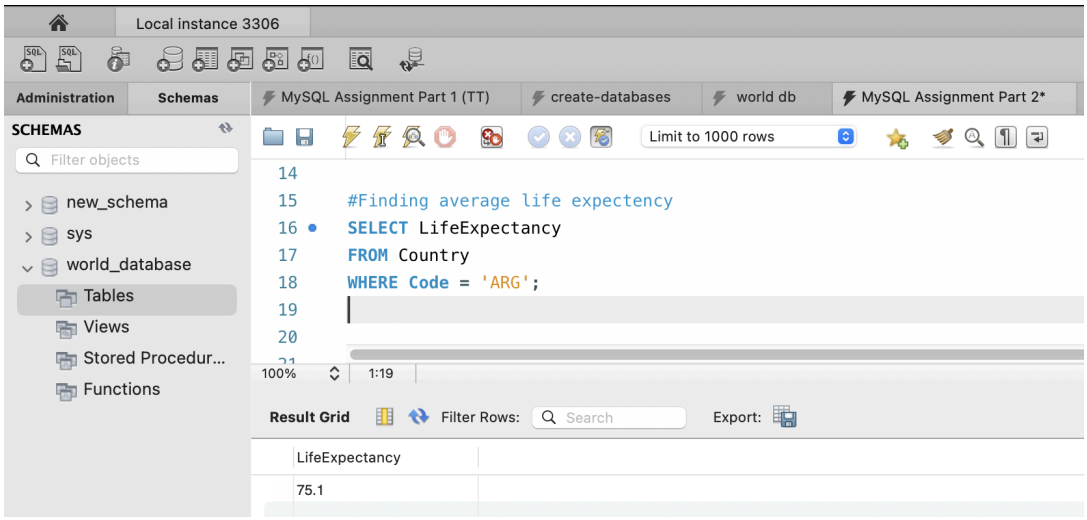
I then used the 'WHERE Code = 'ARG'' function to apply a condition so the results are filtered to only include rows where the 'Code' is equal to ARG, which is the country code for Argentina in this case.



Part 2 - Life Expectancy

In part 2 I used a similar code to the previous one but instead used the ‘SELECT’ statement to select the ‘LifeExpectancy’ column.

I then used the same code from the previous part where the ‘FROM’ statement specifies that this data should be retrieved from the ‘Country’ table and the ‘WHERE Code = 'ARG’’ to view the results specific to Argentina.



Task 3

- - - - X

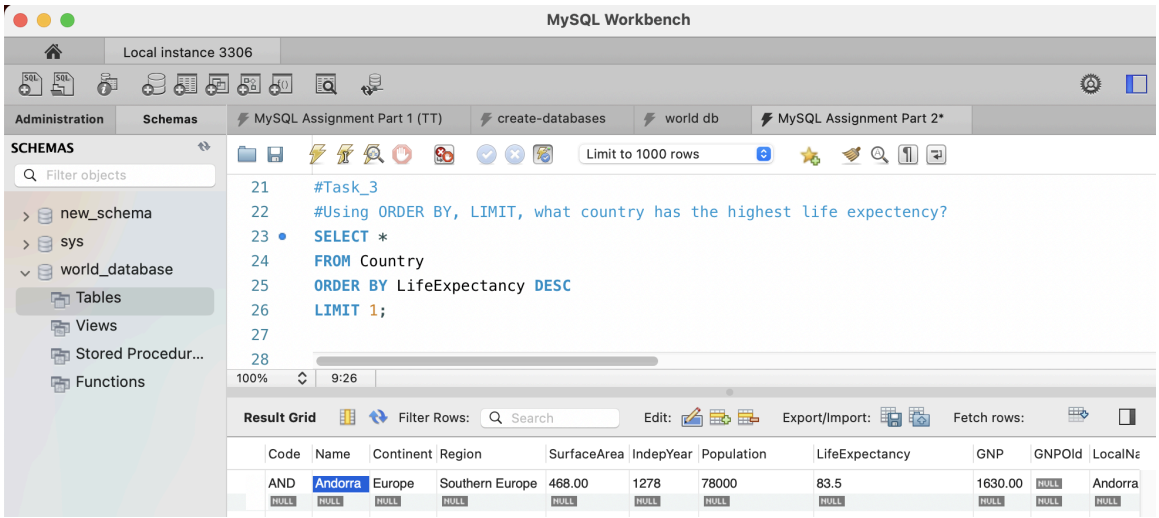
Using ORDER BY, LIMIT, what country has the highest life expectancy?

Answer: Andorra

For this task, I used the asterisk (*) symbol with the 'SELECT' statement to select all the columns from the 'Country' table.

I used the 'ORDER BY' statement to organise results based on the 'LifeExpectancy' column. The 'DESC' statement was added for the column to be sorted in descending order so the countries with the highest life expectancy will be listed first in the results.

I then used the 'LIMIT 1' function for the query to return 1 row only and since the table result is now organised by life expectancy descending, this meant that only the row with the country with the highest life expectancy will be shown.



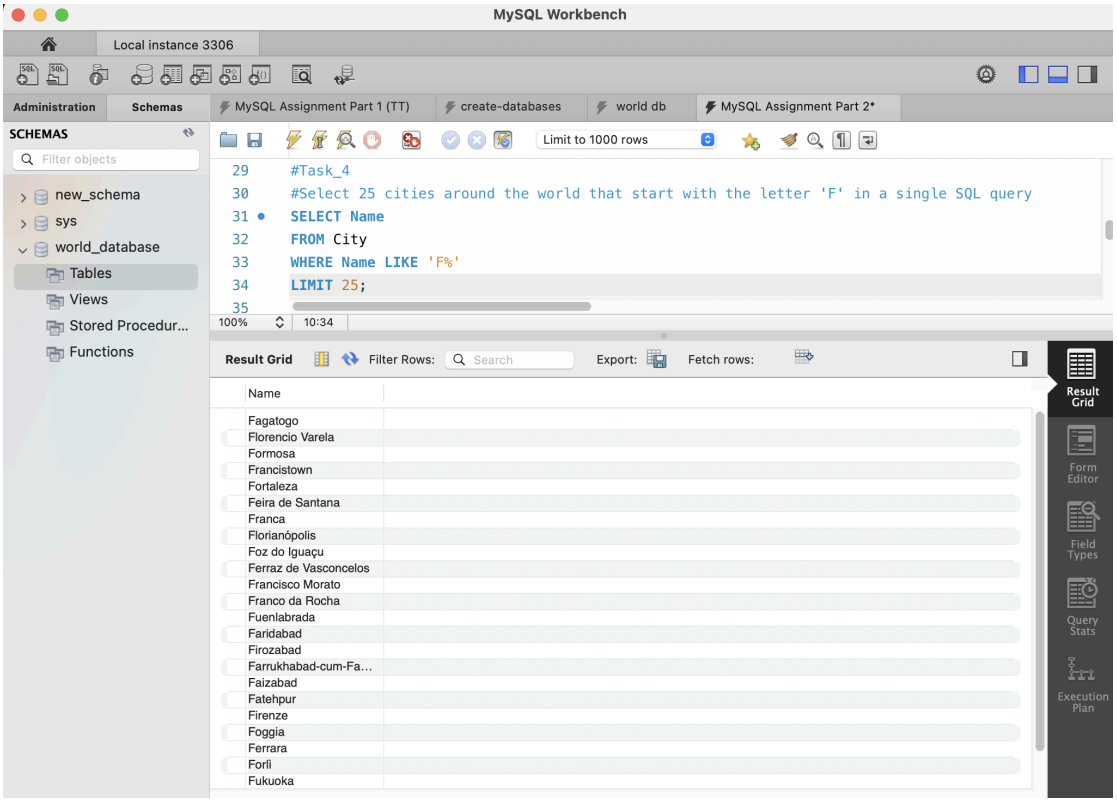
Task 4

- - - - X

Select 25 cities around the world that start with the letter 'F' in a single SQL query.

For Task 4 I have used the ‘SELECT’ statement to select the ‘Name’ column and the ‘FROM’ statement to specify this is to be retrieved from the ‘City’ table.

I used the ‘WHERE’ statement to apply a condition using the ‘LIKE 'F%’ function to instruct the query to return only cities which begin with the letter 'F', followed by zero or more characters. The ‘LIMIT 25’ line is then used to limit the number of rows returned by the query to 25.



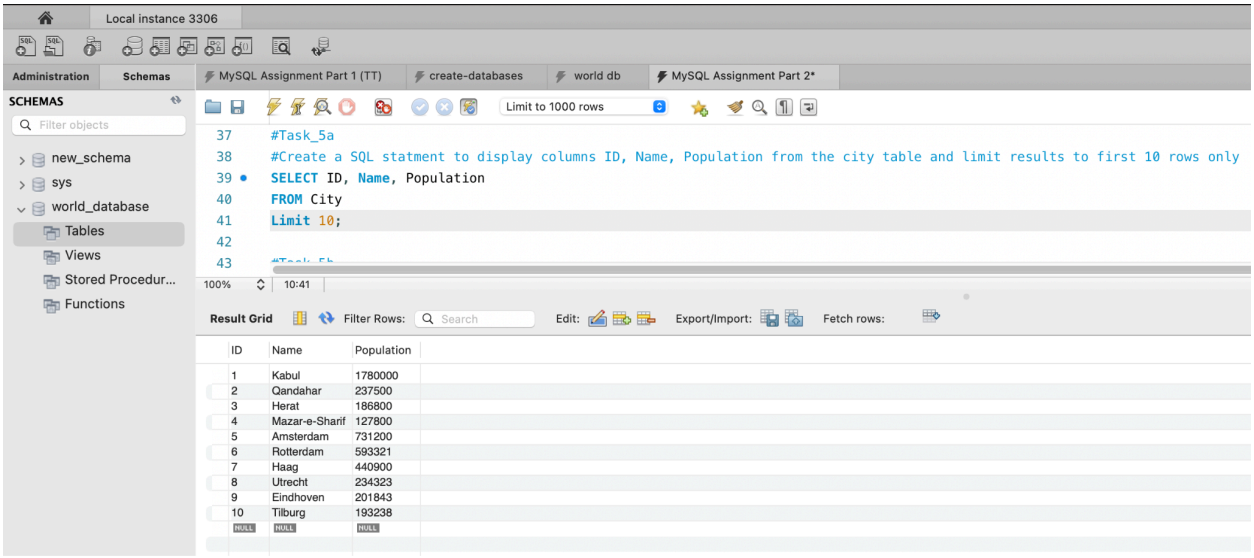
Task 5

- - - - X

Part a)

Create an SQL statement to display columns Id, Name, Population from the city table and limit results to first 10 rows only.

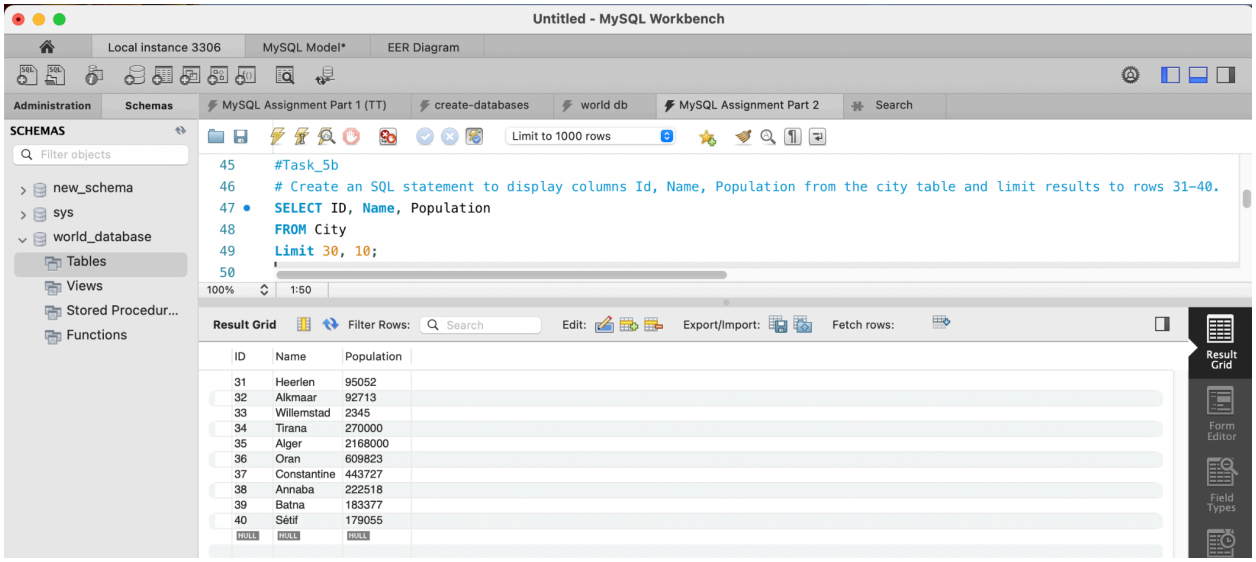
For part a) I used the 'SELECT' statement to select the 'ID', 'Name', and 'Population' columns and the 'FROM' statement to specify this is to be retrieved from the 'City' table. I then used the 'LIMIT 10' statement so that only the first 10 rows are shown in the results.



Part b)

Create an SQL statement to display columns Id, Name, Population from the city table and limit results to rows 31-40.

For part b) I used the same code in part a) but changed the 'LIMIT 10' line to 'LIMIT 30, 10'. As rows in SQL are numbered from zero I had to consider that the 31st row has an offset of 30. Therefore 'LIMIT 30,10' specifies that the results should start from the 31st row (offset 30) and return 10 rows.



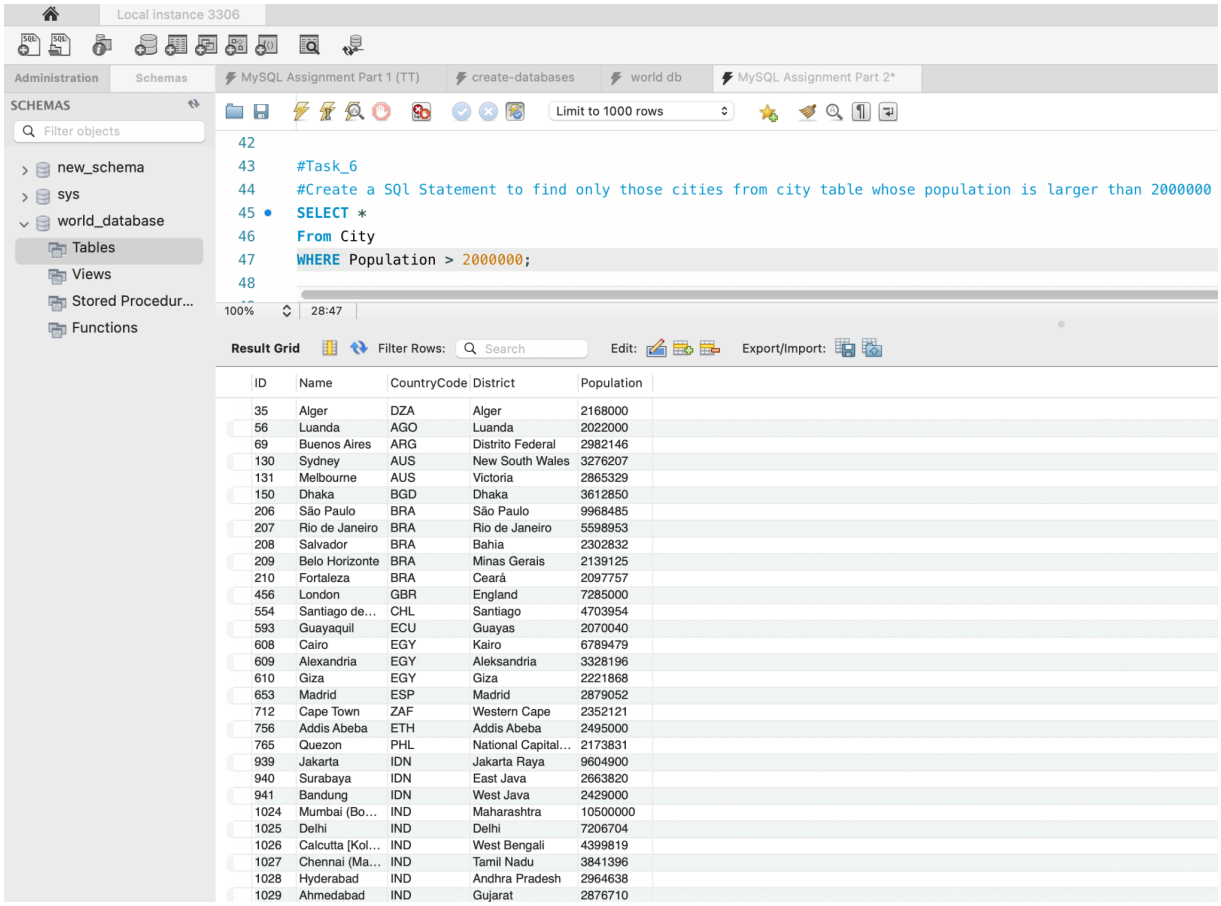
Task 6

- - - - X

Create an SQL statement to find only those cities from city table whose population is larger than 2000000.

For this task, I used the asterisk (*) symbol with the 'SELECT' statement to select all the columns from the 'City' table.

I then used the 'WHERE Population > 2000000' clause to apply a condition so the results are filtered to only include rows where cities with populations exceeding 2,000,000 are shown in the results.



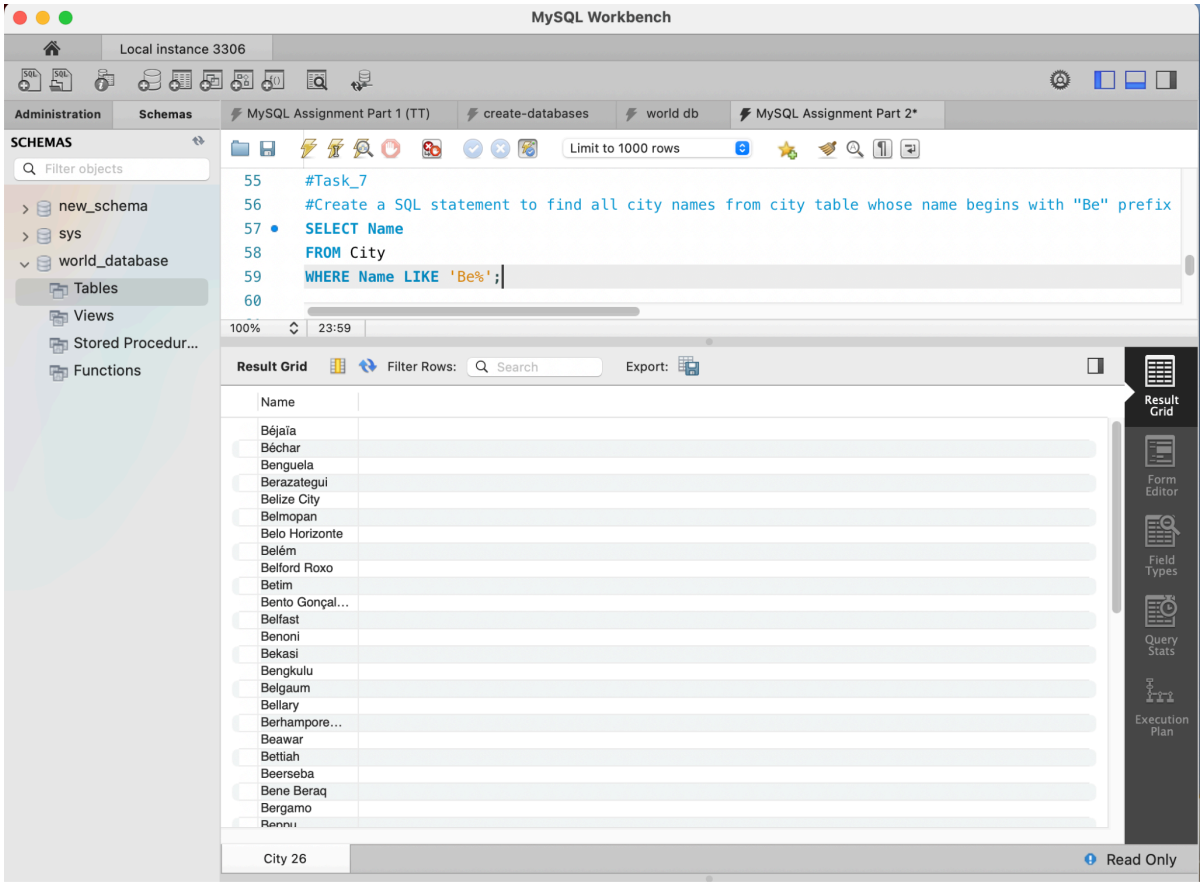
Task 7

- - - - X

Create an SQL statement to find all city names from city table whose name begins with Be prefix.

For Task 7 I have used the ‘SELECT’ statement to select the ‘Name’ column and the ‘FROM’ statement to specify this is to be retrieved from the ‘City’ table.

I used the ‘WHERE’ statement to apply a condition using the ‘LIKE 'Be%’’ function to instruct the query to return only cities whose names start with ‘Be’.



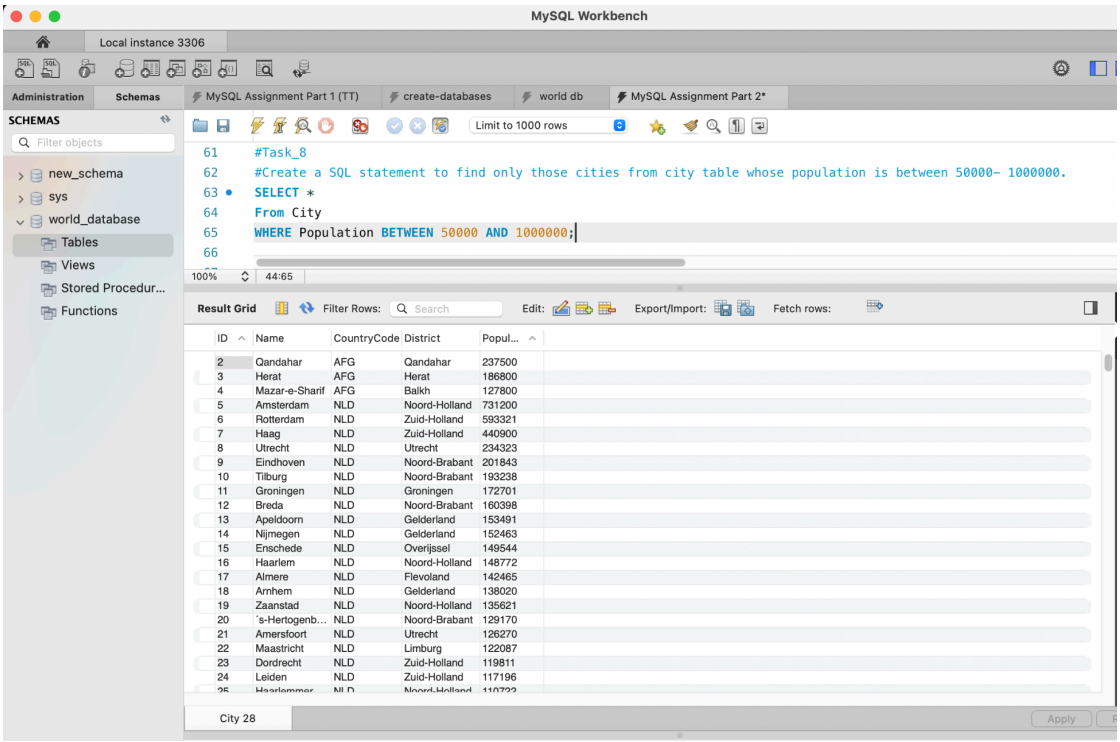
Task 8

- - - - X

Create an SQL statement to find only those cities from city table whose population is between 500000-1000000.

For this task, I used the asterisk (*) symbol with the 'SELECT' statement to select all the columns from the 'City' table.

I then used the 'WHERE Population BETWEEN 50000 AND 1000000' clause to apply a condition so the results are filtered to only include cities where the data in the population column is inclusive of and falls between 50,000 and 1,000,000.



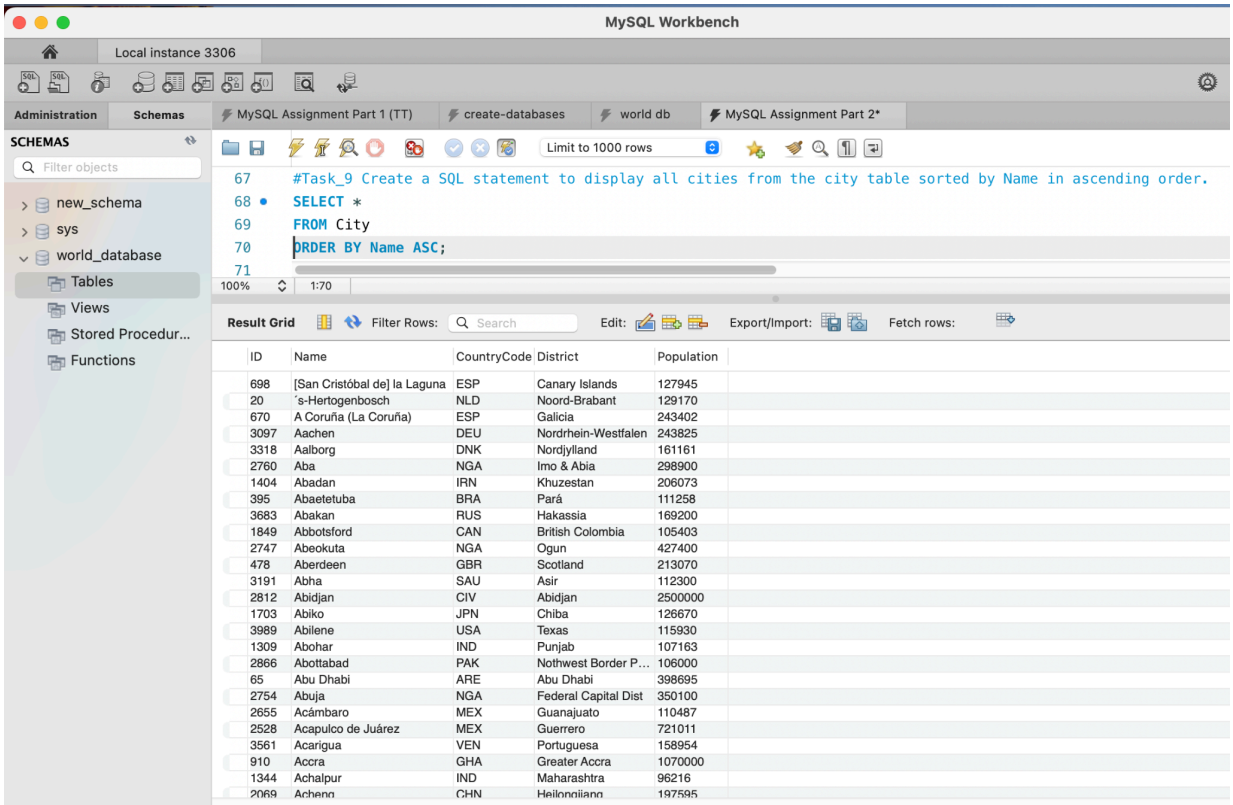
Task 9

- - - - X

Create an SQL statement to display all cities from the city table sorted by Name in ascending order.

For this task, I used the asterisk (*) symbol with the 'SELECT' statement to select all the columns from the 'City' table.

I then used the 'ORDER BY' statement to organise results based on the 'Name' column. The 'ASC' statement enables the column to be sorted in ascending order so information about all cities in the 'City' table is arranged alphabetically by the city name when the query is run.



Task 10

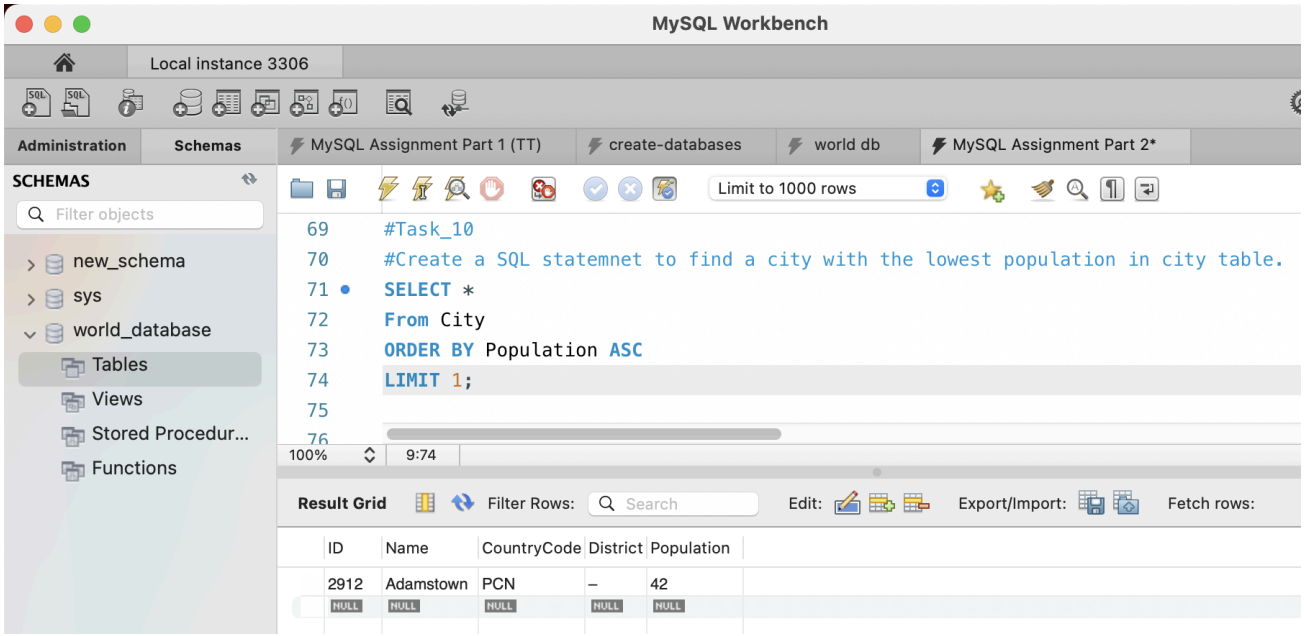
- - - - X

Create an SQL statement to find a city with the lowest population in the city table.

For task 10, I used the asterisk (*) symbol with the 'SELECT' statement to select all the columns from the 'City' table.

I used the 'ORDER BY' statement to organise results based on the 'Population' column. The 'ASC' statement enables the column to be sorted in ascending order so cities with the lowest population will be listed first in the results.

I then used the 'LIMIT 1' function for the query to select only the first row, which now represents the city with the lowest population.



Task 11

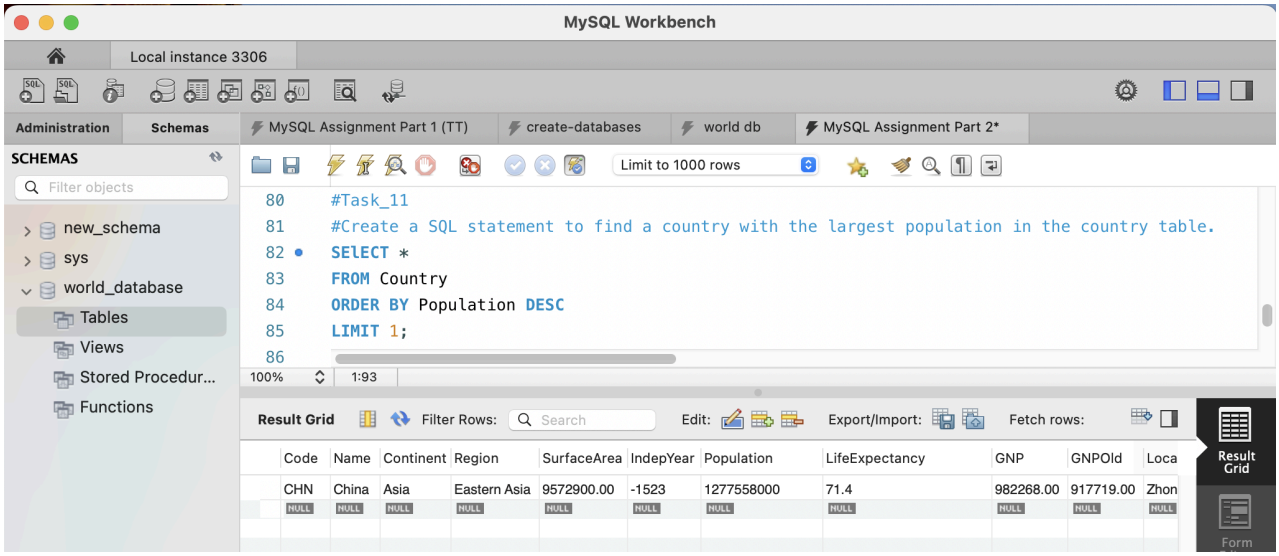
- - - - X

Create an SQL statement to find a country with the largest population in the country table.

For task 11, I used the asterisk (*) symbol with the 'SELECT' statement to select all the columns from the 'Country' table.

I used the 'ORDER BY' statement to organise results based on the 'Population' column. The 'DESC' statement enables the column to be sorted in descending order so the country with the highest population will be listed first in the results.

I then used the 'LIMIT 1' function for the query to show only the first row, which now represents the country with the largest population.



Bonus Tasks

- - - - X

Task 1

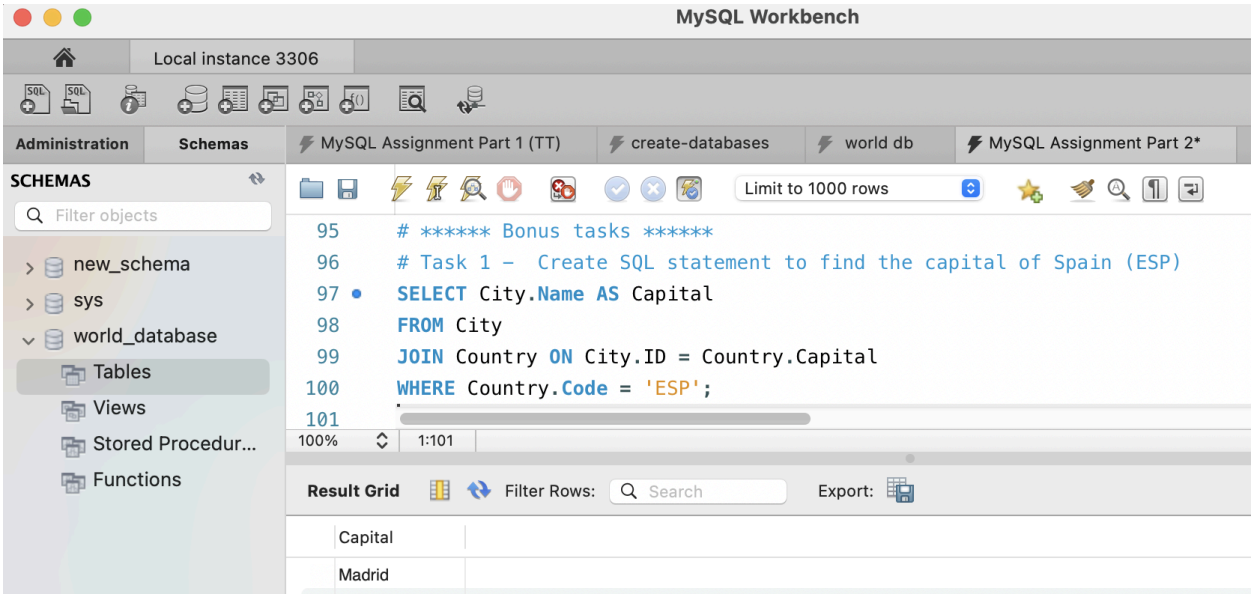
Create SQL statement to find the capital of Spain (ESP)

Answer: Madrid

Within this query, the first line selects the 'Name' column from the 'City' table, and using the AS statement I renamed it "Capital" to improve understanding and readability. This means that the selected column will be displayed as "Capital" when the query is run.

The JOIN statement uses the 'ID' column from the 'City' table and the 'Capital' column from the 'Country' table to join the two tables together. This ensures that only the capital city of each country is retrieved.

The WHERE statement filters the results to only show the country with the country code 'ESP', which represents Spain.



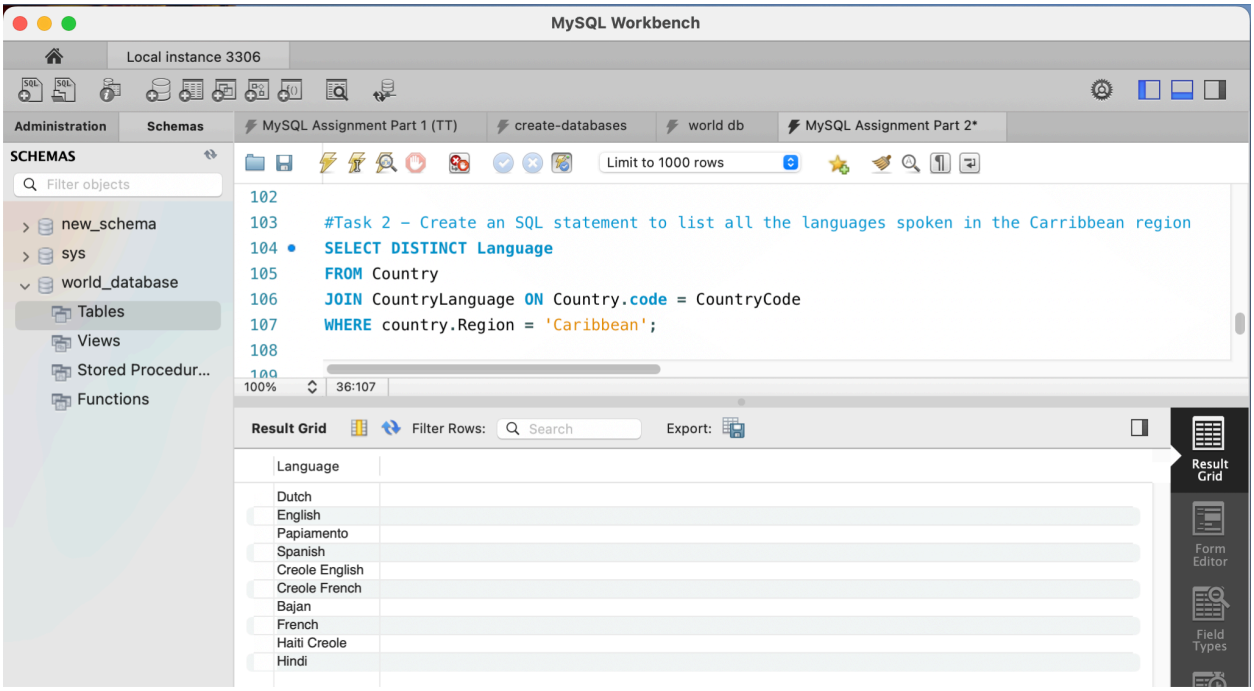
Task 2

Create an SQL statement to list all the languages spoken in the Caribbean region

For this task, the 'Language' column is selected from the 'Country' table. The DISTINCT statement was added to make sure each language is only included once in the result set.

The JOIN statement uses the 'code' column from the 'Country' table and the 'CountryCode' column from the 'CountryLanguage' table to join the two tables together. This ensures that language data is retrieved for each country.

The WHERE statement filters the results to only show countries located in the Caribbean region by applying the condition country.Region = 'Caribbean'.



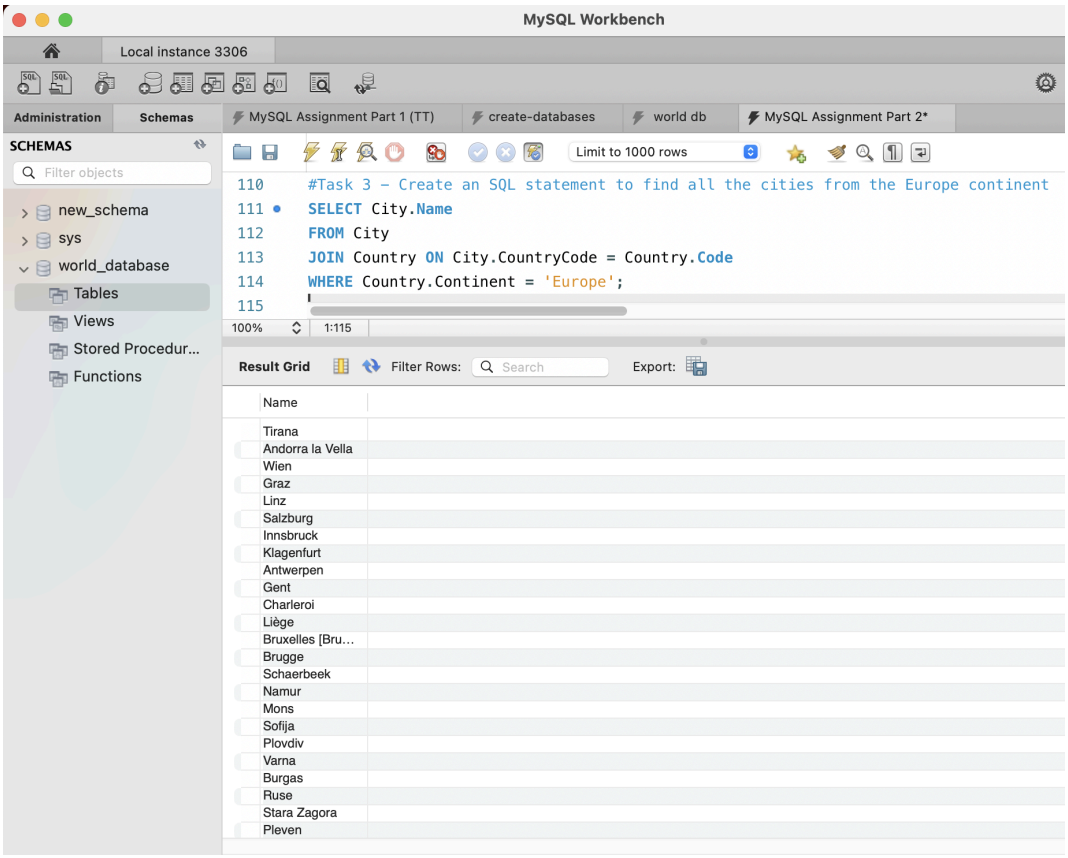
Task 3

Create an SQL statement to find all the cities from the Europe continent

For this bonus task, I have used the 'SELECT' statement to select the 'City.Name' column and the FROM statement to specify this is to be retrieved from the 'City' table.

The JOIN statement uses the 'CountryCode' column from the 'City' table and the 'Code' column from the 'Country' table to join the two tables together. This ensures that city data is retrieved for each country.

The WHERE statement here filters the results to only include countries located in the European continent by specifying the condition Country.Continent = 'Europe'.



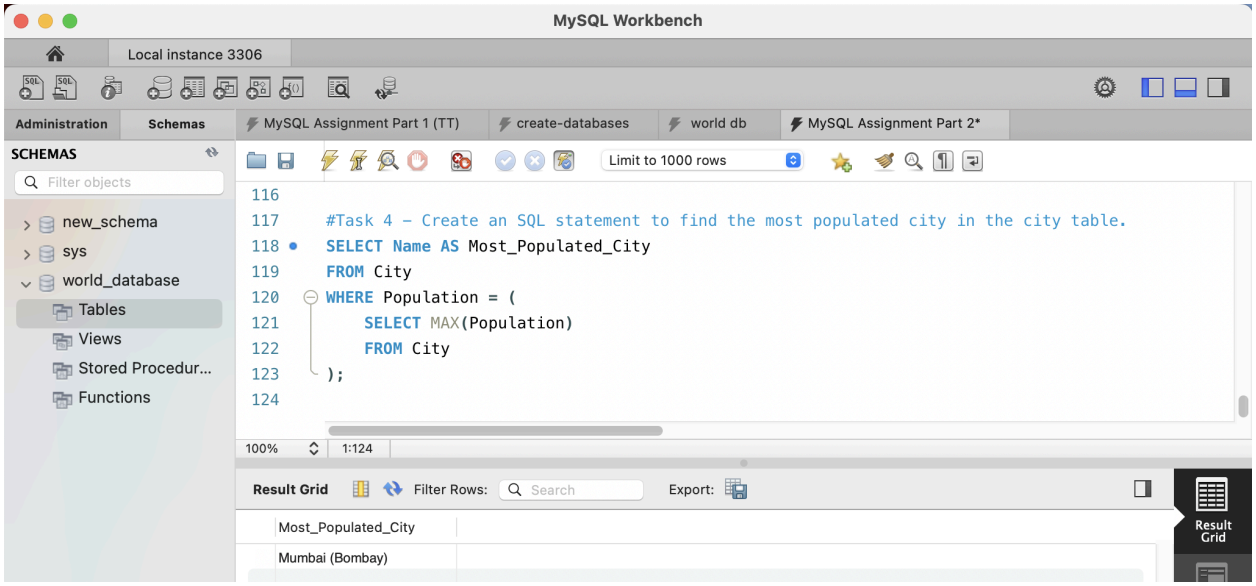
Task 4

Create an SQL statement to find the most populated city in the city table.

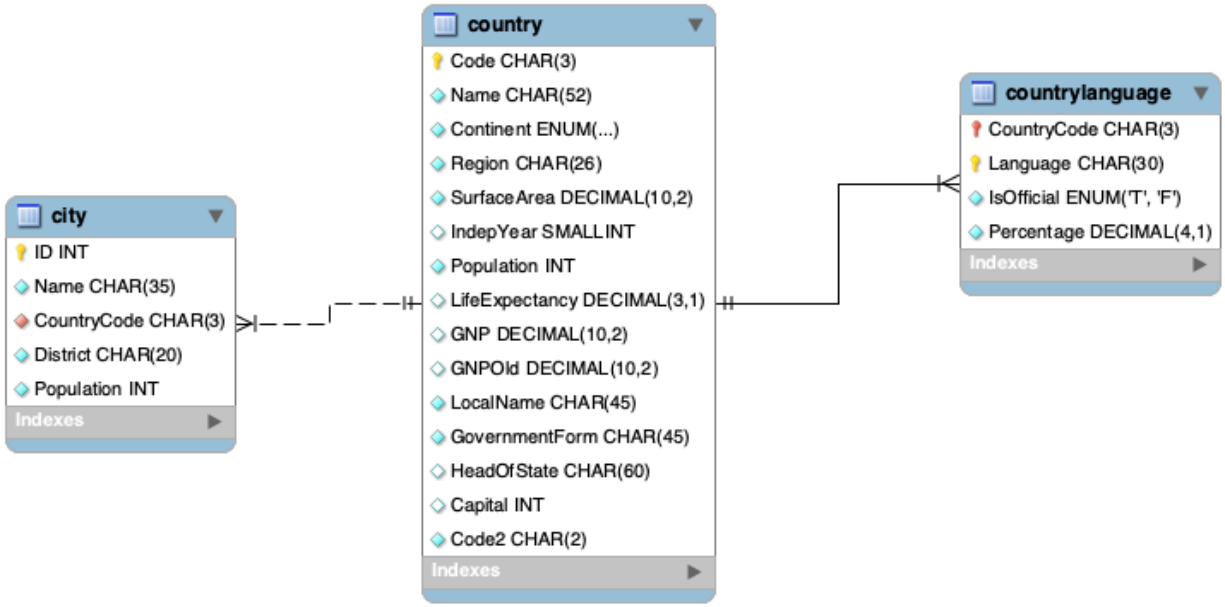
Answer: Mumbai (Bombay)

Within this query, the first line selects the 'Name' column from the 'City' table and using the AS statement I renamed it "Most_Populated_City" to improve understanding and readability. This means that the 'Name' column will be displayed as "Most_Populated_City" when the query is run.

The WHERE statement here filters the results to only include rows where the 'Population' column matches the maximum population value in the 'City' table. The subquery (SELECT MAX(Population) FROM City) finds the maximum population value in the 'City' table, and the outer query selects the city with that population.



EER Diagram



- Identify the primary key in country table: **Code CHAR(3)**
- Identify the primary key in city table: **ID INT**
- Identify the primary key in countrylanguage table: **Language CHAR(30)**
- Identify the foreign key in city table: **CountryCode CHAR(3)**
- Identify the foreign key in countrylanguage table: **CountryCode CHAR(3)**