MySQL Task

Project Assignment 2

Maryam Begum 11/25/24 Databases and SQL

Contents

Task 1 – List the different types of relationships in relational databases and provide examples.

Relationship Type	Examples	Tables and Columns	Description
One-to-One	A country has one government form or head of state	country and GovernmentDetails	A record in one table is related to exactly one record in another table.
One-to-Many (	A country can have multiple languages spoken	country and language	A record in the first table can relate to many records in the second table, but each record in the second table relates to only one record in the first table.
Many-to-Many	A country speaks multiple languages in various regions	country, language, and RegionLanguage	A record in the first table can relate to many records in the second table, and a record in the second table can relate to many records in the first table. This often requires a junction table to manage the relationships.

Task 2 – What is Normalization and why is it important to database development?

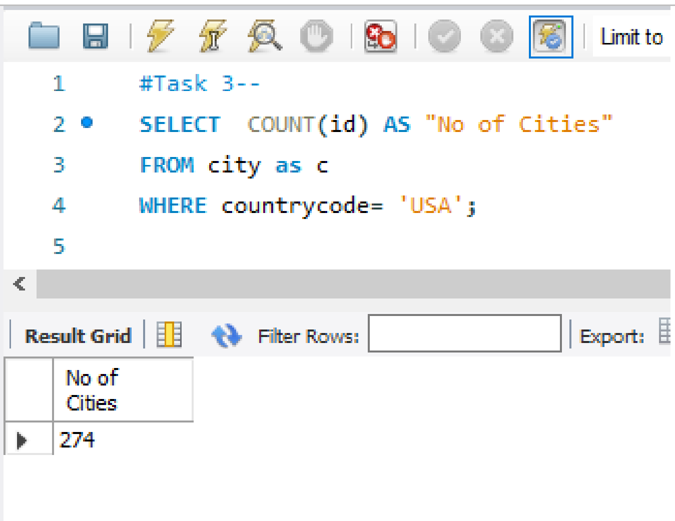
Normalization is the process of organizing data within a database (relational database) to eliminate data anomalies, such as redundancy. In simpler terms, it involves breaking down a large complex table into smaller and simpler tables while maintaining data relationships. Normalization is commonly used when dealing with large datasets.

Normalization plays a crucial role in database design so there are several reasons as to why it is important to database development:

- Reduces Redundancy – Redundancy is when the same information is stored multiple times, and a good way of avoiding this is by splitting data into smaller tables.
- Improves query performance – You can perform faster query execution on smaller tables that have undergone normalization.
- Minimizes update anomalies – With normalized tables, you can easily update data without affecting other records.
- Enhances data integrity – It ensures that data remains consistent and accurate

Task 3- Using count, get the number of cities in the USA

```
SELECT COUNT(id) AS "No of Cities"

FROM city as c

WHERE CountryCode= 'USA';
```

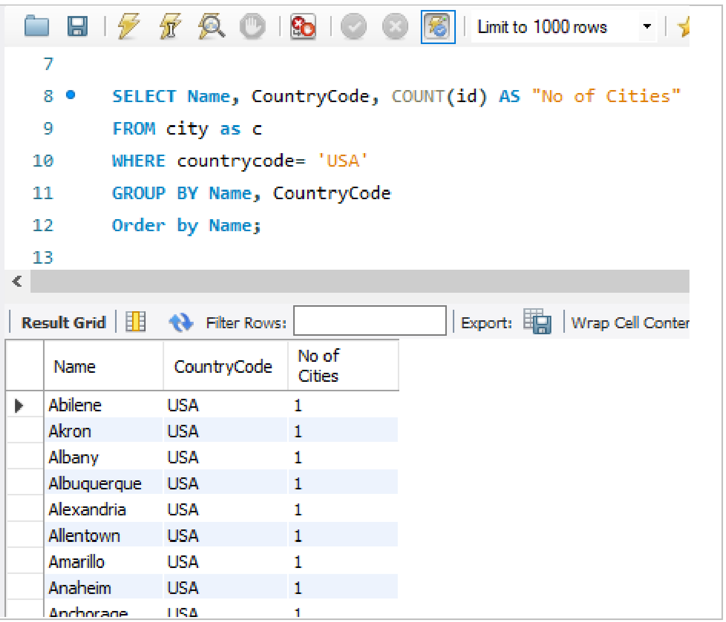


```
#Task 3--
SELECT  COUNT(id) AS "No of Cities"
FROM city as c
WHERE countrycode= 'USA';

```

No of Cities
274

Or

```
SELECT Name, CountryCode,

COUNT(id) AS "No of Cities"

FROM city as c

WHERE countrycode= 'USA'

GROUP BY Name, CountryCode

Order by Name;
```



```

SELECT Name, CountryCode, COUNT(id) AS "No of Cities"
FROM city as c
WHERE countrycode= 'USA'
GROUP BY Name, CountryCode
Order by Name;

```

Name	CountryCode	No of Cities
Abilene	USA	1
Akron	USA	1
Albany	USA	1
Albuquerque	USA	1
Alexandria	USA	1
Allentown	USA	1
Amarillo	USA	1
Anaheim	USA	1
Anchorage	USA	1

Both of these SQL Statements will give you the Number of Cities in the USA, the first statement gives the collective number while the second statement also shows the names of the cities.

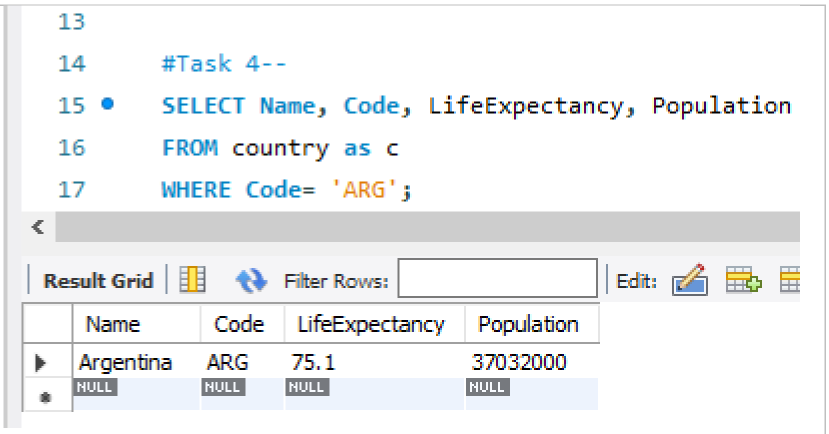
Task 4 – Find out what the population and life expectancy for people in Argentina (ARG) is

SELECT Name, Code,

LifeExpectancy, Population

FROM country as c

WHERE Code= 'ARG';



```
#Task 4--
SELECT Name, Code, LifeExpectancy, Population
FROM country as c
WHERE Code= 'ARG';
```

	Name	Code	LifeExpectancy	Population
▶	Argentina	ARG	75.1	37032000
*	NULL	NULL	NULL	NULL

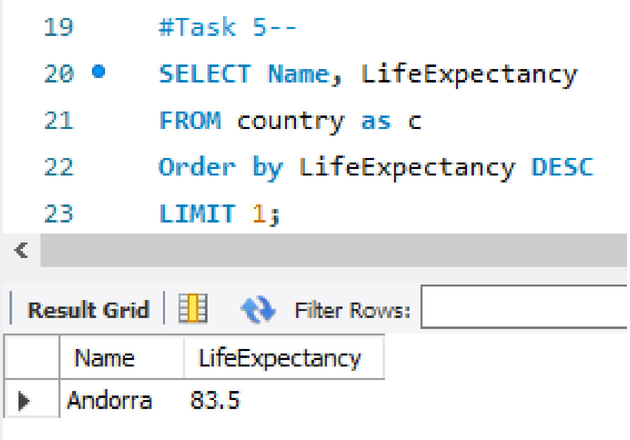
Task 5 – Using ORDER BY, LIMIT, what country has the highest life expectancy?

SELECT Name, LifeExpectancy

FROM country as c

Order by LifeExpectancy DESC

LIMIT 1;



```
#Task 5--
SELECT Name, LifeExpectancy
FROM country as c
Order by LifeExpectancy DESC
LIMIT 1;
```

	Name	LifeExpectancy
▶	Andorra	83.5

Task 6 – Select 25 cities around the world that start with the letter 'F' in a single SQL query.

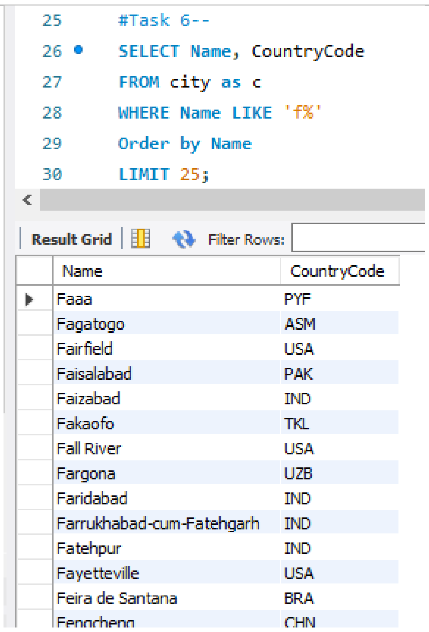
SELECT Name, CountryCode

FROM city as c

WHERE Name LIKE 'f%'

Order by Name

LIMIT 25;

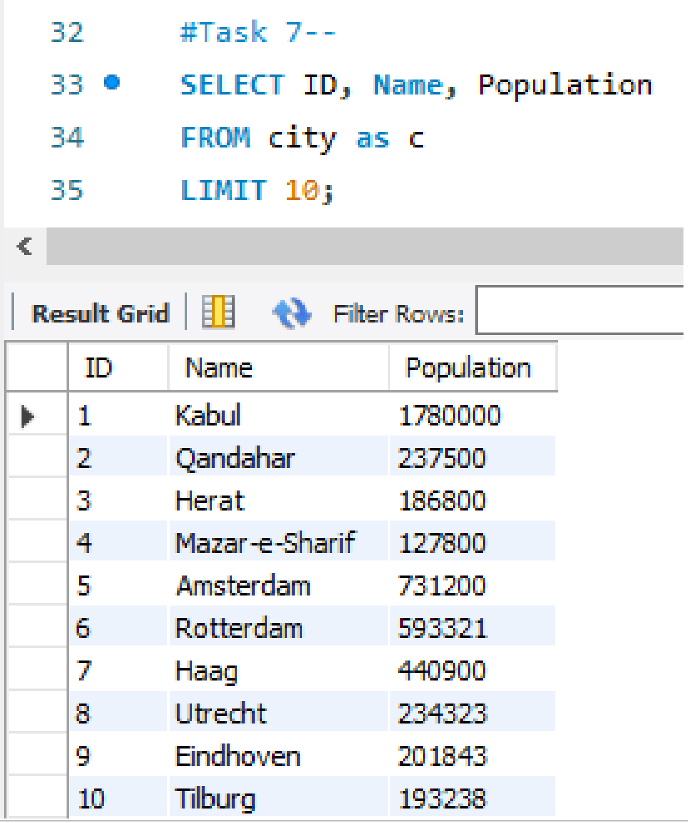


```
#Task 6--
SELECT Name, CountryCode
FROM city as c
WHERE Name LIKE 'f%'
Order by Name
LIMIT 25;
```

	Name	CountryCode
▶	Faaa	PYF
	Fagatogo	ASM
	Fairfield	USA
	Faisalabad	PAK
	Faizabad	IND
	Fakaofo	TKL
	Fall River	USA
	Fargona	UZB
	Faridabad	IND
	Farrukhabad-cum-Fatehgarh	IND
	Fatehpur	IND
	Fayetteville	USA
	Feira de Santana	BRA
	Fengcheng	CHN

Task 7 – Create a SQL statement to display columns Id, Name, Population from the city table and limit results to first 10 rows only.

```
SELECT ID, Name, Population

FROM city as c

LIMIT 10;
```

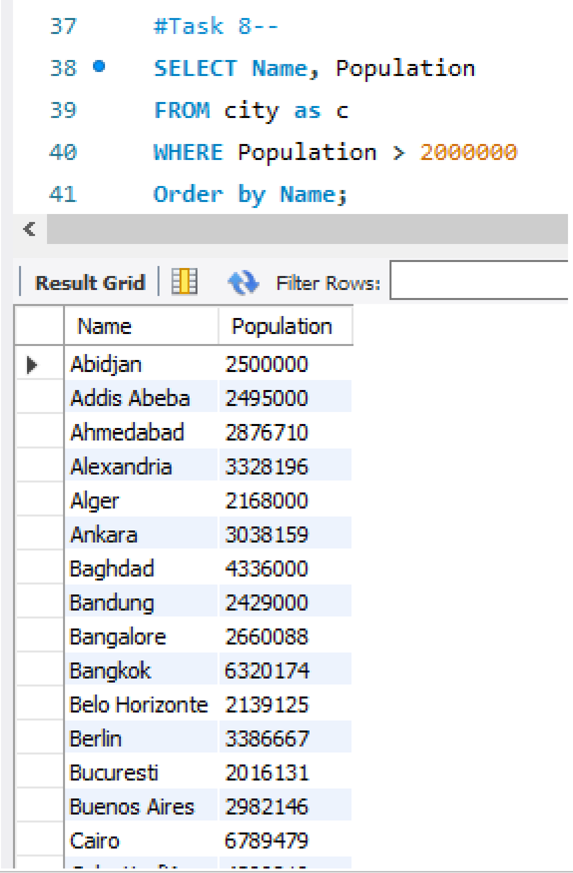


```
#Task 7--
SELECT ID, Name, Population
FROM city as c
LIMIT 10;
```

ID	Name	Population
1	Kabul	1780000
2	Qandahar	237500
3	Herat	186800
4	Mazar-e-Sharif	127800
5	Amsterdam	731200
6	Rotterdam	593321
7	Haag	440900
8	Utrecht	234323
9	Eindhoven	201843
10	Tilburg	193238

Task 8 – Create a SQL statement to find only those cities from city table whose population is larger than 2000000.

```
SELECT Name, Population

FROM city as c

WHERE Population > 2000000

Order by Name;
```

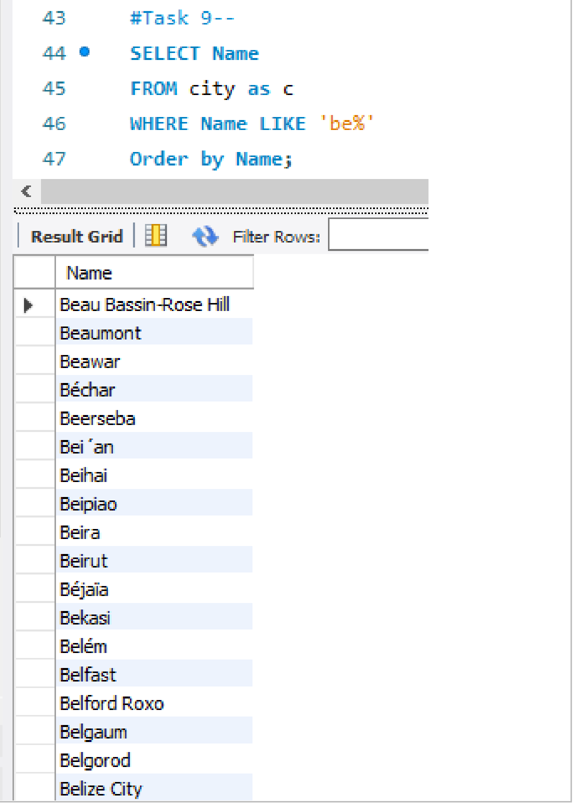


```
#Task 8--
SELECT Name, Population
FROM city as c
WHERE Population > 2000000
Order by Name;
```

Name	Population
Abidjan	2500000
Addis Abeba	2495000
Ahmedabad	2876710
Alexandria	3328196
Alger	2168000
Ankara	3038159
Baghdad	4336000
Bandung	2429000
Bangalore	2660088
Bangkok	6320174
Belo Horizonte	2139125
Berlin	3386667
Bucuresti	2016131
Buenos Aires	2982146
Cairo	6789479

Task 9 – Create a SQL statement to find all city names from city table whose name begins with “Be” prefix.

```
SELECT Name

FROM city as c

WHERE Name LIKE 'be%'

Order by Name;
```

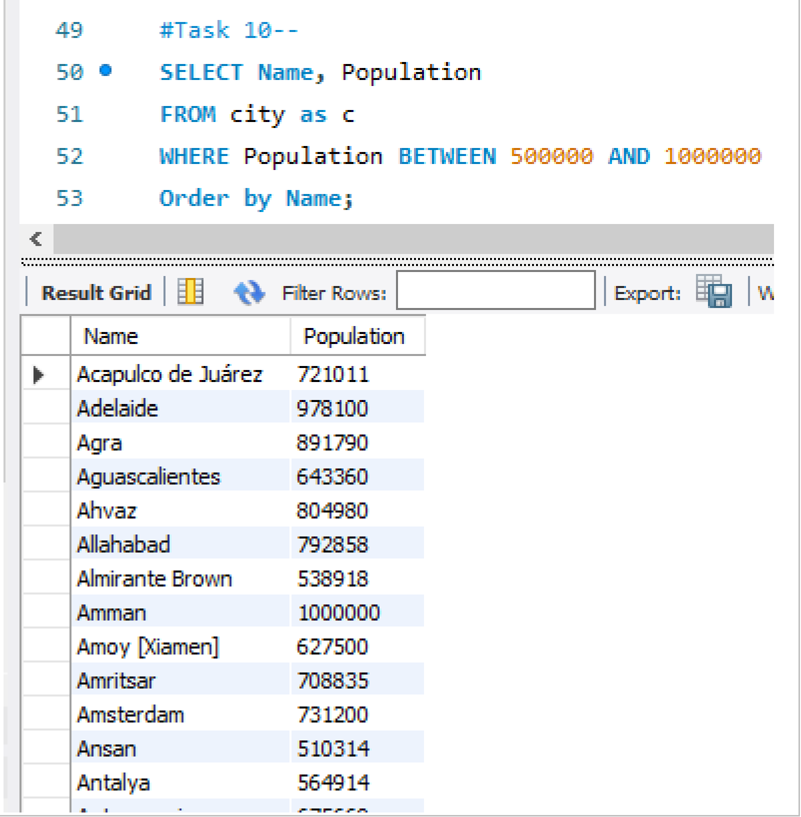


```
#Task 9--
SELECT Name
FROM city as c
WHERE Name LIKE 'be%'
Order by Name;
```

Name
Beau Bassin-Rose Hill
Beaumont
Beawar
Béchar
Beerseba
Bei´an
Beihai
Beipiao
Beira
Beirut
Béjaïa
Bekasi
Belém
Belfast
Belford Roxo
Belgaum
Belgorod
Belize City

Task 10 – Create a SQL statement to find only those cities from city table whose population is between 500000-1000000.

```
SELECT Name, Population

FROM city as c

WHERE Population BETWEEN

500000 AND 1000000

Order by Name;
```



```
#Task 10--
SELECT Name, Population
FROM city as c
WHERE Population BETWEEN 500000 AND 1000000
Order by Name;
```

Name	Population
Acapulco de Juárez	721011
Adelaide	978100
Agra	891790
Aguascalientes	643360
Ahvaz	804980
Allahabad	792858
Almirante Brown	538918
Amman	1000000
Amoy [Xiamen]	627500
Amritsar	708835
Amsterdam	731200
Ansan	510314
Antalya	564914

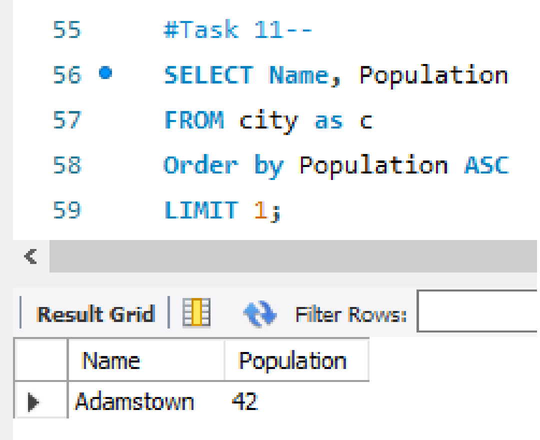
Task 11 – Create a SQL statement to find a city with the lowest population in the city table.

SELECT Name, Population

FROM city as c

Order by Population ASC

LIMIT 1;



```
#Task 11--
SELECT Name, Population
FROM city as c
Order by Population ASC
LIMIT 1;
```

Name	Population
Adamstown	42

Task 12 – Create a SQL statement to show the population of Switzerland and all the languages spoken there.

SELECT Name, Code, Language,

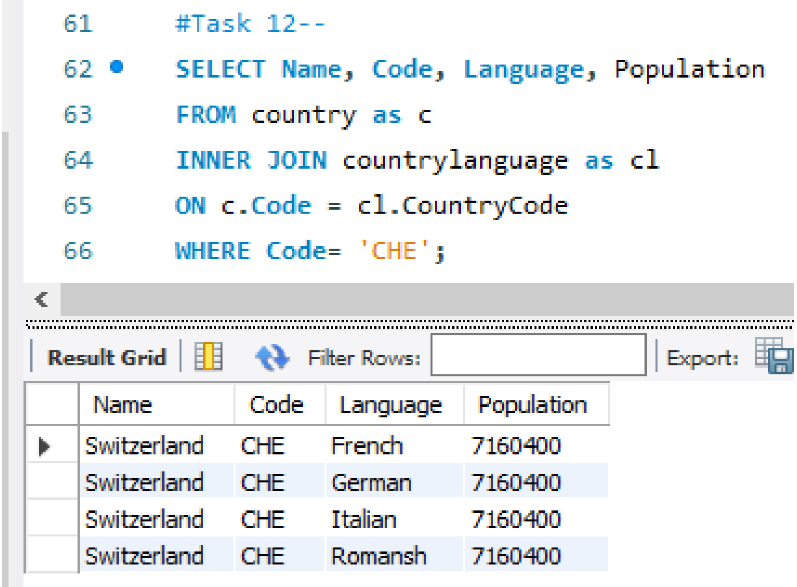
Population

FROM country as c

INNER JOIN countrylanguage as cl

ON c.Code = cl.CountryCode

WHERE Code= 'CHE';



```
#Task 12--
SELECT Name, Code, Language, Population
FROM country as c
INNER JOIN countrylanguage as cl
ON c.Code = cl.CountryCode
WHERE Code= 'CHE';
```

Name	Code	Language	Population
Switzerland	CHE	French	7160400
Switzerland	CHE	German	7160400
Switzerland	CHE	Italian	7160400
Switzerland	CHE	Romansh	7160400

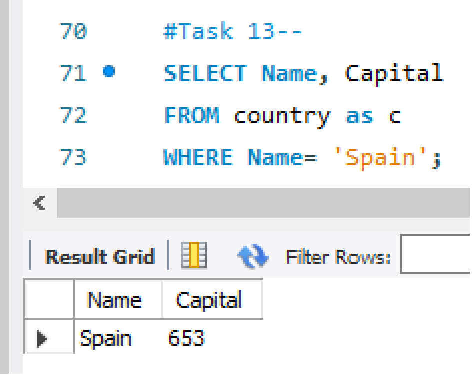
Challenge –

Task 13: Create a SQL statement to find the capital of Spain (ESP).

SELECT Name, Capital

FROM country as c

WHERE Name= 'Spain';

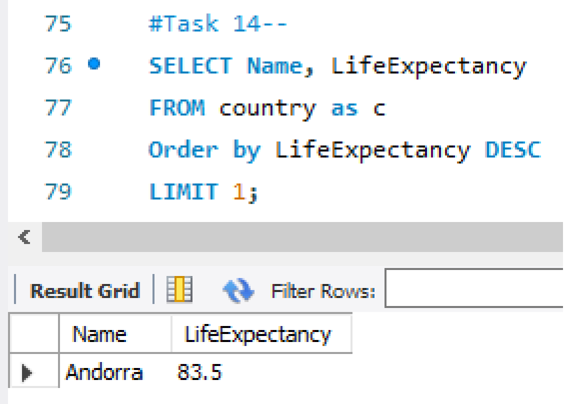


```
#Task 13--
SELECT Name, Capital
FROM country as c
WHERE Name= 'Spain';
```

Name	Capital
Spain	653

Task 14: Create a SQL statement to find the country with the highest life expectancy.

```
SELECT Name, LifeExpectancy

FROM country as c

Order by LifeExpectancy DESC

LIMIT 1;
```

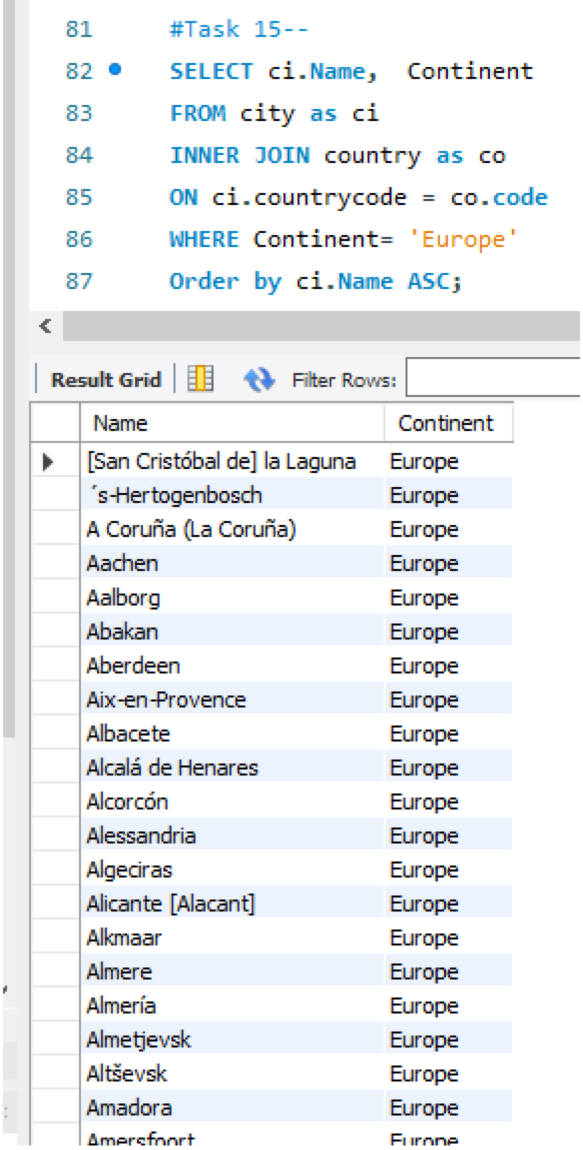


```
#Task 14--
SELECT Name, LifeExpectancy
FROM country as c
Order by LifeExpectancy DESC
LIMIT 1;
```

Name	LifeExpectancy
Andorra	83.5

Task 15: Create a SQL statement to find all cities from the Europe continent.

```
SELECT ci.Name, Continent

FROM city as ci

INNER JOIN country as co

ON ci.countrycode = co.code

WHERE Continent= 'Europe'

Order by ci.Name ASC;
```



```
#Task 15--
SELECT ci.Name,  Continent
FROM city as ci
INNER JOIN country as co
ON ci.countrycode = co.code
WHERE Continent= 'Europe'
Order by ci.Name ASC;
```

Name	Continent
[San Cristóbal de] la Laguna	Europe
´s-Hertogenbosch	Europe
A Coruña (La Coruña)	Europe
Aachen	Europe
Aalborg	Europe
Abakan	Europe
Aberdeen	Europe
Aix-en-Provence	Europe
Albacete	Europe
Alcalá de Henares	Europe
Alcorcón	Europe
Alessandria	Europe
Algeciras	Europe
Alicante [Alacant]	Europe
Alkmaar	Europe
Almere	Europe
Almería	Europe
Almetjevsk	Europe
Altševsk	Europe
Amadora	Europe
Amersfoort	Europe

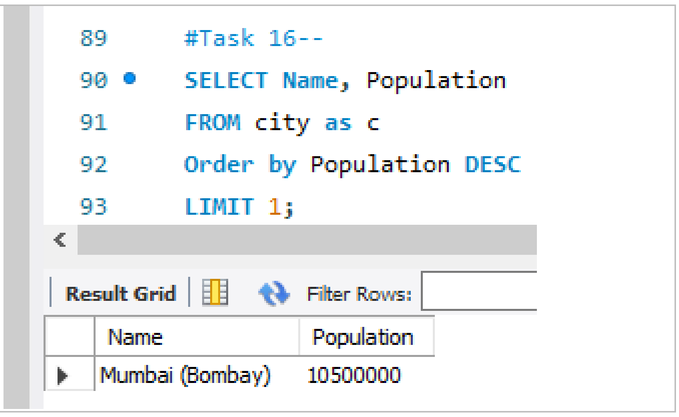
Task 16: Create a SQL statement to find the most populated city in the city table.

SELECT Name, Population

FROM city as c

Order by Population DESC

LIMIT 1;



```
#Task 16--
SELECT Name, Population
FROM city as c
Order by Population DESC
LIMIT 1;
```

Name	Population
Mumbai (Bombay)	10500000

Task 17: Create a SQL statement to find the total population of each continent

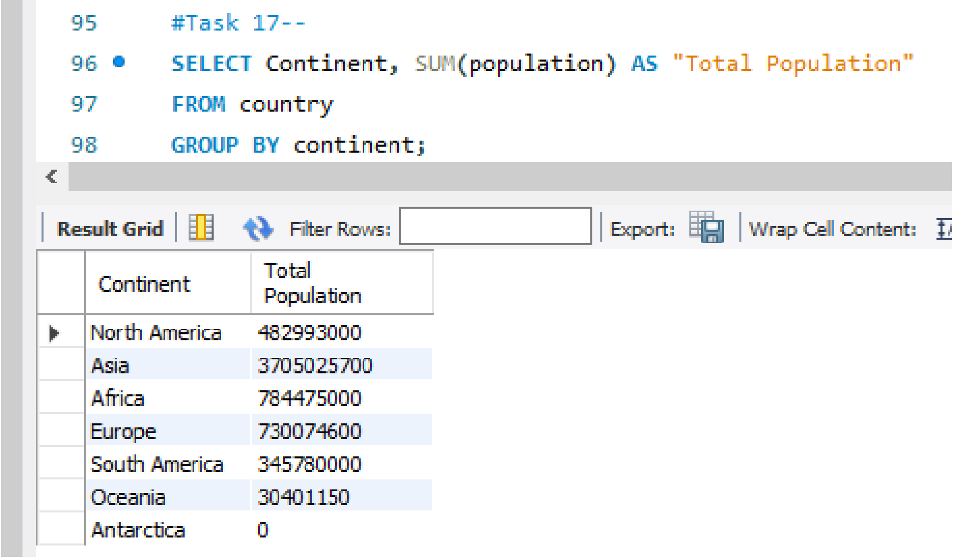
SELECT Continent,

SUM(population) AS

"Total Population"

FROM country

GROUP BY continent;



```
#Task 17--
SELECT Continent, SUM(population) AS "Total Population"
FROM country
GROUP BY continent;
```

Continent	Total Population
North America	482993000
Asia	3705025700
Africa	784475000
Europe	730074600
South America	345780000
Oceania	30401150
Antarctica	0

Task 18: Create a SQL statement to find the average life expectancy by continent

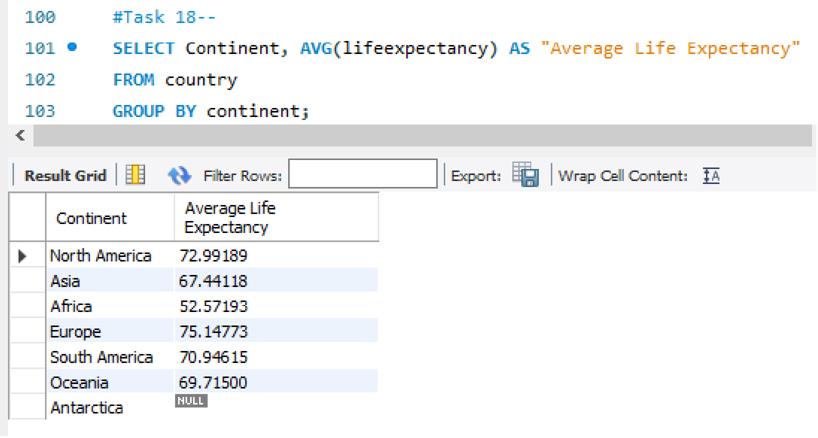
SELECT Continent,

AVG(lifeexpectancy) AS "Average

Life Expectancy"

FROM country

GROUP BY continent;

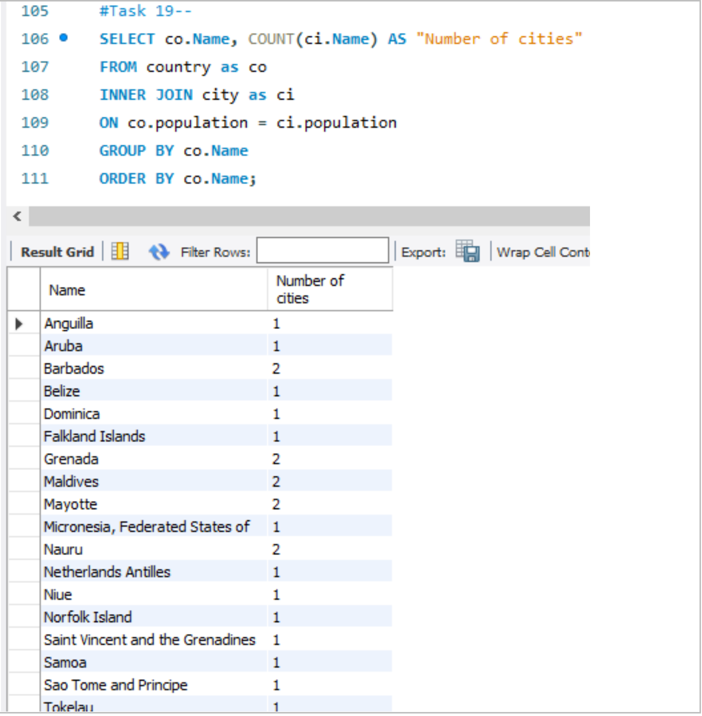


```
#Task 18--
SELECT Continent, AVG(lifeexpectancy) AS "Average Life Expectancy"
FROM country
GROUP BY continent;
```

Continent	Average Life Expectancy
North America	72.99189
Asia	67.44118
Africa	52.57193
Europe	75.14773
South America	70.94615
Oceania	69.71500
Antarctica	NULL

Task 19: Create a SQL statement to list the number of cities in each country

```
SELECT co.Name, COUNT(ci.Name)

AS "Number of cities"

FROM country as co

INNER JOIN city as ci

ON co.population = ci.population

GROUP BY co.Name

ORDER BY co.Name;
```

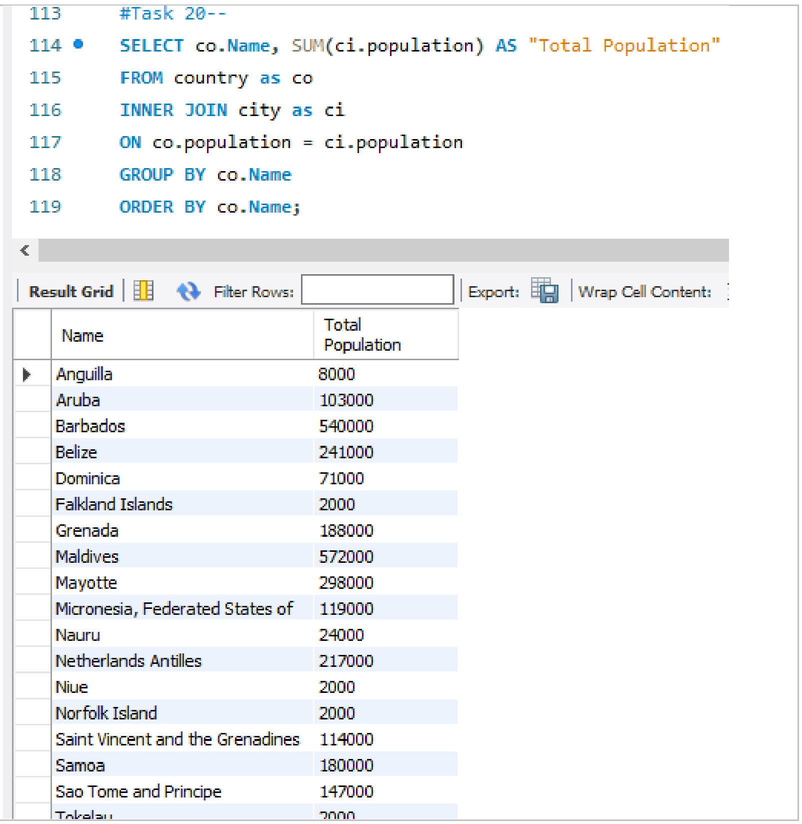


```
#Task 19--
SELECT co.Name, COUNT(ci.Name) AS "Number of cities"
FROM country as co
INNER JOIN city as ci
ON co.population = ci.population
GROUP BY co.Name
ORDER BY co.Name;
```

Name	Number of cities
Anguilla	1
Aruba	1
Barbados	2
Belize	1
Dominica	1
Falkland Islands	1
Grenada	2
Maldives	2
Mayotte	2
Micronesia, Federated States of	1
Nauru	2
Netherlands Antilles	1
Niue	1
Norfolk Island	1
Saint Vincent and the Grenadines	1
Samoa	1
Sao Tome and Principe	1
Tokelau	1

Task 20: Create a SQL statement to find the total population of each country based on its cities

```
SELECT co.Name,

SUM(ci.population) AS "Total

Population"

FROM country as co

INNER JOIN city as ci

ON co.population = ci.population

GROUP BY co.Name

ORDER BY co.Name;
```



```
#Task 20--
SELECT co.Name, SUM(ci.population) AS "Total Population"
FROM country as co
INNER JOIN city as ci
ON co.population = ci.population
GROUP BY co.Name
ORDER BY co.Name;
```

Name	Total Population
Anguilla	8000
Aruba	103000
Barbados	540000
Belize	241000
Dominica	71000
Falkland Islands	2000
Grenada	188000
Maldives	572000
Mayotte	298000
Micronesia, Federated States of	119000
Nauru	24000
Netherlands Antilles	217000
Niue	2000
Norfolk Island	2000
Saint Vincent and the Grenadines	114000
Samoa	180000
Sao Tome and Principe	147000
Tokelau	2000

Task 21: Create a SQL statement to find the most spoken language in each continent

SELECT c.Continent, l.Language,

COUNT(*) AS "Language Count"

FROM country as c

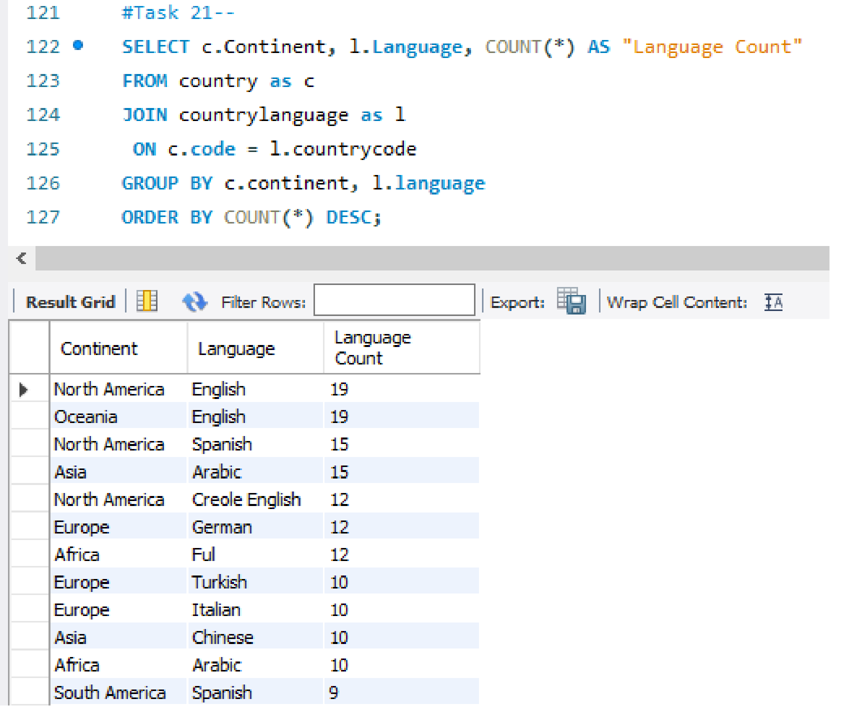
JOIN countrylanguage as l

ON c.code = l.countrycode

GROUP BY c.continent,

l.language

ORDER BY COUNT(*) DESC;



```
#Task 21--
SELECT c.Continent, l.Language, COUNT(*) AS "Language Count"
FROM country as c
JOIN countrylanguage as l
 ON c.code = l.countrycode
GROUP BY c.continent, l.language
ORDER BY COUNT(*) DESC;
```

Continent	Language	Language Count
North America	English	19
Oceania	English	19
North America	Spanish	15
Asia	Arabic	15
North America	Creole English	12
Europe	German	12
Africa	Ful	12
Europe	Turkish	10
Europe	Italian	10
Asia	Chinese	10
Africa	Arabic	10
South America	Spanish	9

Task 22: Create a SQL statement to find countries where the official language is either 'English', 'Spanish', or 'French'

SELECT c.name AS

"Country Name",

l.Language

FROM country AS c

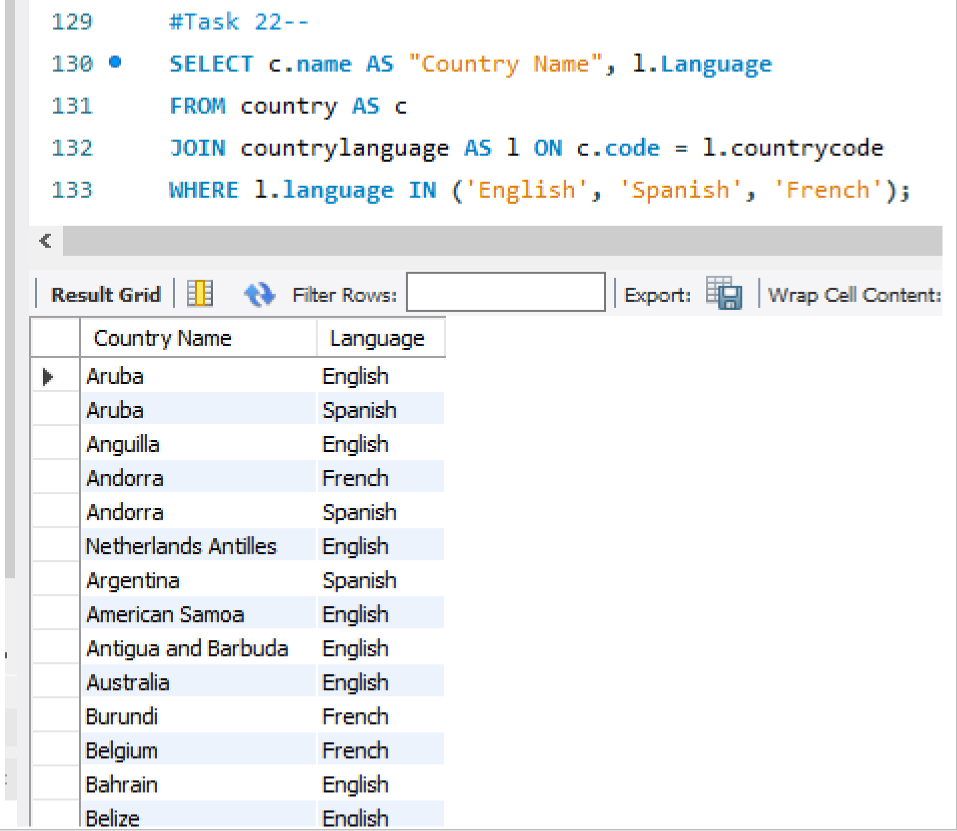
JOIN countrylanguage AS l

ON c.code = l.countrycode

WHERE l.language IN

('English', 'Spanish',

'French');

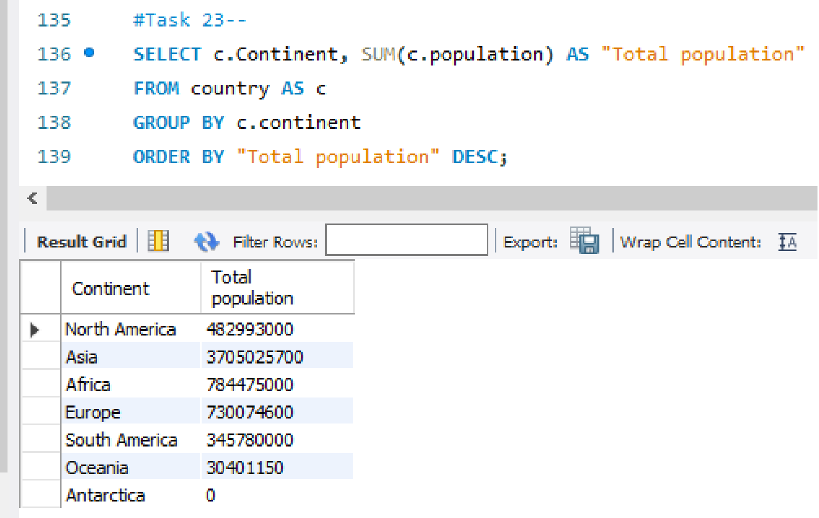


```
#Task 22--
SELECT c.name AS "Country Name", l.Language
FROM country AS c
JOIN countrylanguage AS l ON c.code = l.countrycode
WHERE l.language IN ('English', 'Spanish', 'French');
```

Country Name	Language
Aruba	English
Aruba	Spanish
Anguilla	English
Andorra	French
Andorra	Spanish
Netherlands Antilles	English
Argentina	Spanish
American Samoa	English
Antigua and Barbuda	English
Australia	English
Burundi	French
Belgium	French
Bahrain	English
Belize	English

Task 23: Write a query to display the total population for each continent.

```
SELECT c.Continent,
SUM(c.population) AS "Total population"
FROM country AS c
GROUP BY c.continent
ORDER BY "Total population" DESC;
```

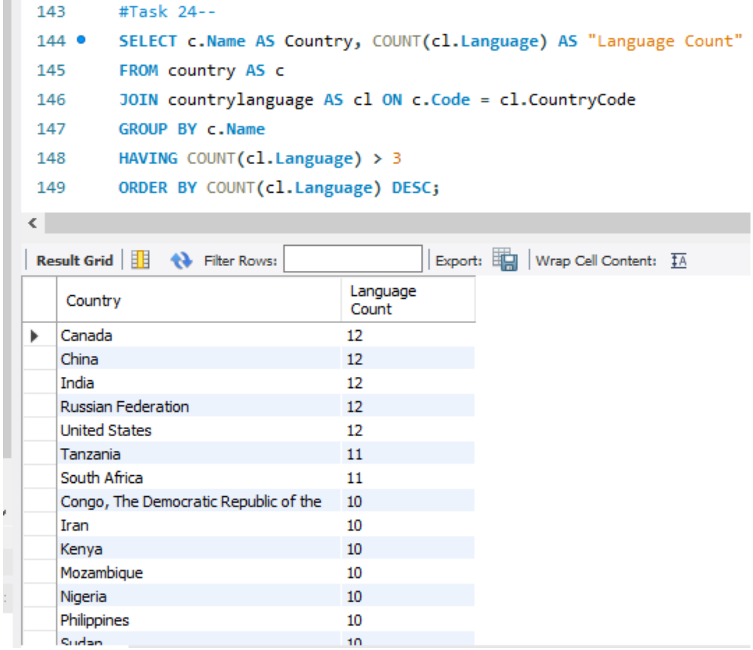


```
#Task 23--
SELECT c.Continent, SUM(c.population) AS "Total population"
FROM country AS c
GROUP BY c.continent
ORDER BY "Total population" DESC;
```

Continent	Total population
North America	482993000
Asia	3705025700
Africa	784475000
Europe	730074600
South America	345780000
Oceania	30401150
Antarctica	0

Task 24: Write a query to list countries that have more than three official languages. (joins, group by, having)

```
SELECT c.Name AS Country,
COUNT(cl.Language) AS "Language Count"
FROM country AS c
JOIN countrylanguage AS cl ON c.Code = cl.CountryCode
GROUP BY c.Name
HAVING COUNT(cl.Language) > 3
ORDER BY COUNT(cl.Language) DESC;
```

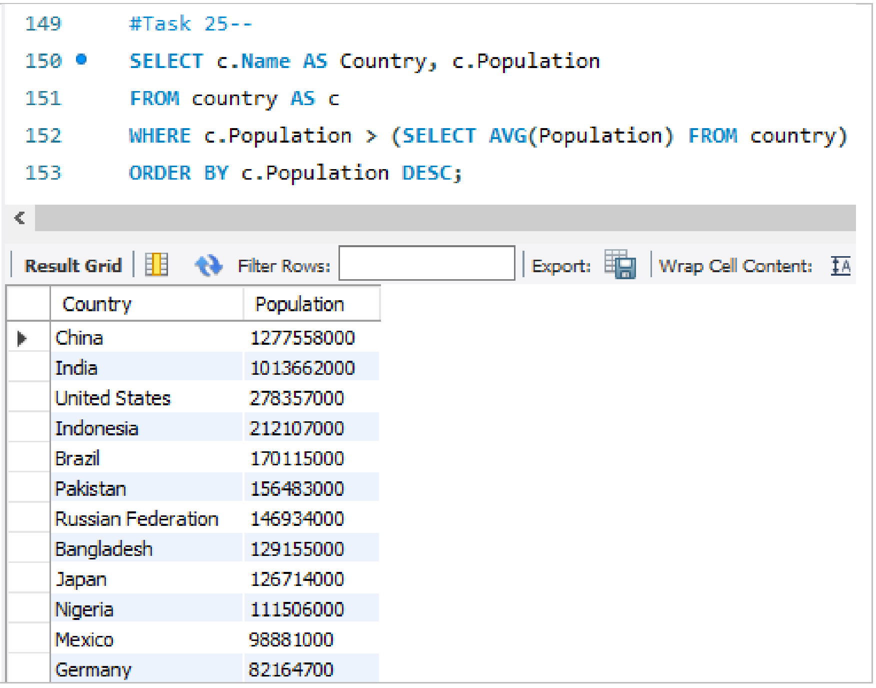


```
#Task 24--
SELECT c.Name AS Country, COUNT(cl.Language) AS "Language Count"
FROM country AS c
JOIN countrylanguage AS cl ON c.Code = cl.CountryCode
GROUP BY c.Name
HAVING COUNT(cl.Language) > 3
ORDER BY COUNT(cl.Language) DESC;
```

Country	Language Count
Canada	12
China	12
India	12
Russian Federation	12
United States	12
Tanzania	11
South Africa	11
Congo, The Democratic Republic of the	10
Iran	10
Kenya	10
Mozambique	10
Nigeria	10
Philippines	10
Sudan	10

Task 25: Find countries whose population is greater than the average population of all countries.

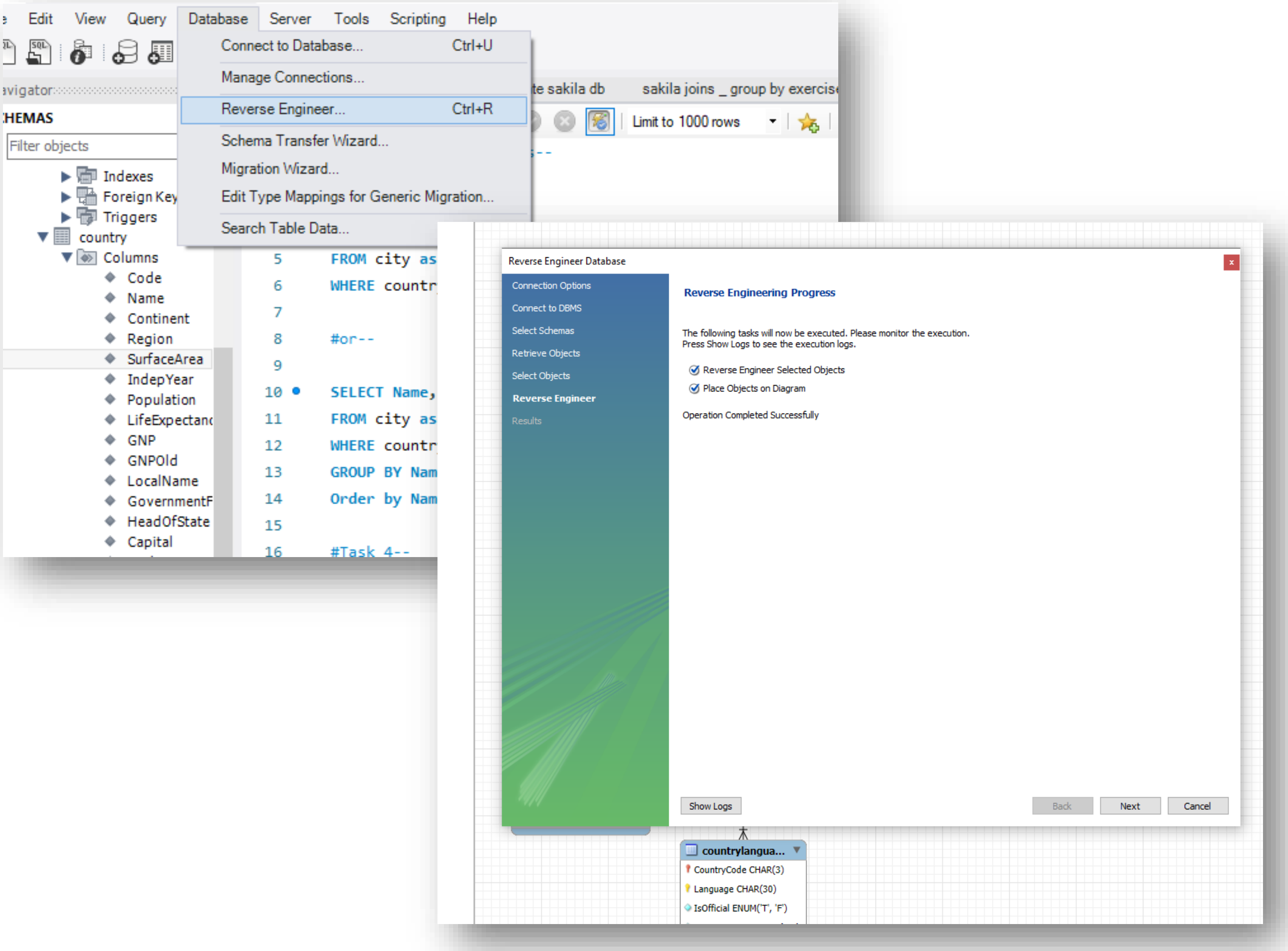
```
SELECT c.Name AS Country,

c.Population

FROM country AS c

WHERE c.Population > (SELECT

AVG(Population) FROM

country)

ORDER BY c.Population DESC;
```



```
#Task 25--
SELECT c.Name AS Country, c.Population
FROM country AS c
WHERE c.Population > (SELECT AVG(Population) FROM country)
ORDER BY c.Population DESC;
```

Country	Population
China	1277558000
India	1013662000
United States	278357000
Indonesia	212107000
Brazil	170115000
Pakistan	156483000
Russian Federation	146934000
Bangladesh	129155000
Japan	126714000
Nigeria	111506000
Mexico	98881000
Germany	82164700

Task 26 – Creating an EER Diagram



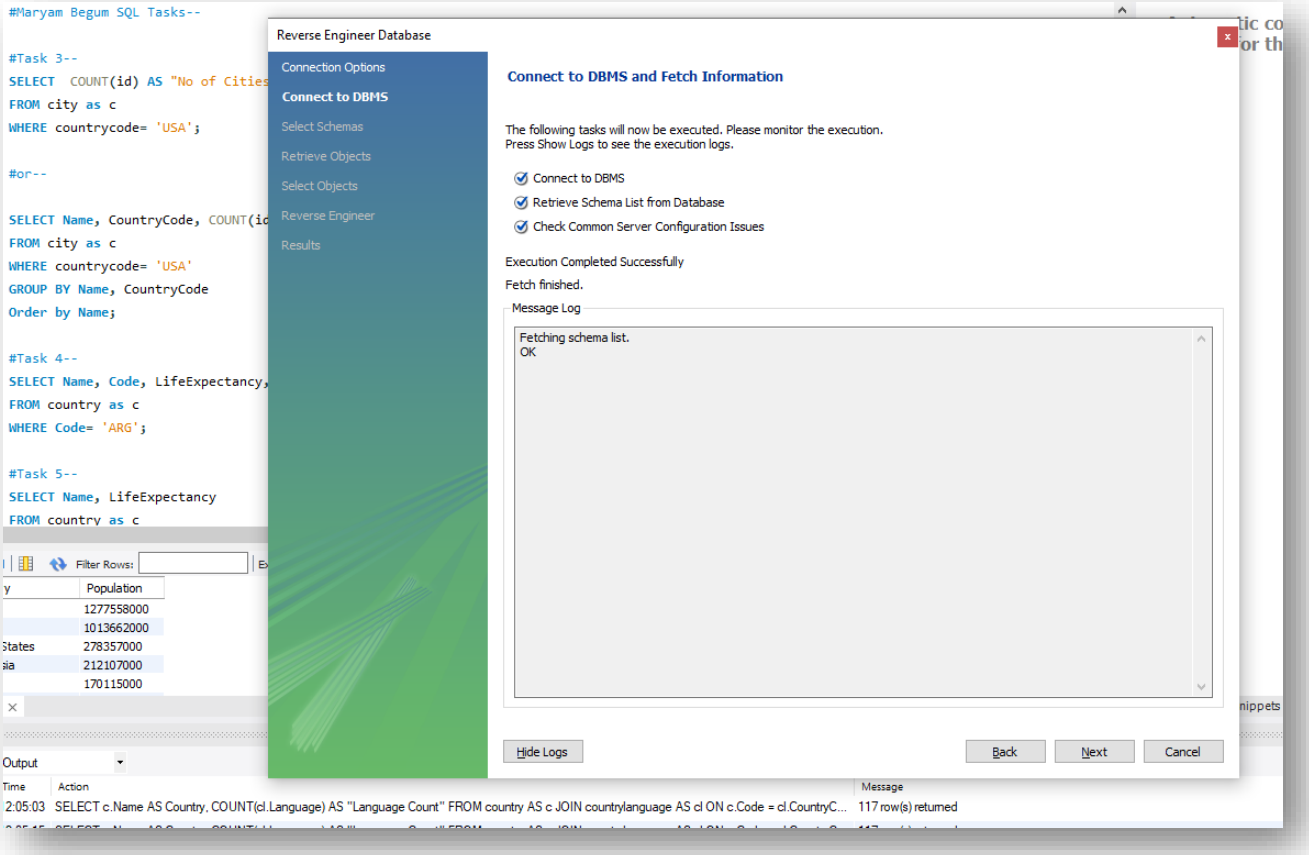
```
FROM city as
WHERE countr

#or--

SELECT Name,
FROM city as
WHERE countr
GROUP BY Nam
Order by Nam

#Task 4--
```

Reverse Engineering Progress

The following tasks will now be executed. Please monitor the execution.
Press Show Logs to see the execution logs.

- ☑ Reverse Engineer Selected Objects
- ☑ Place Objects on Diagram

Operation Completed Successfully



```
#Maryam Begum SQL Tasks--

#Task 3--
SELECT  COUNT(id) AS "No of Cities
FROM city as c
WHERE countrycode= 'USA';

#or--

SELECT Name, CountryCode, COUNT(id
FROM city as c
WHERE countrycode= 'USA'
GROUP BY Name, CountryCode
Order by Name;

#Task 4--
SELECT Name, Code, LifeExpectancy,
FROM country as c
WHERE Code= 'ARG';

#Task 5--
SELECT Name, LifeExpectancy
FROM country as c
```

y	Population
	1277558000
	1013662000
States	278357000
ia	212107000
	170115000

Reverse Engineer Database

Connect to DBMS and Fetch Information

The following tasks will now be executed. Please monitor the execution.
Press Show Logs to see the execution logs.

☑ Connect to DBMS

☑ Retrieve Schema List from Database

☑ Check Common Server Configuration Issues

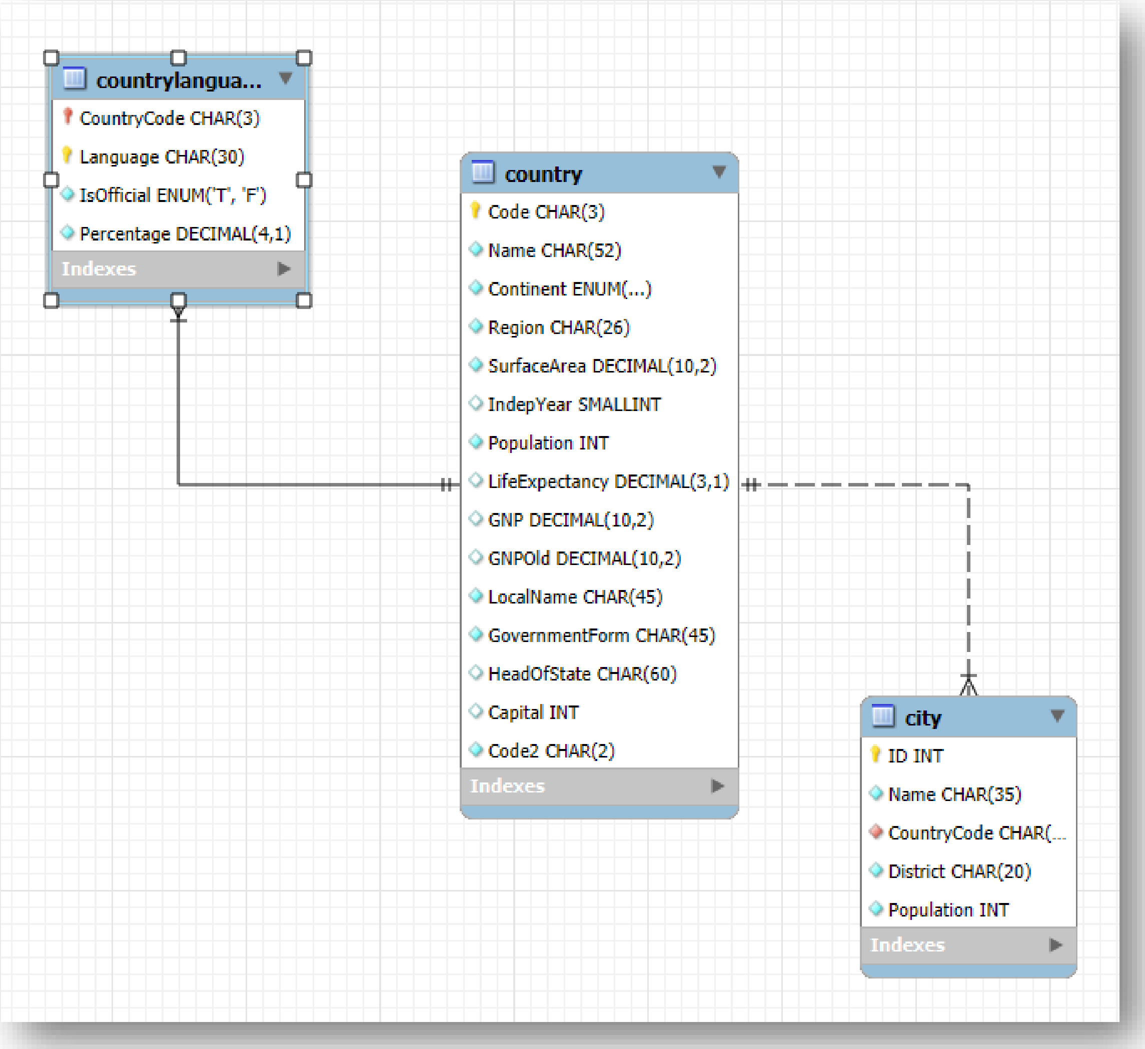
Execution Completed Successfully

Fetch finished.

Message Log

```
Fetching schema list.
OK
```

```
Time     Action                                                                                                          Message
2:05:03  SELECT c.Name AS Country, COUNT(cl.Language) AS "Language Count" FROM country AS c JOIN countrylanguage AS cl ON c.Code = cl.CountryC...  117 row(s) returned
```



Task 27

What is the Primary Key?

The Primary Key uniquely identifies each record in a table. Primary keys must contain UNIQUE values and cannot contain NULL values. A table can only have ONE primary key and this primary key can consist of single or multiple columns (fields)

What is the Foreign Key?

The foreign key is used to prevent actions that would destroy links between tables. A foreign key is a field (or collection of fields) in one table that refers to the Primary key in another table. The table with the foreign key is called the child table and the table with the primary key is called the referenced table or parent table.

Identify the primary key in country table.

Code - This would be the primary key because it uniquely identifies each country. No two countries will share the same Code, making it a natural candidate for the primary key.

Identify the primary key in city table.

ID - The ID is likely a unique integer that identifies each city record. Since city names (Name) and CountryCode can repeat across different records, ID is chosen as the primary key for uniqueness.

Identify the primary key in countrylanguage table.

Language – the most unique value in the table, that doesn’t repeat in any other.

Identify the foreign key in city table.

CountryCode: This references the Code column in the country table. Each city is associated with one country, and CountryCode ensures the link to the country.

Identify the foreign key in countrylanguage table.

CountryCode – This column is a foreign key that references the Code column in the country table. Each record in the countrylanguage table associates a country with a specific language, and CountryCode ensures the connection to the country.