

# MySQL Assignment

Alex Thompson

04/11/2024



JustIT  
Data Technician Bootcamp

## Table of Contents

• Table of Contents .....	1
• Task 1 and Task 2 .....	2
• Task 3 and Task 4 .....	3
• Task 5 and Task 6 .....	4
• Task 7 and Task 8 .....	5
• Task 9 and Task 10 .....	6
• Task 11 and Task 12 .....	7
• Task 13 and Task 14 .....	8
• Task 15 and Task 16 .....	9
• Task 17 and Task 18 .....	10
• Task 19 and Task 20 .....	11
• Task 21 and Task 22 .....	12
• Task 23 and Task 24 .....	13
• Task 25 and Task 26 .....	14

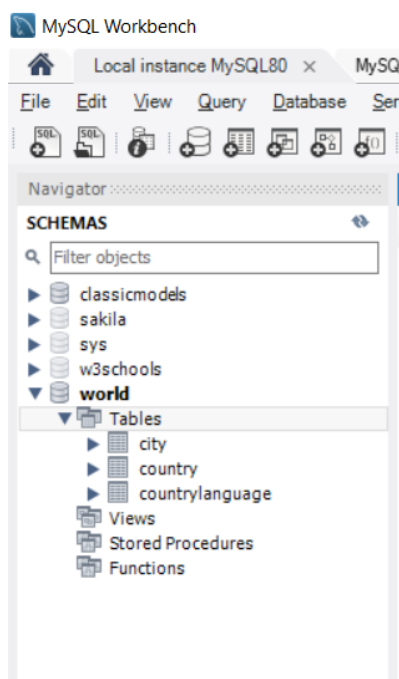
## Task 1: List the different type of relationships in relational databases and provide examples.

1. **One-to-one relationship:** A relationship where each record in one table is linked to one and only one record in another table. For example, a person can have only one passport, and a passport belongs to only one person.
2. **One-to-many relationship:** A relationship where each record in one table is linked to one or more records in another table. For example, a customer can place many orders, and an order belongs to only one customer.
3. **Many-to-many relationship:** A relationship where each record in one table is linked to one or more records in another table, and vice versa. For example, a student can enrol in many courses, and a course can have many students. This type of relationship requires a bridge table to store the links between the two tables.

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

Data normalization is the process of reducing data redundancy, duplication and improving integrity within a table or database. Data normalization ensures that your data remains clean, consistent, and error-free by breaking it into smaller tables and linking them through relationships. This also works to improve database performance and protect against errors appearing within the database as data is changed or updated.

### Set Up:



# SQL Tasks

**Task 3:** Using `count`, get the number of cities in the USA.

The screenshot shows the MySQL Workbench interface. The top toolbar includes icons for file operations, execution, and a dropdown menu set to 'Don't Limit'. The SQL editor contains the following code:

```
1 #Task 3
2 • SELECT COUNT(name) FROM city WHERE CountryCode = "USA";
```

Below the editor, the 'Result Grid' tab is active, displaying the results of the query in a table:

COUNT(name)
274

**Task 4:** Find out what the population and life expectancy for people in Argentina is.

The screenshot shows the MySQL Workbench interface. The top toolbar includes icons for file operations, execution, and a dropdown menu set to 'Don't Limit'. The SQL editor contains the following code:

```
1 #Task 3
2 • SELECT COUNT(name) FROM city WHERE CountryCode = "USA";
3
4 #Task 4
5 • SELECT Name, Population, LifeExpectancy FROM country WHERE Code = "ARG";
```

Below the editor, the 'Result Grid' tab is active, displaying the results of the query in a table:

Name	Population	LifeExpectancy
Argentina	37032000	75.1

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

The screenshot shows the MySQL Workbench interface with a query editor containing three tasks. Task 5 is highlighted, showing a query to select the country with the highest life expectancy. The result grid below shows the output of this query.

```
1 #Task 3
2 • SELECT COUNT(name) FROM city WHERE CountryCode = "USA";
3
4 #Task 4
5 • SELECT Name, Population, LifeExpectancy FROM country WHERE Code = "ARG";
6
7 #Task 5
8 • SELECT Name, LifeExpectancy FROM country ORDER BY LifeExpectancy DESC LIMIT 1;
```

Result Grid

Name	LifeExpectancy
Andorra	83.5

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

The screenshot shows the MySQL Workbench interface with a query editor containing six tasks. Task 6 is highlighted, showing a query to select 25 cities starting with the letter 'F'. The result grid below shows the output of this query.

```
1 #Task 3
2 • SELECT COUNT(name) FROM city WHERE CountryCode = "USA";
3
4 #Task 4
5 • SELECT Name, Population, LifeExpectancy FROM country WHERE Code = "ARG";
6
7 #Task 5
8 • SELECT Name, LifeExpectancy FROM country ORDER BY LifeExpectancy DESC LIMIT 1;
9
10 #Task 6
11 • SELECT Name FROM city WHERE Name LIKE "F%" ORDER BY Name LIMIT 25;
```

Result Grid

Name
Faaa
Fagatogo
Fairfield
Faisalabad
Faizabad
Fakaofo
Fall River
Fargona
Faridabad
Farrukhabad-cum-Fatehgarh
Fatehgarh

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

The screenshot shows the MySQL Workbench interface with the 'city' table selected. The SQL editor displays several tasks, with Task 7 being the active one. The result grid below shows the top 10 most populous cities from the 'city' table, ordered by population in descending order.

```

#Task 4
5 • SELECT Name, Population, LifeExpectancy FROM country WHERE Code = "ARG";
6
#Task 5
8 • SELECT Name, LifeExpectancy FROM country ORDER BY LifeExpectancy DESC LIMIT 1;
9
#Task 6
11 • SELECT Name FROM city WHERE Name LIKE "F%" ORDER BY Name LIMIT 25;
12
#Task 7
14 • SELECT Id, Name, Population FROM city ORDER BY Population DESC LIMIT 10;

```

Id	Name	Population
1024	Mumbai (Bombay)	10500000
2331	Seoul	9981619
206	São Paulo	9968485
1890	Shanghai	9696300
939	Jakarta	9604900
2822	Karachi	9269265
3357	Istanbul	8787958
2515	Ciudad de México	8591309
3580	Moscow	8389200
3793	New York	8008278

Note: As this task doesn't specify how the data is ordered I've chosen to show the 10 most populous cities.

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

The screenshot shows the MySQL Workbench interface with the 'city' table selected. The SQL editor displays several tasks, with Task 8 being the active one. The result grid below shows the cities from the 'city' table where the population is greater than 2,000,000, ordered by population in descending order.

```

#Task 5
8 • SELECT Name, LifeExpectancy FROM country ORDER BY LifeExpectancy DESC LIMIT 1;
9
#Task 6
11 • SELECT Name FROM city WHERE Name LIKE "F%" ORDER BY Name LIMIT 25;
12
#Task 7
14 • SELECT Id, Name, Population FROM city ORDER BY Population DESC LIMIT 10;
15
#Task 8
17 • SELECT Name, Population FROM city where population > 2000000 ORDER BY Population DESC;

```

Name	Population
Mumbai (Bombay)	10500000
Seoul	9981619
São Paulo	9968485
Shanghai	9696300
Jakarta	9604900
Karachi	9269265
Istanbul	8787958
Ciudad de México	8591309
Moscow	8389200
New York	8008278

**Task 9:** Create a SQL statement to find all city names from `city` table whose name begins with 'Be' prefix.

The screenshot shows the MySQL Workbench interface with the following SQL tasks:

```

10 #Task 6
11 • SELECT Name FROM city WHERE Name LIKE "F%" ORDER BY Name LIMIT 25;
12
13 #Task 7
14 • SELECT Id, Name, Population FROM city ORDER BY Population DESC LIMIT 10;
15
16 #Task 8
17 • SELECT Name, Population FROM city where population > 2000000 ORDER BY Population DESC;
18
19 #Task 9
20 • SELECT Name FROM city where Name LIKE "Be%";

```

The result grid for Task 9 shows the following cities:

Name
Béjaia
Béchar
Benguela
Berazategui
Belize City
Belmopan
Belo Horizonte
Belém
Belford Roxo
Betim
Ranin Goncal

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

The screenshot shows the MySQL Workbench interface with the following SQL tasks:

```

13 #Task 7
14 • SELECT Id, Name, Population FROM city ORDER BY Population DESC LIMIT 10;
15
16 #Task 8
17 • SELECT Name, Population FROM city where population > 2000000 ORDER BY Population DESC;
18
19 #Task 9
20 • SELECT Name FROM city where Name LIKE "Be%";
21
22 #Task 10
23 • SELECT Name, Population FROM city WHERE Population BETWEEN 500000 AND 1000000 ORDER BY Population DESC;

```

The result grid for Task 10 shows the following cities and their populations:

Name	Population
Amman	1000000
Mogadishu	997000
Volgograd	993400
Sendai	989975
Peshawar	988005
Baotou	980000
Adelaide	978100
Madurai	977856
Mekka	965700
Köln	962507
Manama	950000

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

The screenshot shows the MySQL Workbench interface with the 'city' table selected. The SQL editor contains the following queries for tasks 8 through 11:

```

16 #Task 8
17 • SELECT Name, Population FROM city where population > 2000000 ORDER BY Population DESC;
18
19 #Task 9
20 • SELECT Name FROM city where Name LIKE "Be%";
21
22 #Task 10
23 • SELECT Name, Population FROM city WHERE Population BETWEEN 500000 AND 1000000 ORDER BY Population DESC;
24
25 #Task 11
26 • SELECT Name, Population FROM city ORDER BY Population LIMIT 1;
  
```

The result grid below the editor shows the output of Task 11:

Name	Population
Adamstown	42

**Task 12:** Create a SQL statement to show the population of [Switzerland](#) and all the languages spoken there.

The screenshot shows the MySQL Workbench interface with the 'country' and 'countrylanguage' tables selected. The SQL editor contains the following queries for tasks 11 and 12:

```

23 • SELECT Name, Population FROM city WHERE Population BETWEEN 500000 AND
24
25 #Task 11
26 • SELECT Name, Population FROM city ORDER BY Population LIMIT 1;
27
28 #Task 12
29 • SELECT country.Name, country.Population, countrylanguage.Language
30 FROM country
31 JOIN countrylanguage
32 ON country.Code = countrylanguage.CountryCode
33 WHERE country.Code = "CHE";
  
```

The result grid below the editor shows the output of Task 12:

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



**Task 13:** Create a SQL statement to find the capital of Spain.

The screenshot shows the MySQL Workbench interface with a query editor and a result grid. The query editor contains the following SQL code:

```
30 FROM country
31 JOIN countrylanguage
32 ON country.Code = countrylanguage.CountryCode
33 WHERE country.Code = "CHE";
34
35 #Task 13
36 • SELECT country.Name, city.Name
37 FROM country
38 JOIN city
39 ON country.Capital = city.ID
40 WHERE country.Name = "Spain";
```

The result grid shows the following data:

Name	Name
Spain	Madrid

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

The screenshot shows the MySQL Workbench interface with a query editor and a result grid. The query editor contains the following SQL code:

```
36 • SELECT country.Name, city.Name
37 FROM country
38 JOIN city
39 ON country.Capital = city.ID
40 WHERE country.Name = "Spain";
41
42 #Task 14
43 • SELECT Name, LifeExpectancy
44 FROM country
45 ORDER BY LifeExpectancy DESC
46 LIMIT 1;
```

The result grid shows the following data:

Name	LifeExpectancy
Andorra	83.5

**Task 15:** Create a SQL statement to find all cities from the [Europe](#) continent.

The screenshot shows the MySQL Workbench interface with a SQL editor and a result grid. The SQL editor contains the following code:

```
#Task 14
42 • SELECT Name, LifeExpectancy
43 FROM country
44 ORDER BY LifeExpectancy DESC;
45
46
47 #Task 15
48 • SELECT city.Name AS City, country.Name AS Country
49 FROM country
50 JOIN city
51 ON country.Code = city.CountryCode
52 WHERE country.Continent = "Europe";
```

The result grid shows the following data:

City	Country
Tirana	Albania
Andorra la Vella	Andorra
Wien	Austria
Graz	Austria
Linz	Austria
Salzburg	Austria
Innsbruck	Austria
Klagenfurt	Austria
Antwerpen	Belgium
Gent	Belgium
Charleroi	Belgium

**Task 16:** Create a SQL statement to find the most populated city in the [city](#) table.

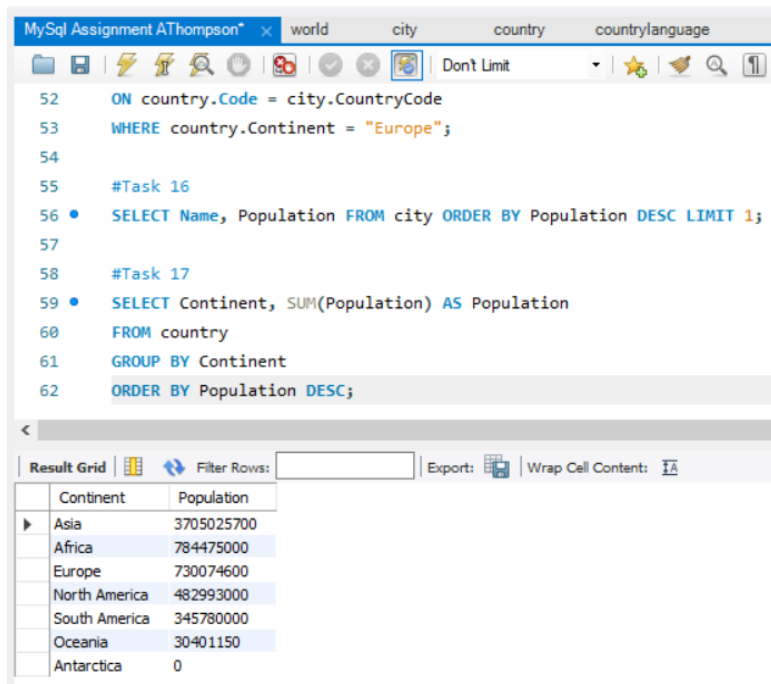
The screenshot shows the MySQL Workbench interface with a SQL editor and a result grid. The SQL editor contains the following code:

```
46 LIMIT 1;
47
48 #Task 15
49 • SELECT city.Name AS City, country.Name AS Country
50 FROM country
51 JOIN city
52 ON country.Code = city.CountryCode
53 WHERE country.Continent = "Europe";
54
55 #Task 16
56 • SELECT Name, Population FROM city ORDER BY Population DESC LIMIT 1;
```

The result grid shows the following data:

Name	Population
Mumbai (Bombay)	10500000

**Task 17:** Create a SQL statement to find population of each continent.



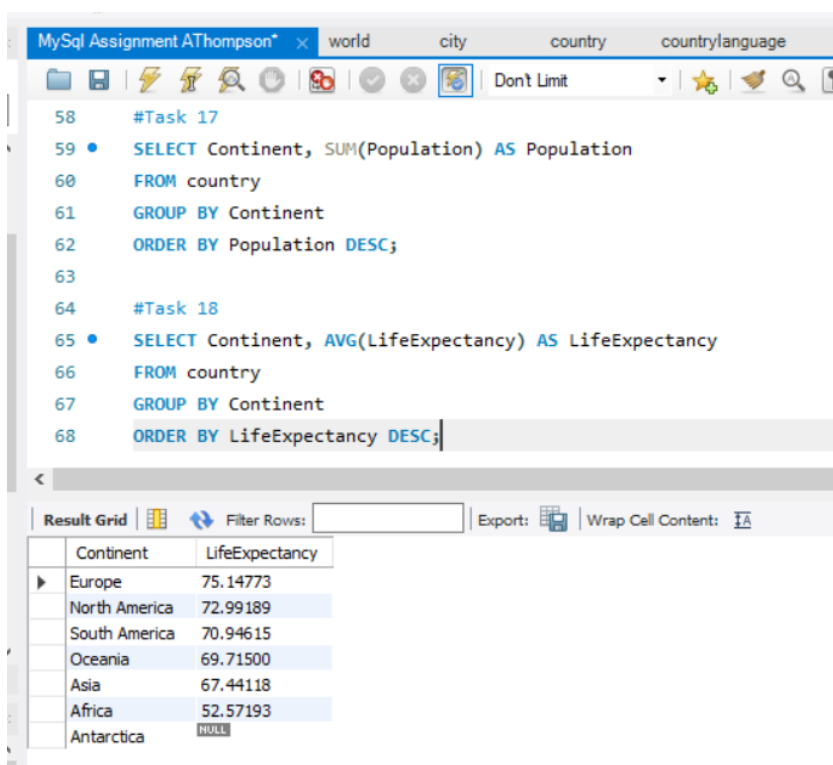
The screenshot shows the MySQL Workbench interface with a SQL editor and a result grid. The SQL editor contains the following code:

```
52 ON country.Code = city.CountryCode
53 WHERE country.Continent = "Europe";
54
55 #Task 16
56 • SELECT Name, Population FROM city ORDER BY Population DESC LIMIT 1;
57
58 #Task 17
59 • SELECT Continent, SUM(Population) AS Population
60 FROM country
61 GROUP BY Continent
62 ORDER BY Population DESC;
```

The result grid displays the following data:

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

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



The screenshot shows the MySQL Workbench interface with a SQL editor and a result grid. The SQL editor contains the following code:

```
58 #Task 17
59 • SELECT Continent, SUM(Population) AS Population
60 FROM country
61 GROUP BY Continent
62 ORDER BY Population DESC;
63
64 #Task 18
65 • SELECT Continent, AVG(LifeExpectancy) AS LifeExpectancy
66 FROM country
67 GROUP BY Continent
68 ORDER BY LifeExpectancy DESC;
```

The result grid displays the following data:

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

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

The screenshot shows the MySQL Workbench interface with a SQL query editor and a results grid. The query is as follows:

```

66 FROM country
67 GROUP BY Continent
68 ORDER BY LifeExpectancy DESC;
69
70 #Task 19
71 • SELECT country.Name, COUNT(city.CountryCode) AS NoofCities
72 FROM country
73 JOIN city
74 ON country.Code = city.CountryCode
75 GROUP BY country.Name
76 ORDER BY NoofCities DESC;

```

The results grid displays the following data:

Name	NoofCities
China	363
India	341
United States	274
Brazil	250
Japan	248
Russian Federation	189
Mexico	173
Philippines	136
Germany	93
Indonesia	85
United Kingdom	81

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

The screenshot shows the MySQL Workbench interface with a SQL query editor and a results grid. The query is as follows:

```

76 ORDER BY NoofCities DESC;
77
78 #Task 20
79 • SELECT country.Name, SUM(city.Population) AS PopulationByCity,
80 country.Population AS TotalPopulation,
81 (country.Population - SUM(city.Population)) AS Difference
82 FROM country
83 JOIN city
84 ON country.Code = city.CountryCode
85 GROUP BY country.Name, country.Population
86 ORDER BY PopulationByCity DESC;
87

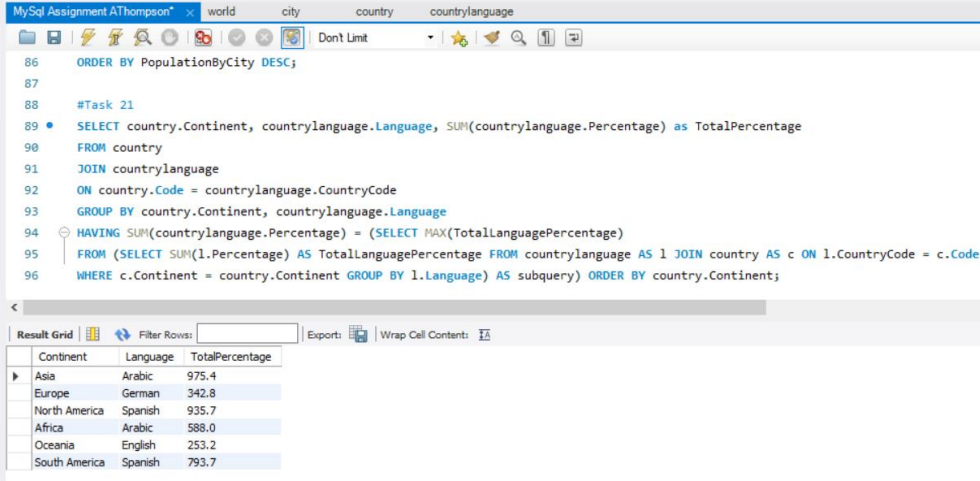
```

The results grid displays the following data:

Name	PopulationByCity	TotalPopulation	Difference
China	175953614	1277558000	1101604386
India	123298526	1013662000	890363474
Brazil	85876862	170115000	84238138
United States	78625774	278357000	199731226
Japan	77965107	126714000	48748893
Russian Federation	69150700	146934000	77783300
Mexico	59752521	98881000	39128479
South Korea	38999893	46844000	7844107
Indonesia	37485695	212107000	174621305
Pakistan	31546745	156483000	124936255
Philippines	30934791	75967000	45032209
Turkey	28327028	66591000	38263972

Note: I got cocky and included the difference between population and the population totals by city.

**Task 21:** Create a SQL statement to find the most spoken language on each continent.



The screenshot shows a MySQL IDE window with a SQL query for Task 21. The query is as follows:

```

86 ORDER BY PopulationByCity DESC;
87
88 #Task 21
89 SELECT country.Continent, countrylanguage.Language, SUM(countrylanguage.Percentage) as TotalPercentage
90 FROM country
91 JOIN countrylanguage
92 ON country.Code = countrylanguage.CountryCode
93 GROUP BY country.Continent, countrylanguage.Language
94 HAVING SUM(countrylanguage.Percentage) = (SELECT MAX(TotalLanguagePercentage)
95 FROM (SELECT SUM(l.Percentage) AS TotalLanguagePercentage FROM countrylanguage AS l JOIN country AS c ON l.CountryCode = c.Code
96 WHERE c.Continent = country.Continent GROUP BY l.Language) AS subquery) ORDER BY country.Continent;

```

Below the query, the 'Result Grid' shows the following data:

Continent	Language	TotalPercentage
Asia	Arabic	975.4
Europe	German	342.8
North America	Spanish	935.7
Africa	Arabic	588.0
Oceania	English	253.2
South America	Spanish	793.7

Note: Even with Chat GPT's help this still looks wrong, but it was the best I could do.

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

MySql Assignment AThompson\* x world city country countrylang

96 WHERE c.Continent = country.Continent GROUP BY 1.Language)

97

98 #Task 22

99 • SELECT country.name, countrylanguage.Language

100 FROM country

101 JOIN countrylanguage

102 ON country.Code = countrylanguage.CountryCode

103 WHERE countrylanguage.IsOfficial = "T"

104 AND countrylanguage.Language = "English"

105 OR countrylanguage.Language = "Spanish"

106 OR countrylanguage.Language = "French";

Result Grid Filter Rows: Export: Wrap Cell Content:

	name	Language
▶	Aruba	Spanish
	Anguilla	English
	Andorra	French
	Andorra	Spanish
	Argentina	Spanish
	American Samoa	English
	Antigua and Barbuda	English
	Australia	English
	Burundi	French
	Belgium	French
	Belgium	French

Result 65 x

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

MySQL Assignment AThompson\* x world city country con

102 ON country.Code = countrylanguage.CountryCode  
 103 WHERE countrylanguage.IsOfficial = "T"  
 104 AND countrylanguage.Language = "English"  
 105 OR countrylanguage.Language = "Spanish"  
 106 OR countrylanguage.Language = "French";  
 107  
 108 #TASK 23  
 109 • SELECT Continent, SUM(Population) AS Population  
 110 FROM country  
 111 GROUP BY Continent  
 112 ORDER BY Population DESC;

Result Grid | Filter Rows: | Export: | Wrap Cell Cor

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

**Task 24:** Write a query to list countries that have more than three official languages.

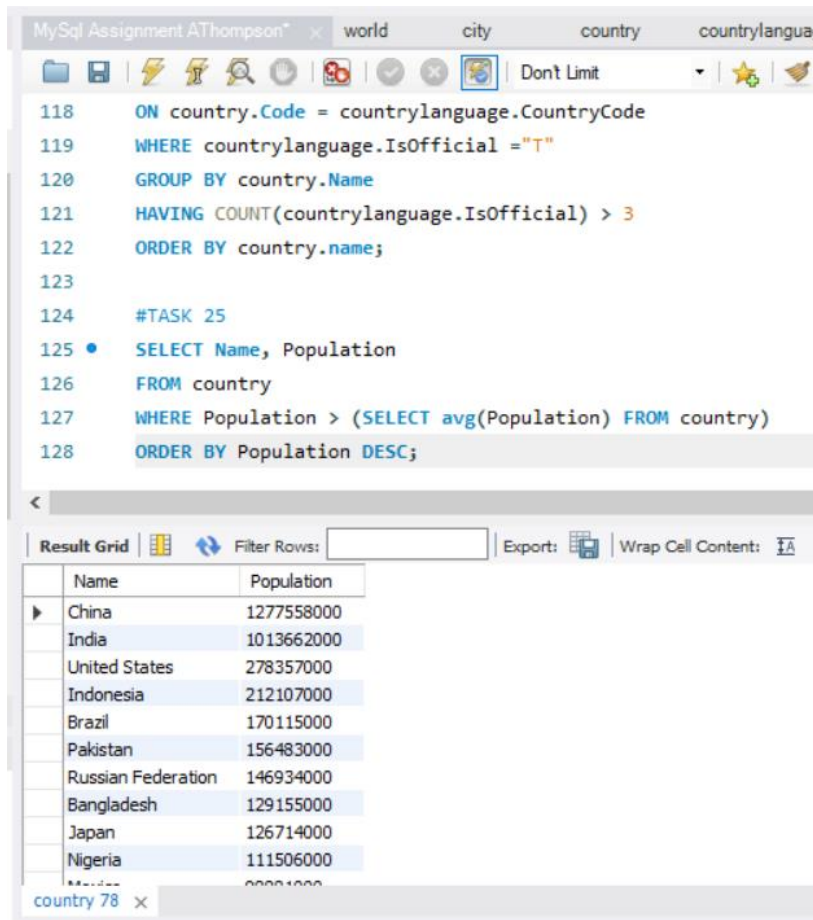
MySQL Assignment AThompson\* x world city country countrylanguage

112 ORDER BY Population DESC;  
 113  
 114 #Task 24  
 115 • SELECT country.Name AS Country, COUNT(countrylanguage.Language) AS OfficialLanguageCount  
 116 FROM country  
 117 JOIN countrylanguage  
 118 ON country.Code = countrylanguage.CountryCode  
 119 WHERE countrylanguage.IsOfficial = "T"  
 120 GROUP BY country.Name  
 121 HAVING COUNT(countrylanguage.IsOfficial) > 3  
 122 ORDER BY country.name;

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

Country	OfficialLanguageCount
South Africa	4
Switzerland	4

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



The screenshot shows a MySQL IDE window with a SQL query editor and a results grid. The query is as follows:

```

118 ON country.Code = countrylanguage.CountryCode
119 WHERE countrylanguage.IsOfficial = "T"
120 GROUP BY country.Name
121 HAVING COUNT(countrylanguage.IsOfficial) > 3
122 ORDER BY country.name;
123
124 #TASK 25
125 • SELECT Name, Population
126 FROM country
127 WHERE Population > (SELECT avg(Population) FROM country)
128 ORDER BY Population DESC;

```

The results grid displays the following data:

Name	Population
China	1277558000
India	1013662000
United States	278357000
Indonesia	212107000
Brazil	170115000
Pakistan	156483000
Russian Federation	146934000
Bangladesh	129155000
Japan	126714000
Nigeria	111506000

### Task 26:

Primary Key in country Table – 'Code'

Primary Key in city Table – 'ID'

Primary Key in countrylanguage Table – 'CountryCode'

Foreign Key in city Table – 'CountryCode'

Foreign Key in countrylanguage Table – 'CountryCode'