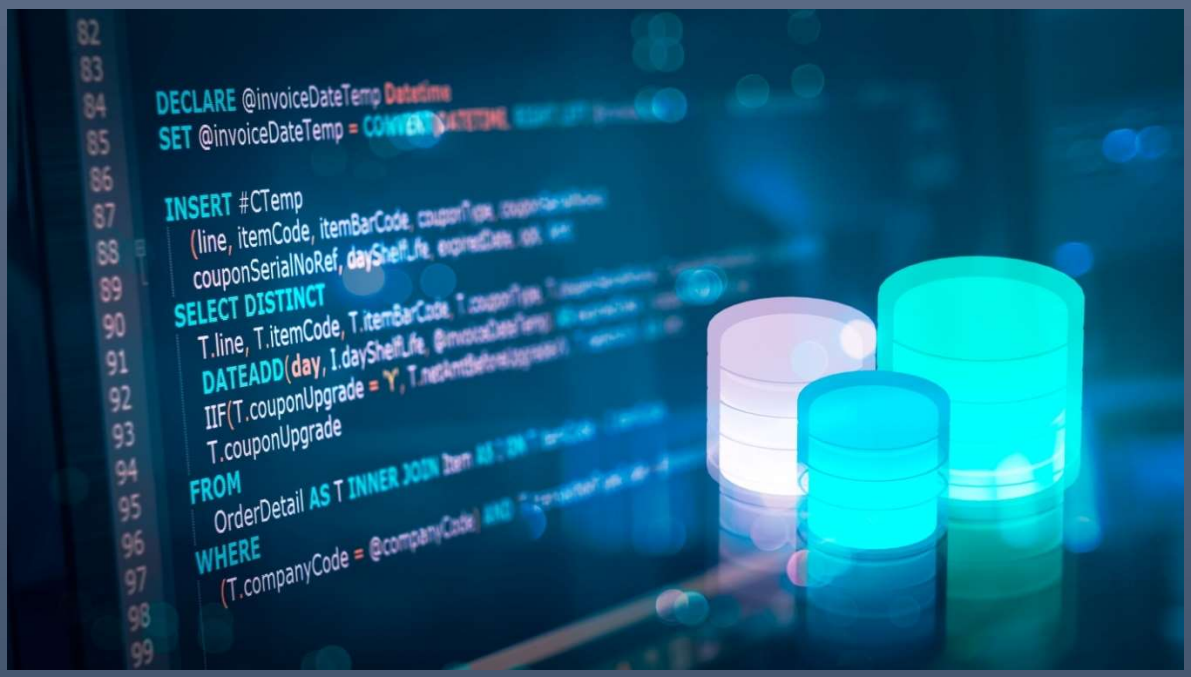


MySQL Assignment

AJAY KACHHALA

Contents

Introduction

MySQL is a popular open-source relational database management system (RDBMS) that is widely used for managing and manipulating databases. MySQL is widely used for both small- and large-scale applications due to its reputation for speed, dependability, and user-friendliness.

Task 1

In relational databases, there are several types of relationships that define how data in different tables are related to each other. The most common types of relationships are:

- One-to-One (1:1) Relationship: Each record in one table is associated with exactly one record in another table, and vice versa.
  Example: A database for storing employee information where each employee has one and only one unique employee ID, and each ID is associated with a single employee record.
- One-to-Many (1: N) Relationship Each record in one table can be associated with multiple records in another table, but each record in the second table is associated with only one record in the first table.
  Example: A database for a library where one author can have multiple books. Each author is related to multiple book records, but each book is associated with only one author.
- Many-to-One (N:1) Relationship: Multiple records in one table can be associated with a single record in another table.
  Example: In an order processing system, multiple orders can be placed by different customers. Each order is associated with one customer, but a customer can have multiple orders.
- Many-to-Many (N: N) Relationship: Multiple records in one table can be associated with multiple records in another table, typically using a junction or linking table.
  Example: A database for a university where students can enroll in multiple courses, and each course can have multiple students. This is typically implemented using a junction table to connect students and courses.
- Self-Referencing Relationship: Records in the same table are related to each other, often representing hierarchical or parent-child relationships.
  Example: In an organizational database, you may have an employee table where each employee can have a manager who is also an employee. In this case, the manager and employee relationships are both based on the same "employee" table.
- Recursive Relationship: A specific type of self-referencing relationship where records in a table are related hierarchically to other records in the same table.
  Example: In a hierarchical data structure like an organizational chart, where each employee has a supervisor who is also an employee, creating a recursive relationship.
- Weak Entity Relationship: An entity that cannot be uniquely identified by its attributes alone and depends on another entity for identification.

Example: A weak entity is an entity that cannot be uniquely identified by its attributes alone, and it depends on another entity (the owner entity) for identification. For instance, in a database for tracking parts in a manufacturing process, a part might be a weak entity if it relies on the job number for its uniqueness.

- Unary Relationship: Records in a single entity are related to other records within the same entity.
  Example: In a social networking database, a unary relationship could represent a "Friendship" where each user can be friends with other users. This is essentially a one-to-many relationship within a single entity (User).
- Inclusive Relationship: A many-to-many relationship with additional attributes describing the relationship.
  Example: A database for a store inventory system, where a product can belong to multiple categories. This is a many-to-many relationship but with additional attributes that describe the relationship (e.g., the date the product was added to a category).

These are the most common types of relationships in relational databases, and they are used to define how data in different tables or entities are interconnected. The choice of relationship type depends on the specific requirements and data modelling needs of the database system.

Task 2:

Normalization is a process in database design that aims to reduce data redundancy and improve data integrity by organizing data into separate related tables. It is essential to database development for several reasons:

1. Data Integrity: Normalization helps maintain data integrity by reducing anomalies such as insertion, update, and deletion anomalies. This ensures that data is accurate and consistent.

2. Reducing Data Redundancy: Normalization minimizes the duplication of data, which not only conserves storage space but also makes it easier to update data since changes only need to be made in one place.

3. Improved Data Structure: Normalized databases have a clear and well-structured schema, making it easier to understand and work with the database. This leads to better query performance and maintainability.

4. Easier Querying: Normalized databases are often more straightforward to query since data is organized logically into related tables. This simplifies the process of retrieving information.

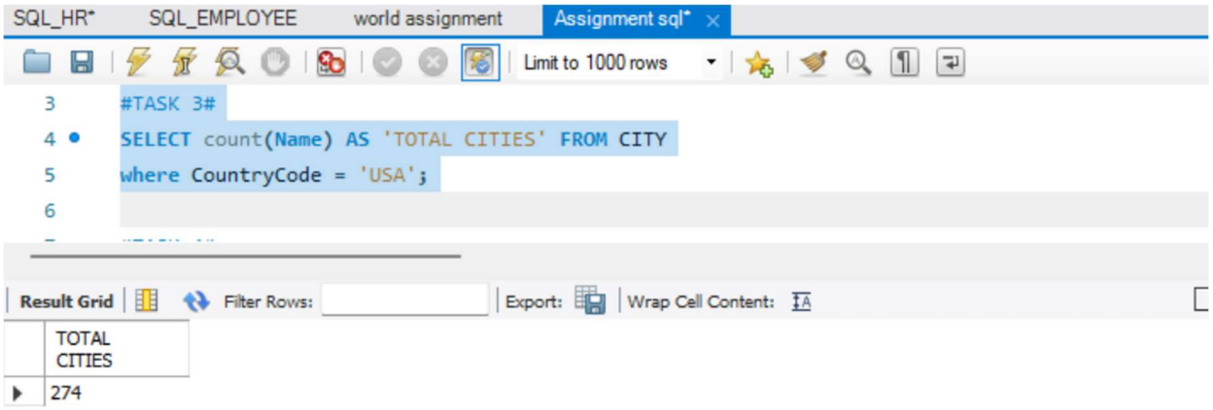
5. Scalability: Normalized databases are more adaptable to changing business requirements. New tables can be added to accommodate new types of data without affecting existing data.

6. Consistency: Data consistency is enhanced in normalized databases because related information is stored in a single place, reducing the chances of conflicting or contradictory data.

Normalization typically involves dividing a database into multiple related tables, each with a specific purpose, and establishing relationships between these tables using keys (e.g., primary keys and foreign keys). The process usually follows specific normalization forms, such as First Normal Form (1NF), Second Normal Form (2NF), and so on, each of which defines certain criteria for organizing data.

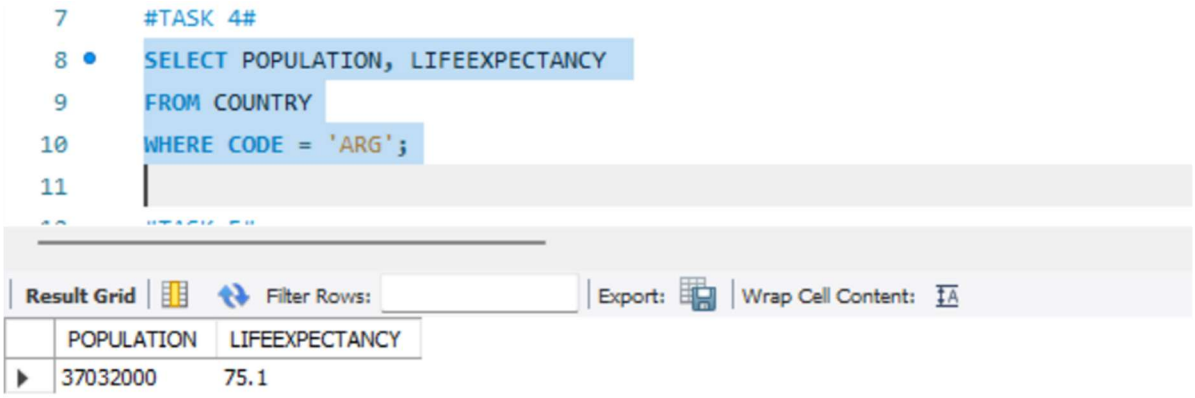
In summary, normalization is crucial to database development because it ensures data accuracy, reduces redundancy, improves data structure, enhances query performance, and makes the database more adaptable to changing requirements. It is a fundamental step in creating efficient and maintainable database systems.

TASK-3



Used the count to get the total number of cities in USA as shown above.

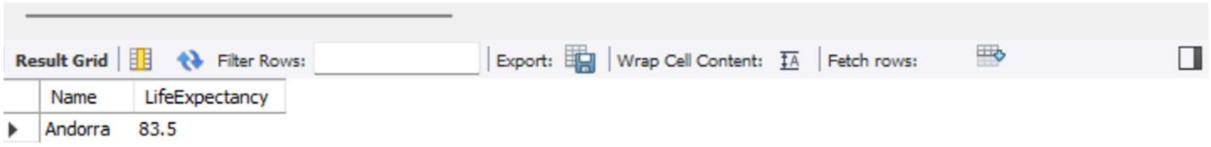
TASK-4



The population and life expectancy in Argentina are 37032000 and 75.1 respectively.

TASK-5

```
12    #TASK 5#
13 •  SELECT Name,LifeExpectancy
14    FROM country
15    ORDER BY LifeExpectancy DESC
16    LIMIT 1;
```

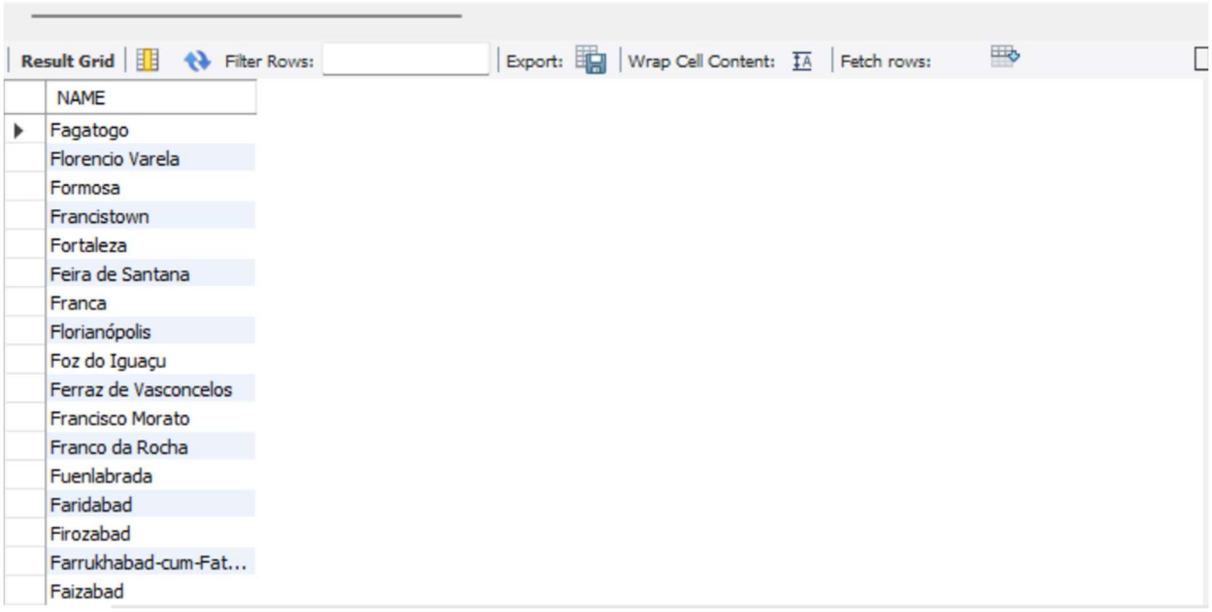


Name	LifeExpectancy
Andorra	83.5

As shown above, using *order by, limit*, Andorra has highest life expectancy of 83.5.

TASK-6

```
18    #TASK 6#
19 •  SELECT NAME
20    FROM CITY
21    WHERE NAME LIKE 'F%'
22    LIMIT 25;
```



NAME
Fagatogo
Florencio Varela
Formosa
Francistown
Fortaleza
Feira de Santana
Franca
Florianópolis
Foz do Iguaçu
Ferraz de Vasconcelos
Francisco Morato
Franco da Rocha
Fuenlabrada
Faridabad
Firozabad
Farrukhabad-cum-Fat...
Faizabad

```
18      #TASK 6#
19 •    SELECT NAME
20      FROM CITY
21      WHERE NAME LIKE 'F%'
22      LIMIT 25;
```

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

NAME
Foz do Iguaçu
Ferraz de Vasconcelos
Francisco Morato
Franco da Rocha
Fuenlabrada
Faridabad
Firozabad
Farrukhabad-cum-Fat...
Faizabad
Fatehpur
Firenze
Foggia
Ferrara
Forlì
Fukuoka
Funabashi
Fukuyama

Here are 25 cities from around the world, that start with letter F.

TASK-7

```
24      #TASK 7#
25 •    SELECT ID, Name, Population
26      FROM city
27      limit 10;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: | Fetch row

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

Here is the result that displays only first 10 columns id, name and population from city table.

TASK-8

```
29      #TASK 8#
30 •    SELECT NAME,Population
31      FROM city
32      WHERE Population > 2000000;
```

Result Grid | Filter Rows: | Export

NAME	Population
Alger	2168000
Luanda	2022000
Buenos Aires	2982146
Sydney	3276207
Melbourne	2865329
Dhaka	3612850
São Paulo	9968485
Rio de Janeiro	5598953
Salvador	2302832
Belo Horizonte	2139125
Fortaleza	2097757
London	7285000
Santiago de ...	4703954

As shown above, these are the cities from the city table, whose population is larger than 2000000.

Note: For the above query, there are a greater number of cities and cannot be captured in one screenshot.

TASK-9

```
34      #TASK 9#
35 •    SELECT Name
36      FROM city
37      where name like 'Be%'
```

Result Grid | Filter Rows:

Name
Bengasi
Beni-Mellal
Beau Bassin-...
Benito Juárez
Bender (Tighi...
Beira
Benin City
Bergen
Besançon
Berlin
Bergisch Glad...
Bern
Berdjansk
Berdytšiv
Belgorod
Berezniki
Beaumont

Here are all the cities with their names beginning with ‘Be’ prefix.

Note: For the above query, there are a greater number of cities and cannot be captured in one screenshot.

TASK-10

```
39      #TASK 10#
40 •    SELECT Name,Population
41      FROM city
42      WHERE Population between 500000 AND 1000000;
```

Result Grid | Filter Rows: | Export: | Wrap

Name	Population
Amsterdam	731200
Rotterdam	593321
Oran	609823
Dubai	669181
Rosario	907718
Lomas de Zamora	622013
Quilmes	559249
Almirante Brown	538918
La Plata	521936
Mar del Plata	512880
Adelaide	978100
Khulna	663340
Cotonou	536827
Santa Cruz de la...	935361
La Paz	758141

As shown above, these are the cities with population between 500000-1000000.

TASK-11

```
44      #TASK 11#
45 •    SELECT Name
46      FROM city
47      ORDER BY Population ASC
48      LIMIT 1;
49
```

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

Name
Adamstown

Adamstown has the lowest population.

TASK-12

```
50    #TASK 12#
51 •  SELECT c.name as country,c.population as Population,
52    l.language as Language,l.percentage as Percentage
53    FROM country AS c
54    JOIN countrylanguage as l
55    ON c.code= l.countrycode
56    WHERE c.name='SWITZERLAND';
57
```

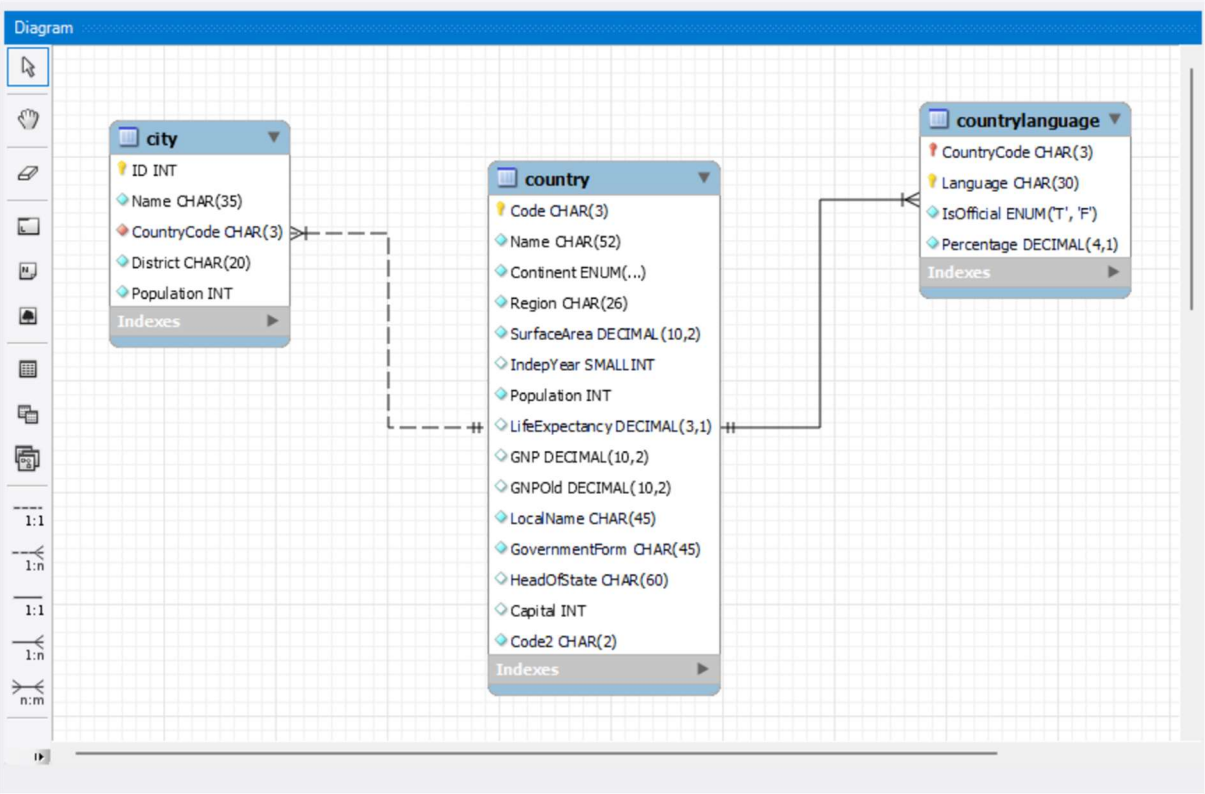
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

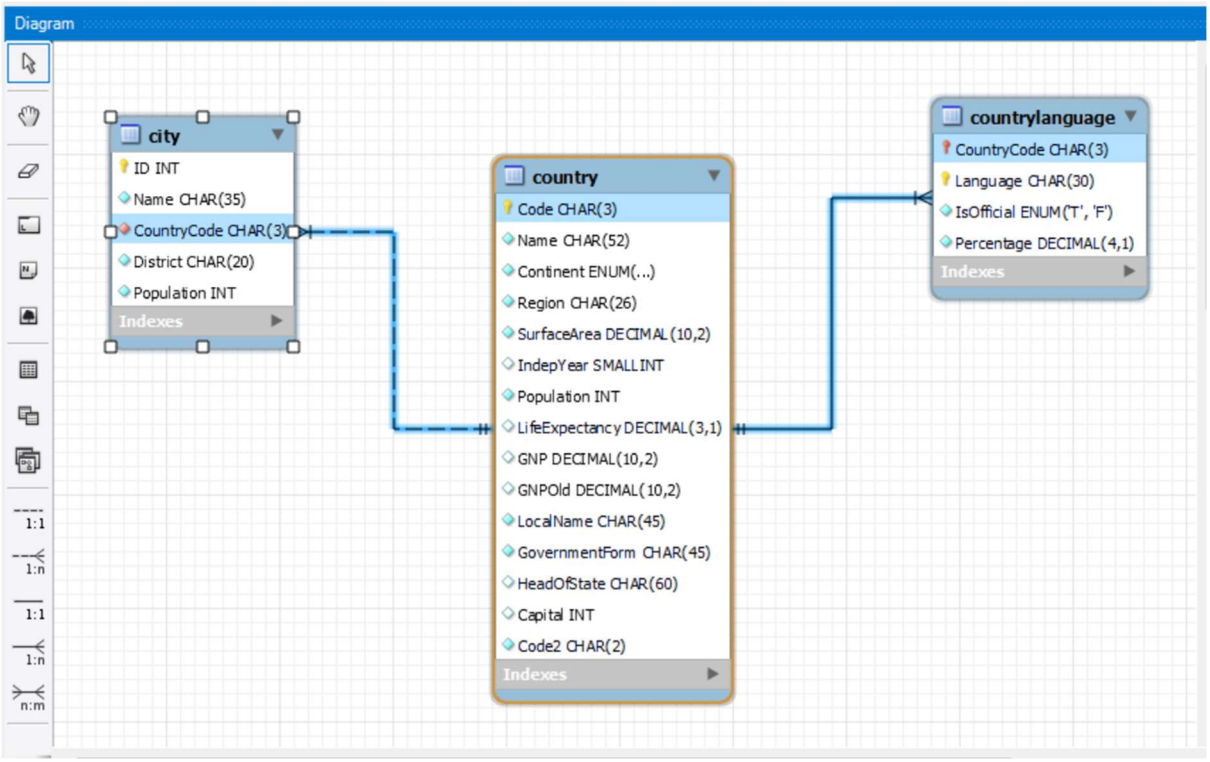
country	Population	Language	Percentage
Switzerland	7160400	French	19.2
Switzerland	7160400	German	63.6
Switzerland	7160400	Italian	7.7
Switzerland	7160400	Romansh	0.6

The output shows the population and languages spoken in Switzerland.

TASK-13

Creating an EER Diagram





TASK-14

- Identify the primary key in country table
  - code is the primary key in country table

- Identify the primary key in city table
  - Id is the primary key in city table

- Identify the primary key in country language table
  -Language is the primary key in country language table

- Identify the foreign key in city table
  -Country code is the foreign key in city table

- Identify the foreign key in country language table
  -Country code is the foreign key and also, it is primary key in country language table

TASK-15

Capital per country with population in descending order.

```
54      #TASK 15#  Show capital per country with population desc order
55 •    SELECT Name,Capital,Population
56      FROM country
57      ORDER BY  Capital desc;
58
```

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

Name	Capital	Population
Palestine	4074	3101000
Zimbabwe	4068	11669000
Virgin Islands, U.S.	4067	93000
United States	3813	278357000
Estonia	3791	1439200
Vietnam	3770	79832000
Russian Federation	3580	146934000
Venezuela	3539	24170000
Holy See (Vatican City State)	3538	1000
Vanuatu	3537	190000
Wallis and Futuna	3536	15000
Belarus	3520	10236000
Uzbekistan	3503	24318000
New Zealand	3499	3862000

TASK-16

Names starting with N and end with e.

```
59      #TASK 16# Show name start with N and end with e.
60 •    SELECT Name
61      FROM country
62      WHERE Name like 'N%e';
```

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

Name
Niue

TASK-17

Name, population and life expectancy with more than 70 and top 15.

```
64    #TASK 17# show name,population and lifeexpectancy with more than 70 and top 15.
65    SELECT Name,Population,LifeExpectancy
66    FROM country
67    WHERE LifeExpectancy >70
68    LIMIT 15;
```

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

Name	Population	LifeExpectancy
Aruba	103000	78.4
Anguilla	8000	76.1
Albania	3401200	71.6
Andorra	78000	83.5
Netherlands Antilles	217000	74.7
United Arab Emirates	2441000	74.1
Argentina	37032000	75.1
American Samoa	68000	75.1
Antigua and Barbuda	68000	70.5
Australia	18886000	79.8
Austria	8091800	77.7
Belgium	10239000	77.8
Bulgaria	8190900	70.9
Bahrain	617000	73.0
Bahamas	307000	71.1

Reflection

MySQL is completely new for me and I found it interesting to learn. Some of the queries I found easy to learn and understand while queries like in task 12, I found difficult but I managed to solve it. I need more practise in MySQL to get a better grasp on it. I found that once I knew the relationships between the tables, it made it easier for me to understand the data. By finishing this assignment, I have learnt more advanced way to use database systems like in Task 13, which included creating and using EER Diagram where I gained good understanding about the primary and foreign keys.