Rohit Gokani

Mysql and assignment

Table of Contents

[Task 1: 2](#_Toc174957077)

[Types of Relationships in Relational Databases 2](#_Toc174957078)

[One-to-One Relationship: 2](#_Toc174957079)

[One-to-Many Relationship 2](#_Toc174957080)

[Many-to-Many Relationship 2](#_Toc174957081)

[Task 2: 4](#_Toc174957082)

[What is Normalisation and why is it important to database development? 4](#_Toc174957083)

[Key advantages of normalisation 4](#_Toc174957084)

[Normal Forms 4](#_Toc174957085)

[Database 6](#_Toc174957086)

[Looking at the Data 8](#_Toc174957087)

[JOINS in SQL 9](#_Toc174957088)

[Creating an EER Diagram 16](#_Toc174957089)

[Name of my answers. 18](#_Toc174957090)

# Task 1:

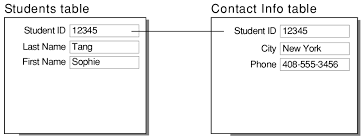
List the different types of relationships in relational databases and provide examples.

## Types of Relationships in Relational Databases

Relational databases primarily utilise three types of relationships to structure data.

### One-to-One Relationship:

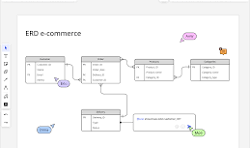
* Definition: A single record in one table corresponds to exactly one record in another table.
* Example: A person may have only one passport.
* Implementation: Often achieved through primary and foreign key constraints with unique indexes.



*Figure .1.*

*Source: [https://www.google.com/imgres?q=images%20of%20one%20to%20one%20relationship%20in%20databases&imgurl=https%3A%2F%2Fhelp.claris.com%2Farchive%2Fhelp%2F18%2Ffmp%2Fen%2FFMP\_Help%2Fimages%2Frelational.07.03.2.png&imgrefurl=https%3A%2F%2Fhelp.claris.com%2Farchive%2Fhelp%2F18%2Ffmp%2Fen%2FFMP\_Help%2Fone-to-one-relationships.html&docid=JcPIgYb5huPsWM&tbnid=jL5Cvr7LzBu90M&vet=12ahUKEwjitpLGvvmHAxUfWUEAHV6yHtIQM3oECBYQAA..i&w=821&h=313&hcb=2&ved=2ahUKEwjitpLGvvmHAxUfWUEAHV6yHtIQM3oECBYQAA] accessed [16/8/2024]*

### One-to-Many Relationship

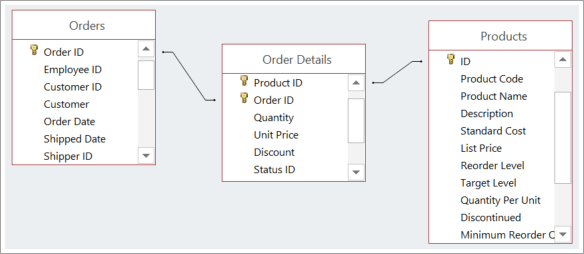
* Definition: A single record in one table can be associated with multiple records in another table, but a record in the second table can only be associated with one record in the first table.
* Example: A customer can have multiple orders, but an order belongs to only one customer.
* Implementation: Typically involves a primary key in one table and a foreign key in the other.

*Figure .2.*

*Source: [https://miro.com/diagramming/er-diagram-one-to-many-relationship/]accessed [16/8/2024]*

### Many-to-Many Relationship

* Definition: A record in one table can be associated with multiple records in another table, and vice versa.
* Example: A customer can buy many products, and a product can be bought by many customers.
* Implementation: Requires a junction table to represent the relationship.



*Figure .3.*

*Source:[https://support.microsoft.com/en-us/office/video-create-many-to-many-relationships-e65bcc53-8e1c-444a-b4fb-1c0b8c1f5653] accessed [16/8/2024]*

# **Task 2:**

## What is Normalisation and why is it important to database development?

Database normalisation is the process of organising the data in a relational database; it's one of the most fundamental requirements. A lot of the work in normalisation focuses on reducing data redundancy and the dependency of data elements upon one another, which leads to a more effective, reliable, and maintainable database structure.

Key advantages of normalisation:

* **Reduction of redundancy**: Normalisation saves space by not repeating the same data in multiple places. Anomalies arising from information duplication are also prevented.
* **Improved Data Integrity**: When the data is stored in a normalised structure, the updates and changes can be only made in one central place and thereby ensure that there is consistency across the data in the database.
* **Improved Query Performance:** Due to well-defined relations in normalised tables, data retrieval by optimized queries becomes faster and more efficient. Complex joins and data searches become less demanding on the database system.
* **Simplified Database Maintenance**: It is easier to maintain and change a well-normalised database as data requirements change. It is relatively easy to maintain, add, remove, or change data elements since the dependencies between data elements at a point start to get minimized.

### Normal Forms

The objective is to normalise the database to a standard form of rules.

The most common one includes the following:

**First Normal Form (1NF**): Eliminates repeating groups within a table and identifies a unique identifier for every row.

**Second Normal Form (2NF)**: Full dependency of attribute on the primary key, rather than just a part of it.

**Third Normal Form (3NF):** Elimination of interdependency between non-key attributes; an attribute is dependent on the primary key.

Example:

Imagine an unnormallised table having information about customers, such as names and addresses, and the items ordered, such as the product ordered and the price, along with the identification of the orders. The violation of the principle of normalisation in the table disappears if the design uses separate tables as follows:

Customers: (CustomerID, CustomerName, CustomerAddress)

Products: (ProductID, ProductName, ProductPrice)

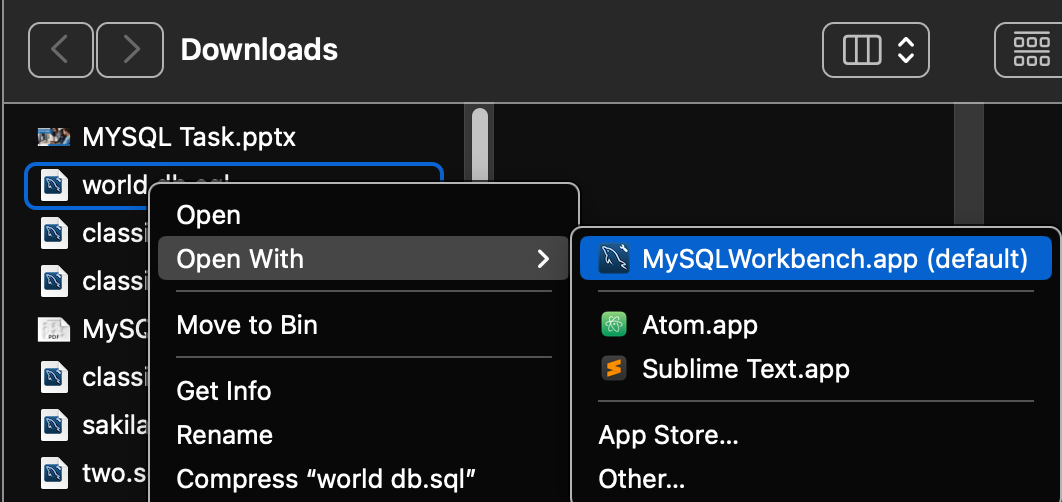
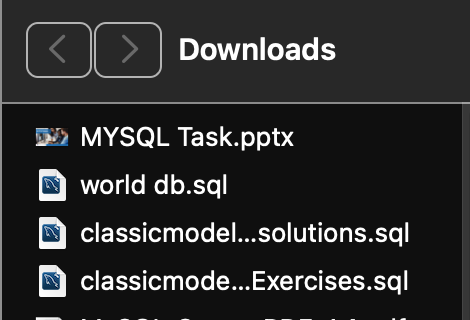
Orders: (OrderID, CustomerID, OrderDate)

OrderDetails: (OrderID, ProductID, Quantity)

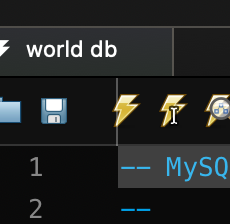
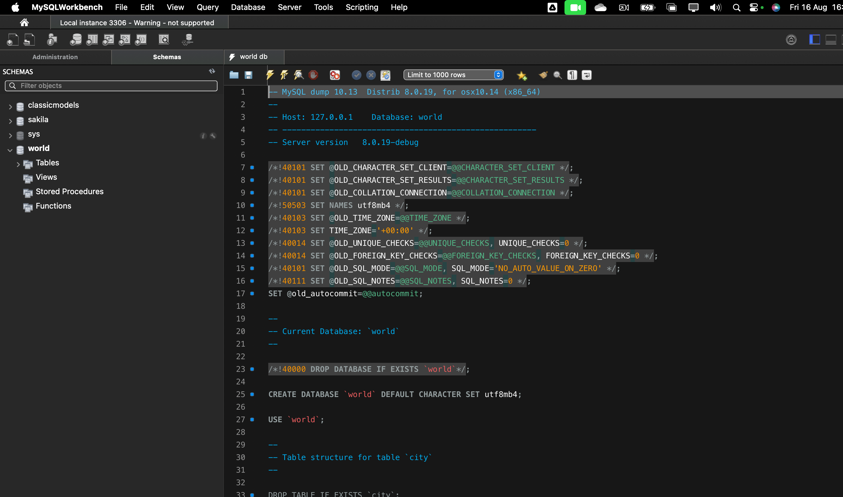
By linking tables, data redundancy is reduced, data consistency is enhanced, and query performance is improved.

# Database

The “world db .sql” file containing the dataset was downloaded from the MS Teams location.

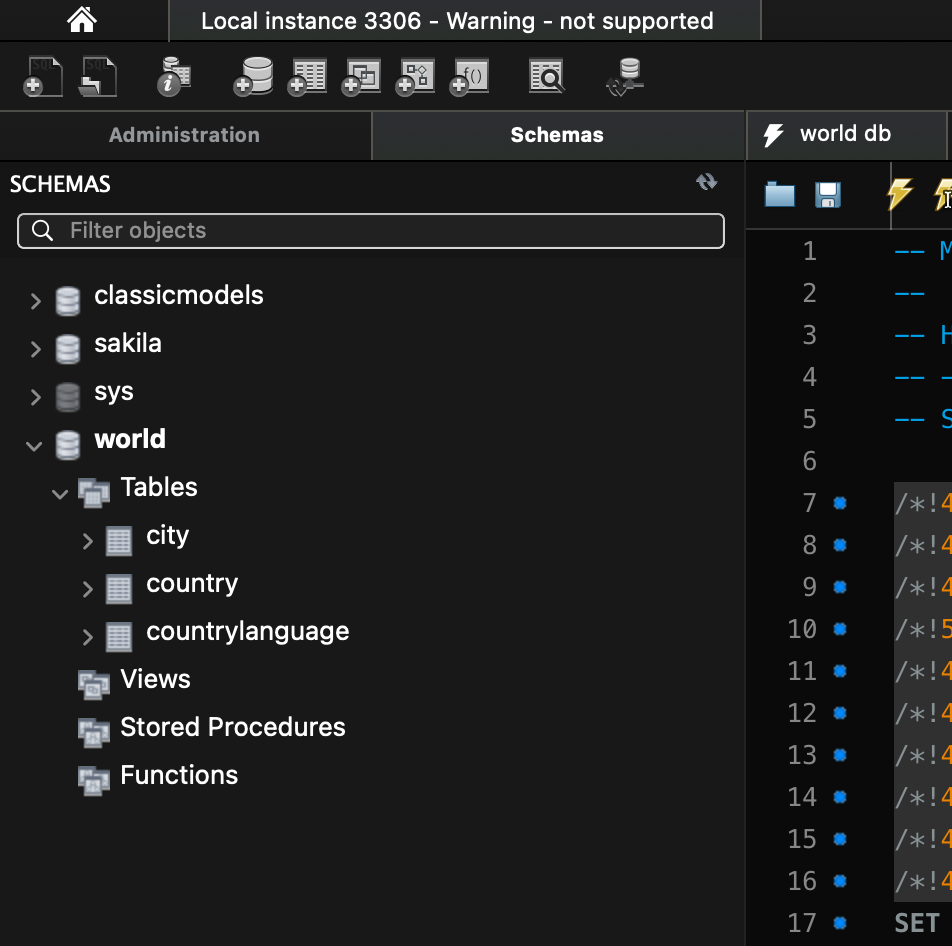


Opened using the” MySQLWorkbench .app”



The file was executed using the “execute” icon.

Now the dataset has been loaded onto the MySQL Workbench and the data is ready to use.



## Looking at the Data

The data has a set of three tables:

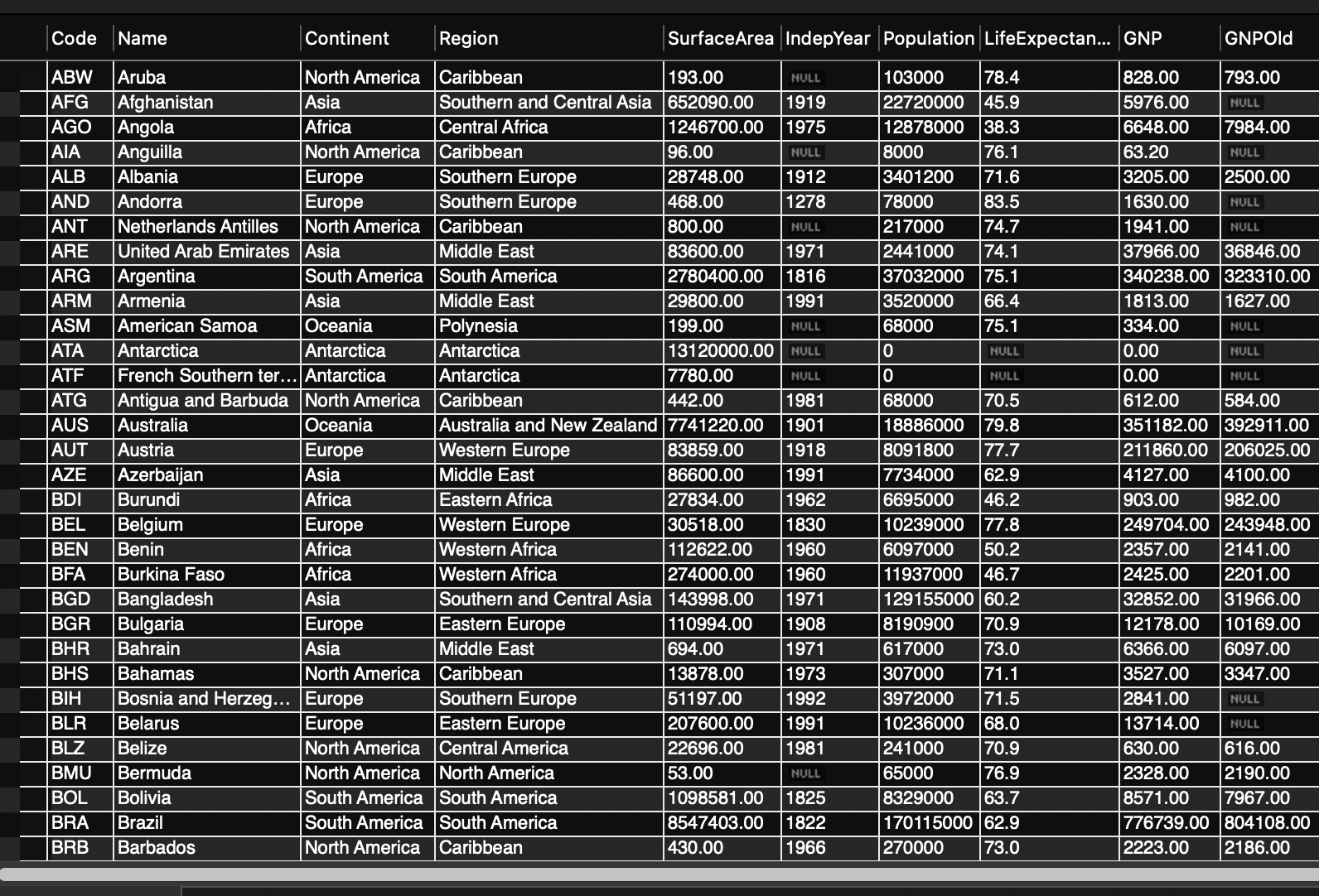
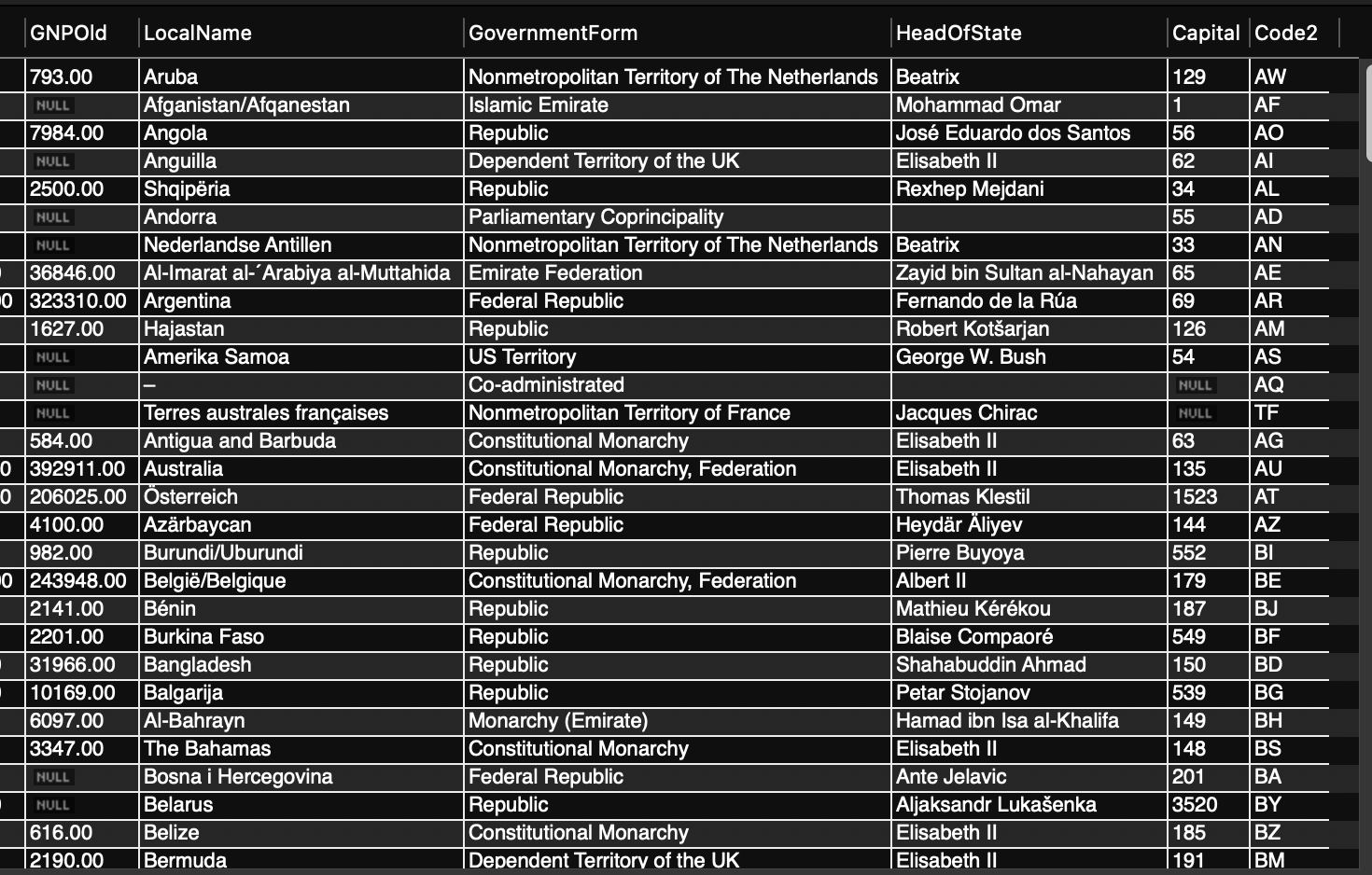
* **city:** Information about some of the cities in those countries.

A screenshot of a black and white table

Description automatically generated

The “city Table” has 5 columns: ID, NAME, CountryCode, District and Population.

* **country:** Information about countries of the world.



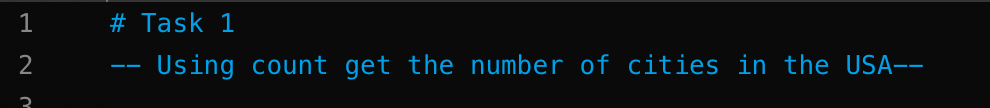
The 15 columns include: Code, Name, Continent, Region, SurfaceArea, indepYear, Population, LifeExpectency, GNP,GNPOld, LocalName, GovermentForm, HeadOfState, Capital and Code2.

* **country/language**: Languages spoken in each country.

A black and white table with white text

Description automatically generated

The “country/language” has 4 columns: CountryCode, Language, IsOfficial and Percentage.

Task 1 within the database section requires the result of the number of cities in the USA within the dataset.

A black background with white text

Description automatically generated

Firstly, the workbench must be connected to the “world” datasets

To return the count of the cities in the USA.

* Two tables were joined:

1. The “city table to the “country” table using an “INNER JOIN” on the “country\_id” column
2. A computer screen shot of a number of cities

   Description automatically generatedAll fields were selected from the tables

### JOINS in SQL

SQL Joins can be regarded as instrumental in relational databases since they combine data from two or more tables with related columns.

Types of Joins:

INNER JOIN: Returns rows having the matching values in both tables.

LEFT JOIN: This will return all the rows from the left table and the matched rows from the right.

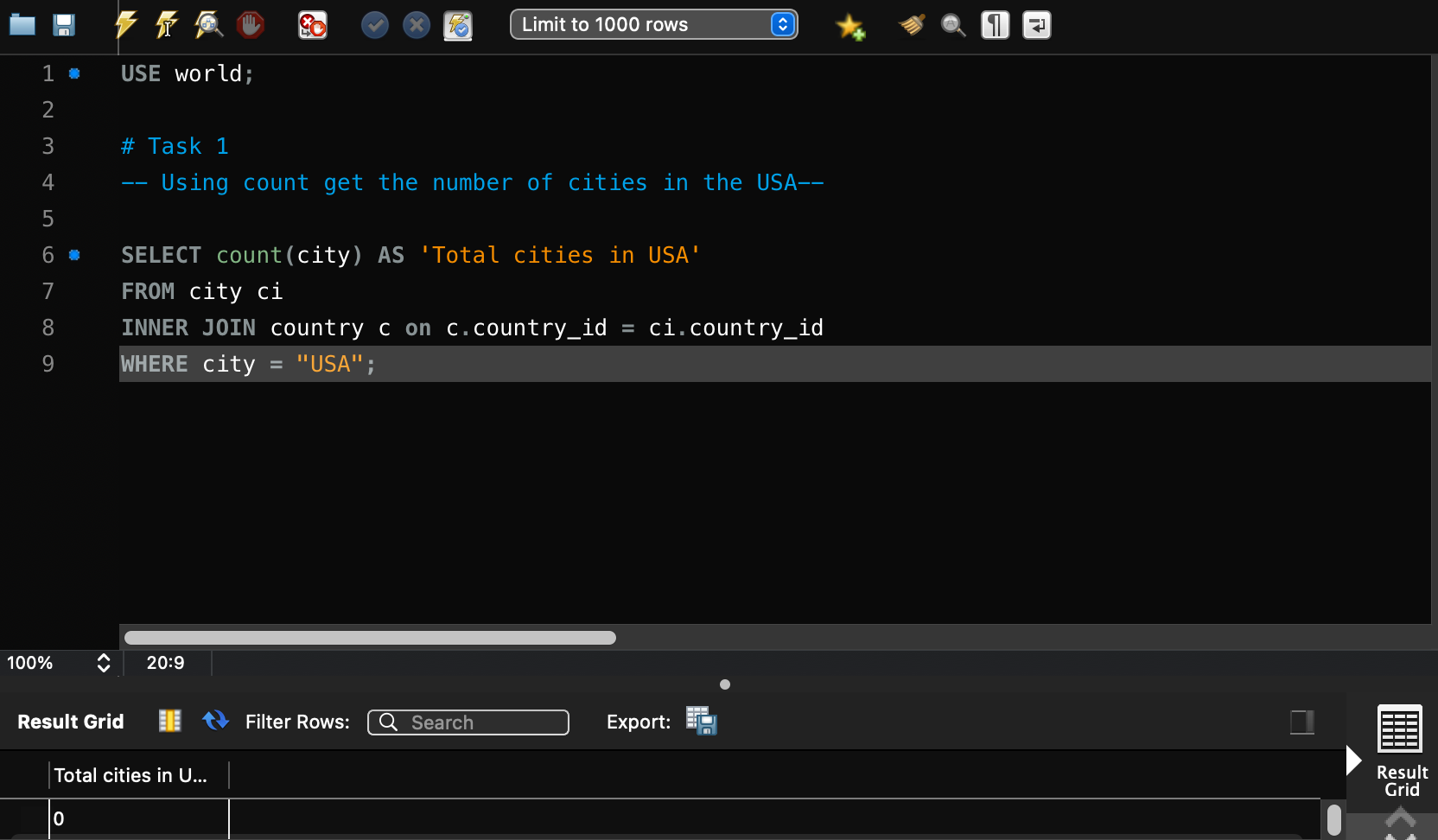
RIGHT JOIN: Returns all rows from the right table and matched rows from the left one.

FULL OUTER JOIN: This would return all rows from either the left or the right table where there is a match. SELF JOIN: Joining a table to itself on a related column.

A screenshot of a computer

Description automatically generatedThe result:

Now that the relevant data has been acquired the search for the appropriate columns and the right clause and conditions may be applied, for the retrieval of data indicating the number of cities in the USA.



Using the “count(city)” function in SQL related to the number of rows in a table or the number of rows that satisfied this specified condition.

The “AS” statement created a column and applied the name: ‘Total cities in USA’.

The “ci” and the “c” created aliases for “city table” and “country table” saving time on having to constantly retype these fields.

The WHERE clause was used to filter records, retrieving where the city was in “USA” therefore retrieving only those fields specified where conditions correlated within this query.

A screenshot of a computer

Description automatically generatedThe result:

A renamed column and count of cities in the USA: 600.

A screenshot of a computer

Description automatically generated To find out what the population and life expectancy for people in Argentina (ARG) is, the search was based on the “Population, LifeExpentancy, Code, Name: columns from “world. country” table with a Where (Code = (operator) ‘ARG’) was utilized.

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

A screenshot of a computer

Description automatically generated

**SELECT Name, LifeExpectancy**: Select the Name and the Life Expectancy columns from the world.country table.

**ORDER BY LifeExpectancy DESC**: It would typically order the results by the life expectancy in descending order (highest first).

**LIMIT 1**: Limits the result to the first row, which should be the country with the highest life expectancy .

A screenshot of a computer

Description automatically generatedThe result

A black and grey screen with yellow text

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

**WHERE:** Name **LIKE** 'F%': Filters cities starting with the letter 'F'

**LIKE:** function seeks patterns, % at the end of the “F” signifies the code to search for patterns beginning with ‘F”.

**ORDER BY\_Name:** Sorts the results alphabetically by city name.

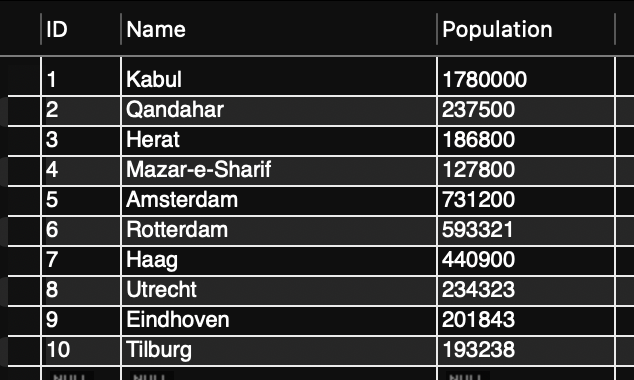
**LIMIT 25**: Limits the result to the first 25 cities.

A screenshot of a computer

Description automatically generatedThe Result

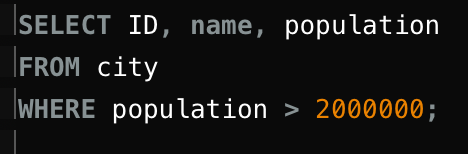
A close-up of a black and white background

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



A screenshot of a black screen

Description automatically generatedCreate an SQL statement to find only those cities from the city table whose population is larger than 2000000.



The (>) greater than operator was used on this query.

A black and white list with white text

Description automatically generatedA black background with white text

Description automatically generated Create an SQL statement to find all city names from the city table whose name begins with the “Be” prefix.

The (%) percentage operator at the end of ‘Be’ was used on this query.

A black and white list with white text

Description automatically generatedCreate an SQL statement to find only those cities from the city table whose population is between 500000-1000000.



The BETWEEN operator in SQL is used to select values within a specified range.

A black rectangular object with white lines

Description automatically generatedA screenshot of a computer

Description automatically generatedCreate an SQL statement to find the city with the lowest population in the city table.

This query utilized the “ASC” operator to retrieve the order by ascension.

A black screen with white text

Description automatically generatedCreate an SQL statement to show the population of Switzerland and all the languages spoken there.

**SELECT** c.Name**, GROUP\_CONCAT(**cl.Language **SEPARATOR ', ') AS** LanguagesSpoken: Selects the country name and concatenates languages with commas.

**INNER JOIN** countrylanguage **cl ON** c.Code **=** cl.CountryCode: Joins the country and countrylanguage tables based on the Code column.

**WHERE** c.Code = 'CHE': Filters for Switzerland.

**GROUP BY** c.Name: Groups the results by country name.



A black screen with white text and yellow text

Description automatically generatedCreate an SQL statement to find the capital of Spain (ESP).

This joins the country table (c) with the city table (ci) but then uses the District column in the WHERE clause.

A black and white line

Description automatically generated with medium confidence

Create an SQL statement to find the country with the highest life expectancy.

A screenshot of a computer

Description automatically generatedSELECT Name, LifeExpectancy

FROM world.country

ORDER BY LifeExpectancy DESC

LIMIT 1;

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

A black and white text on a black background

Description automatically generated

**SELECT** c.Name **AS** ‘Cities of Europe’: Select the Name column from the city table and assign the alias Cities of Europe for clarity.

**FROM** city c: Specifies the city table with the alias c.

**INNER JOIN** country co **ON** c.CountryCode = co.Code: Joins the city and country tables based on the CountryCode.

**WHERE** co.Continent **= 'Europe'**: Filters for cities in European countries.

**LIMIT 1000**: Limits the result to the first 1000 rows.

A screenshot of a computer

Description automatically generatedCreate an SQL statement to find the most populated city in the city table.

A black and white line

Description automatically generated with medium confidence

This query will select the city name and population, order the results by population in descending order, and then limit the result to the first row, which represents the city with the highest population.

# Creating an EER Diagram

A screenshot of a computer

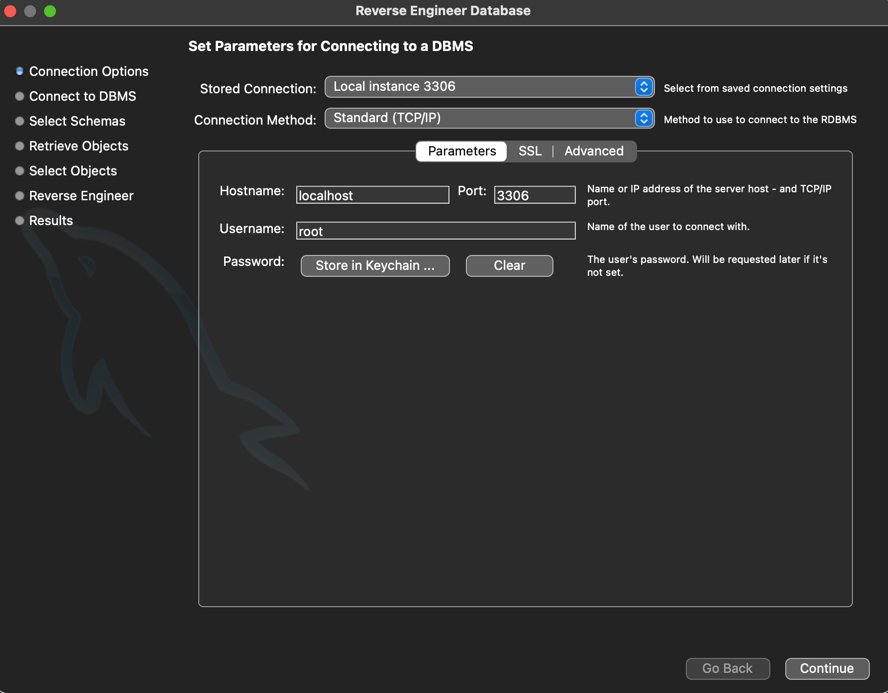
Description automatically generatedStep 1:

Selected “Database” tab on the

pop-down menu choose “Reverse Engineer”

Step 2:

Connection to Database Management System:



* Setting Parameters.

A screenshot of a computer

Description automatically generated

* Password entered

A screenshot of a computer

Description automatically generated

* Connection successful
* A screenshot of a computer

  Description automatically generatedWorld Schema selected for reverse engineering.
* A screenshot of a computer

  Description automatically generatedReentered the password and the retrieval is confirmed.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

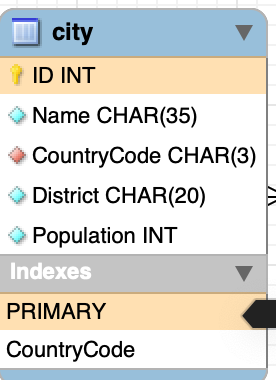
Description automatically generatedThe result

## Name of my answers.

A screenshot of a graph

Description automatically generated

Identify the primary key in the country table.



Identify the primary key in the city table.

A screenshot of a computer

Description automatically generated

Identify the primary key in the countrylanguage table.

A screenshot of a computer

Description automatically generated

Identify the foreign key in the city table.

A screenshot of a computer

Description automatically generated

Identify the foreign key in the countrylanguage table.