|  |  |  |
| --- | --- | --- |
| MySQL Assignment | | May 1  2024 |
|  | Data Technician Bootcamp | |

By Komal Shaheen

Table of Contents

**Abstract ---------------------------------------------------------------------------- pg 3**

**Aims & Objectives ----------------------------------------------------------------- pg 3**

**Part 1 ------------------------------------------------------------------------------- pg 3-9**

**Entity-Relationship Diagram (ERD) ---------------------------------------------- pg 10**

**Part 2 ------------------------------------------------------------------------------- pg 11-21**

**Entity Relationship Diagram (ERD) ---------------------------------------------- pg 22-23**

**Interview Questions-Part 1 ------------------------------------------------------- pg 23 -24**

**Interview Questions-Part 2 ------------------------------------------------------- pg 24-27**

# Abstract:

The assignment involves working with a provided Entity-Relationship Diagram (ERD). Each table has specific attributes and relationships defined within the database schema. The task includes understanding the structure of the ERD, describing the tables, their attributes, relationships, and primary keys. Additionally, SQL queries are developed based on the ERD to retrieve specific information from the database. The assignment likely encompasses both theoretical understanding of database concepts and practical SQL query writing skills.

# Aims & Objectives:

The aim of my assignment is to provide me with practical scenarios to apply my understanding of MySQL basics using MySQL Workbench. Through hands-on exercises, I will deepen my grasp of MySQL concepts and showcase my proficiency in essential database operations. My objective is to empower myself with practical skills and confidence in utilizing MySQL for fundamental database tasks, thereby preparing me for entry-level positions in database management and administration. By completing this assignment, I aim to gain the ability to create SQL queries, comprehend database structures, and effectively manipulate data within MySQL Workbench.

PART 1

# Task 1

1. **Using the Query 2 you created change the points to reads times by 10 and plus 100. Record your results in your word document.**

SELECT

last\_name,

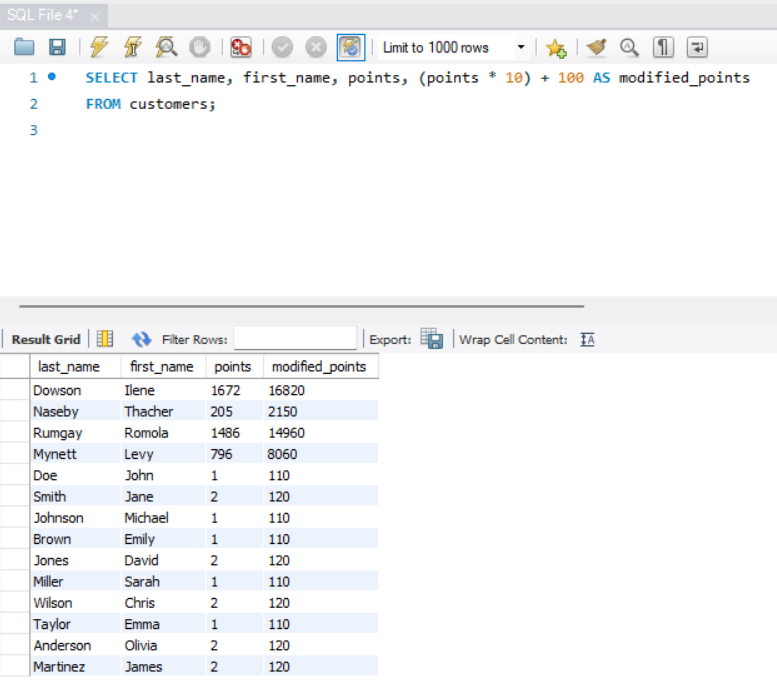
first\_name,

points,

(points \* 10) + 100 AS modified\_points

FROM

customers;



1. **Change the Query 2 code to create a discount factor so the table now shows a discount header and changing the (point + 10) \*100**

SELECT

last\_name,

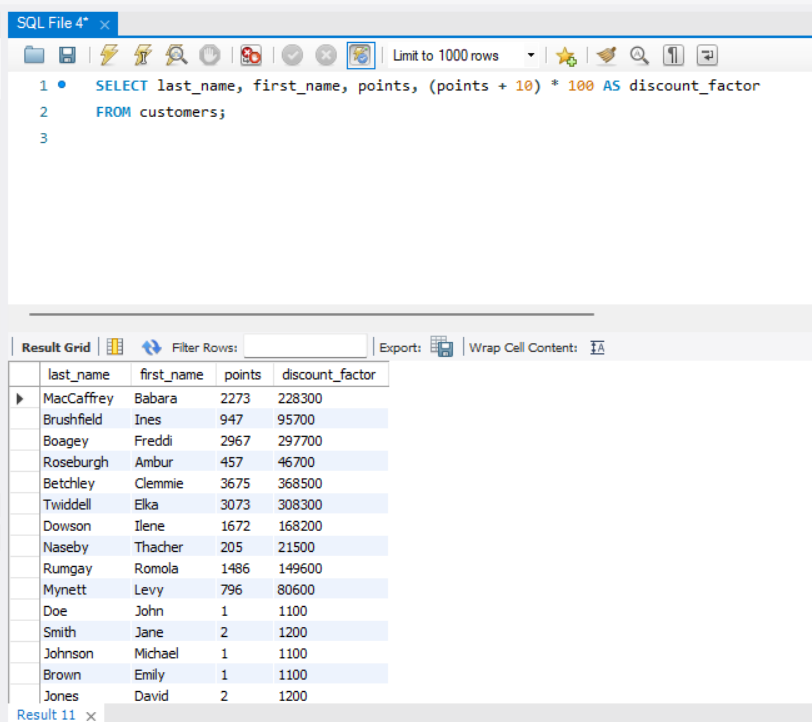
first\_name,

points,

(points + 10) \* 100 AS discount\_factor

FROM

customers;



# Task 2

**Write a SQL query to return all the products in our database in the result set. I want you to show columns; name, unit price, and new column called new price which is based on this expression, (unit price \* 1.1 ).**

**So what you are doing is increasing the product price of each by 10%.**

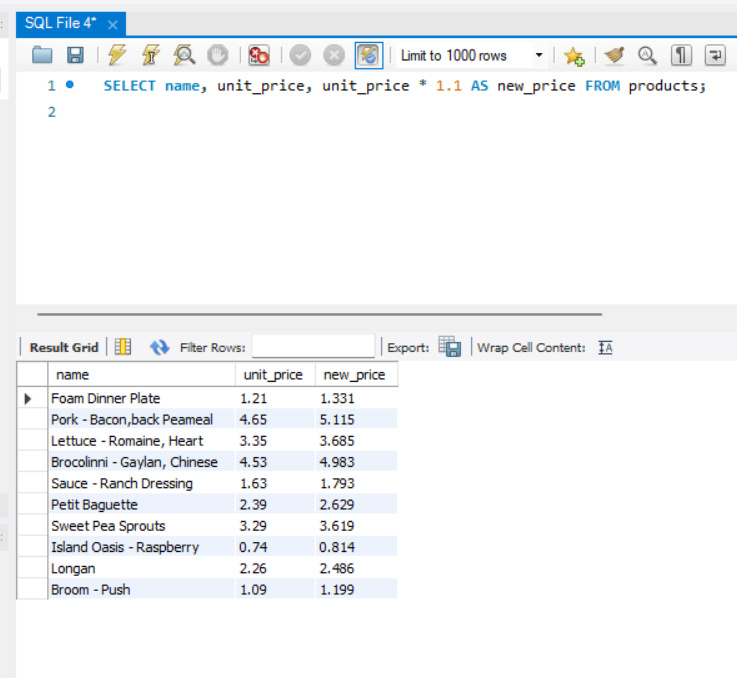
**So with the query we want all the products the original price and the new price.**

SELECT

name, unit\_price, unit\_price \* 1.1 AS new\_price

FROM

products;



# Task 3

**In this task create a new query to find all the customers with a birth date of > '1990-01-01'**

SELECT

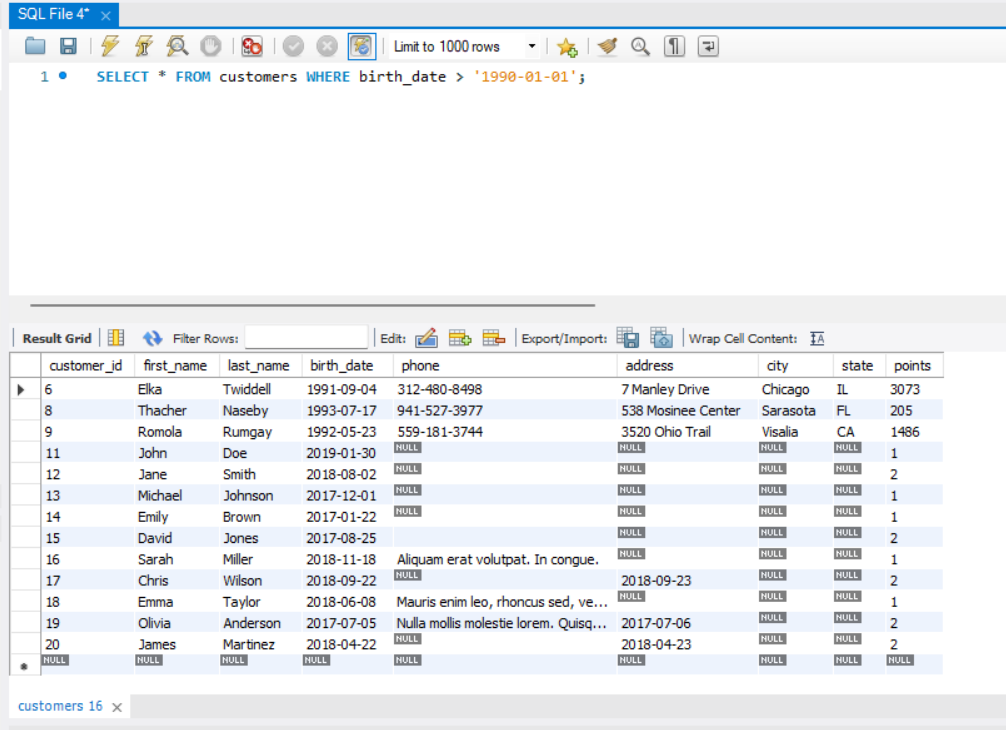
\*

FROM

customers

WHERE

birth\_date > '1990-01-01';



# Task 4

**Write a query to find out the name of the product with most amount in stock.**

SELECT

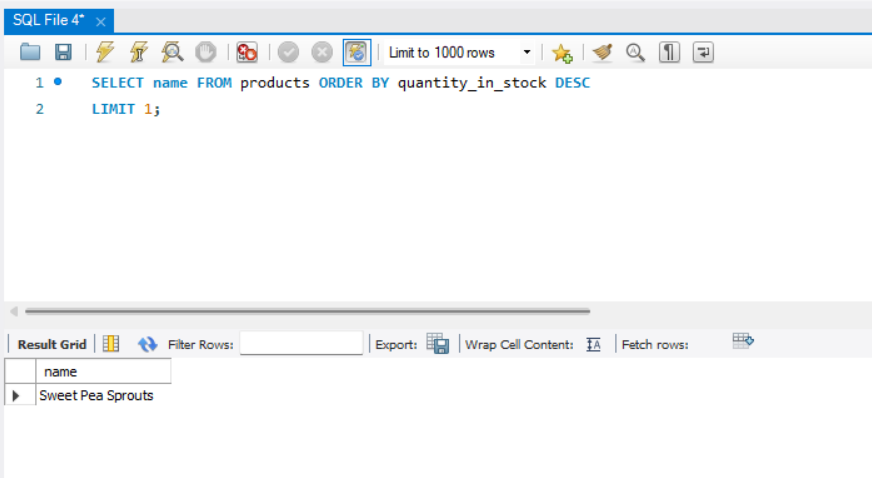
name

FROM

products

ORDER BY quantity\_in\_stock DESC

LIMIT 1;



# Task 5

**Write a query to find out the name of the most expensive product.**

SELECT

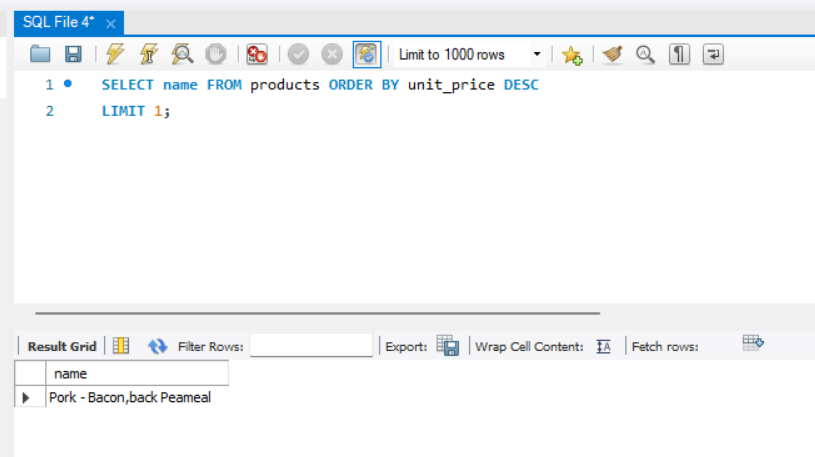
name

FROM

products

ORDER BY unit\_price DESC

LIMIT 1;



# Task 6

**Write a query to find out the first name, last name, address and the birthdate of the oldest customer.**

SELECT

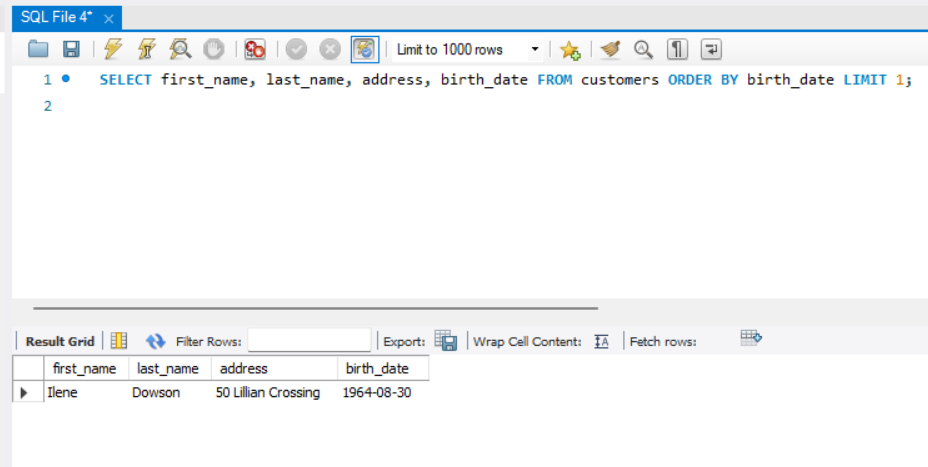
first\_name, last\_name, address, birth\_date

FROM

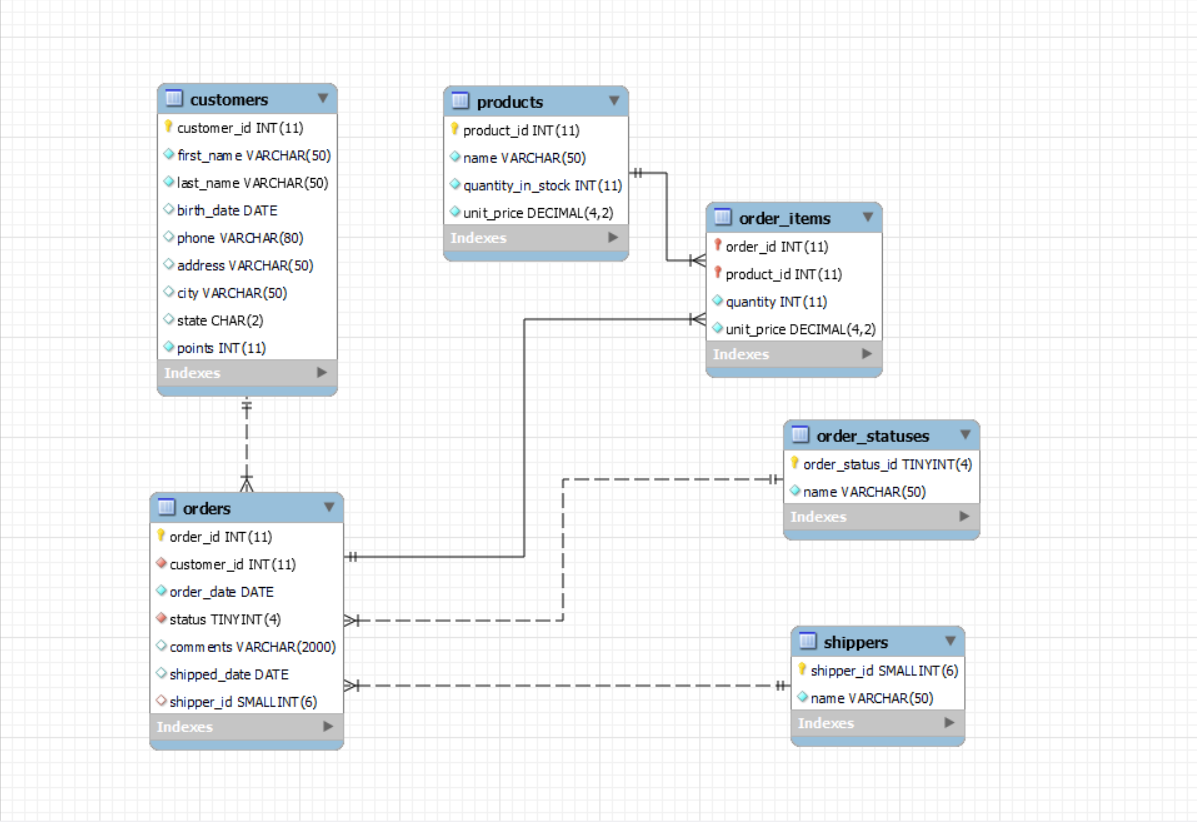
customers

ORDER BY birth\_date

LIMIT 1;



# Entity-Relationship Diagram (ERD)



## Explanation:

There are several tables representing different entities in a sql\_store's database: products, shippers, customers, order\_statuses, orders, order\_items, and order\_item\_notes.

Each table has a primary key which uniquely identifies each row: **product\_id** in products, **shipper\_id** in shippers, **customer\_id** in customers, **order\_status\_id** in order\_statuses, **order\_id** in orders, and **note\_id** in order\_item\_notes.

The relationships between these tables are established through foreign keys. For example:

The orders table has a **customer\_id** column which is a foreign key referencing the customer\_id column in the customers table, indicating that each order is associated with a customer.

The order\_items table has both **order\_id** and **product\_id** columns which are foreign keys referencing the order\_id column in the orders table and the product\_id column in the products table, respectively, indicating the products ordered within each order.

Similarly, the order\_item\_notes table has **order\_id** **and product\_id** columns as foreign keys referencing the respective columns in the orders and products tables, establishing relationships between order items and their associated notes.

These relationships allow for the retrieval and manipulation of data across multiple tables in the database, facilitating various operations such as querying customer orders, tracking inventory, and managing shipping information.

PART 2

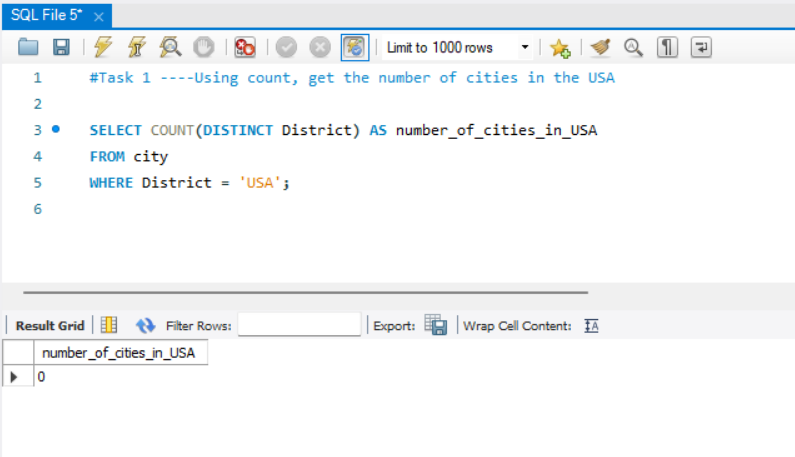
# Task 1

**Using count, get the number of cities in the USA.**

SELECT COUNT(DISTINCT District) AS number\_of\_cities\_in\_USA

FROM city

WHERE District = 'USA';



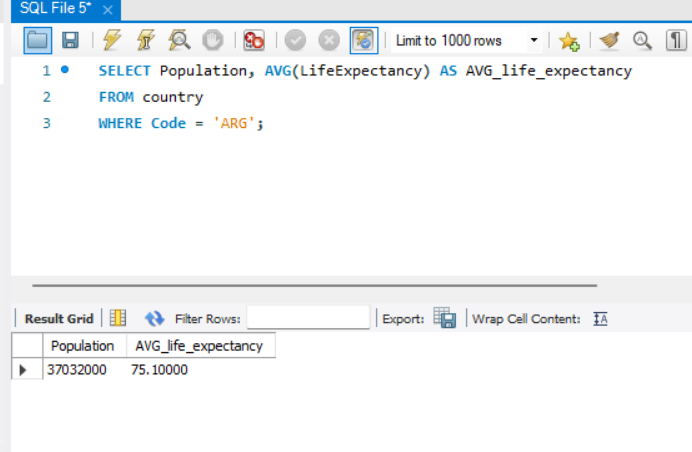
# Task 2

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

SELECT Population, AVG(LifeExpectancy) AS AVG\_life\_expectancy

FROM country

WHERE Code = 'ARG';



# Task 3

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

SELECT

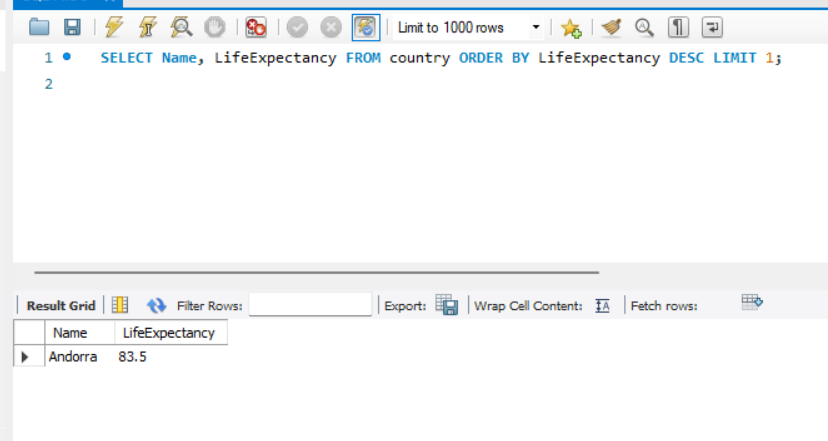
Name, LifeExpectancy

FROM

country

ORDER BY LifeExpectancy DESC

LIMIT 1;

****

# Task 4

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

SELECT

Name

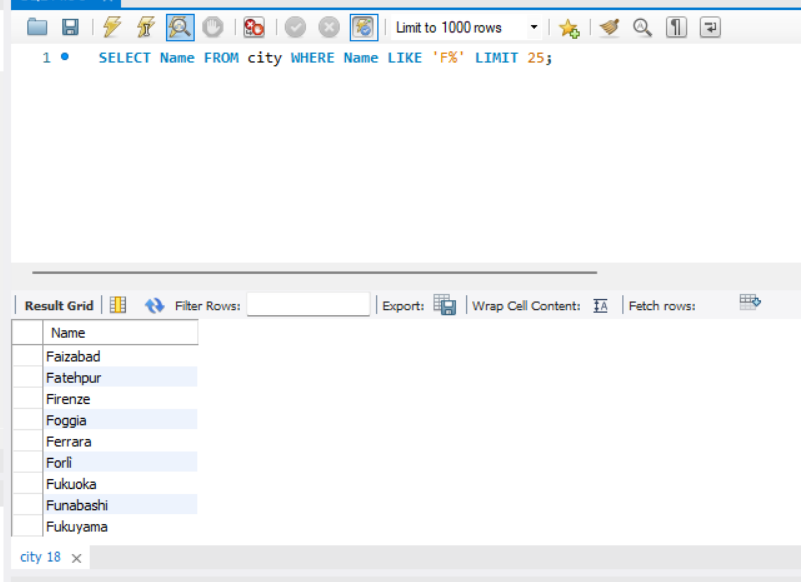
FROM

city

WHERE

Name LIKE 'F%'

LIMIT 25;



# Task 5

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

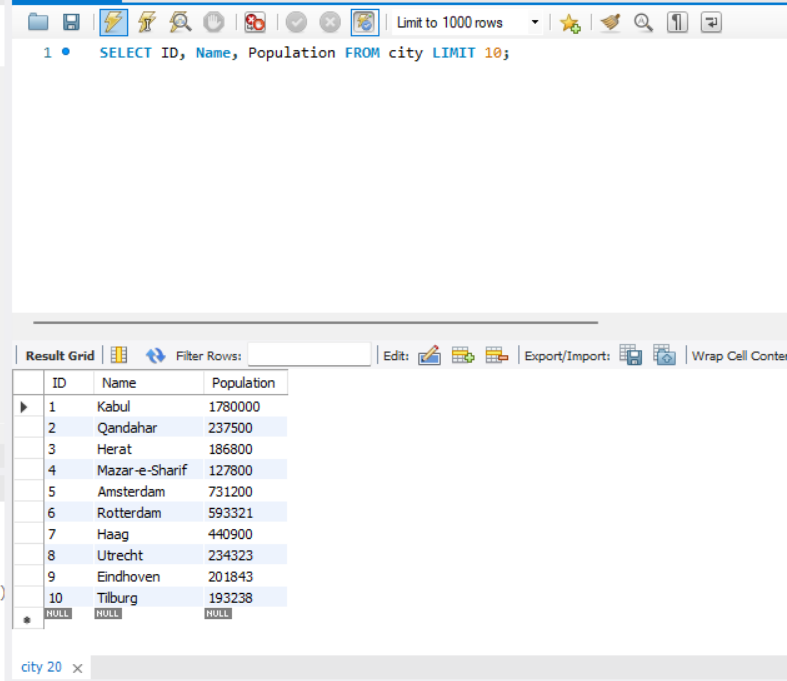
SELECT

ID, Name, Population

FROM

city

LIMIT 10;



# Task 6

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

SELECT

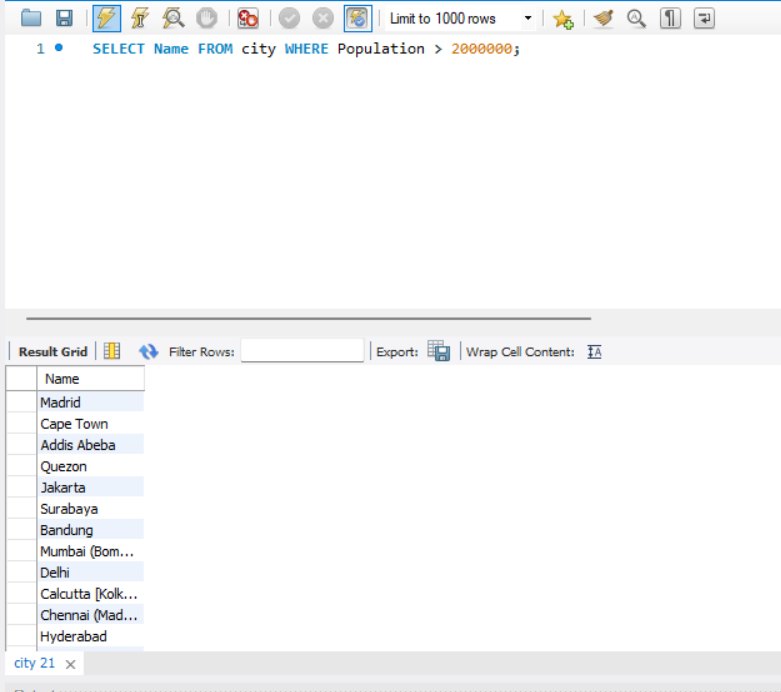
Name

FROM

city

WHERE

Population > 2000000;



# Task 7

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

SELECT

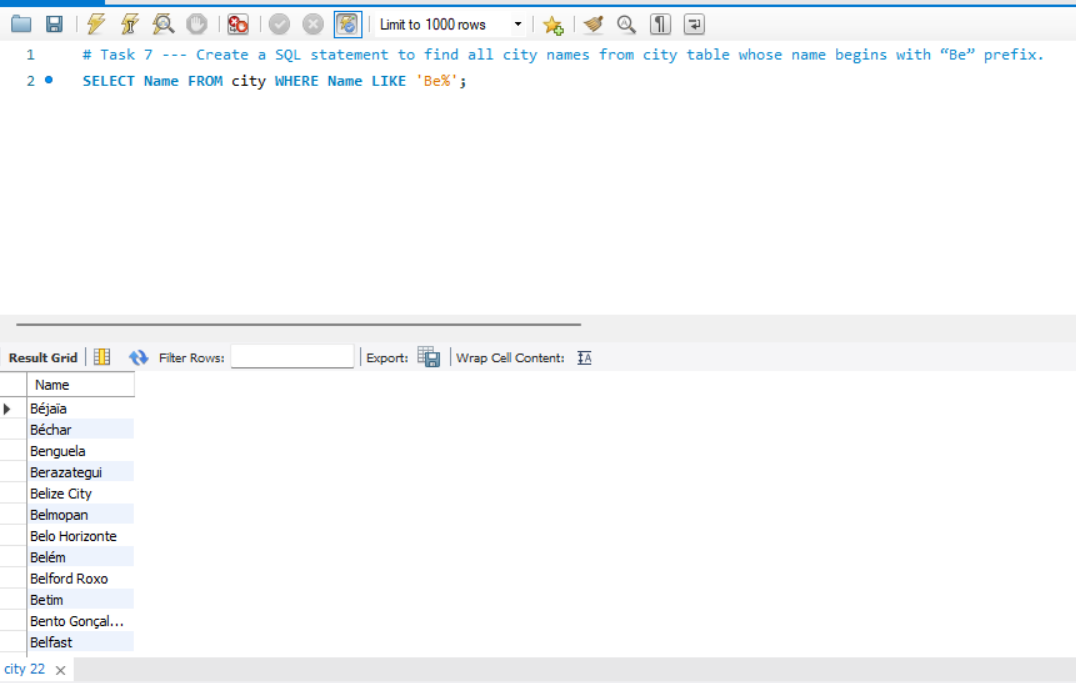
Name

FROM

city

WHERE

Name LIKE 'Be%';



# Task 8

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

SELECT

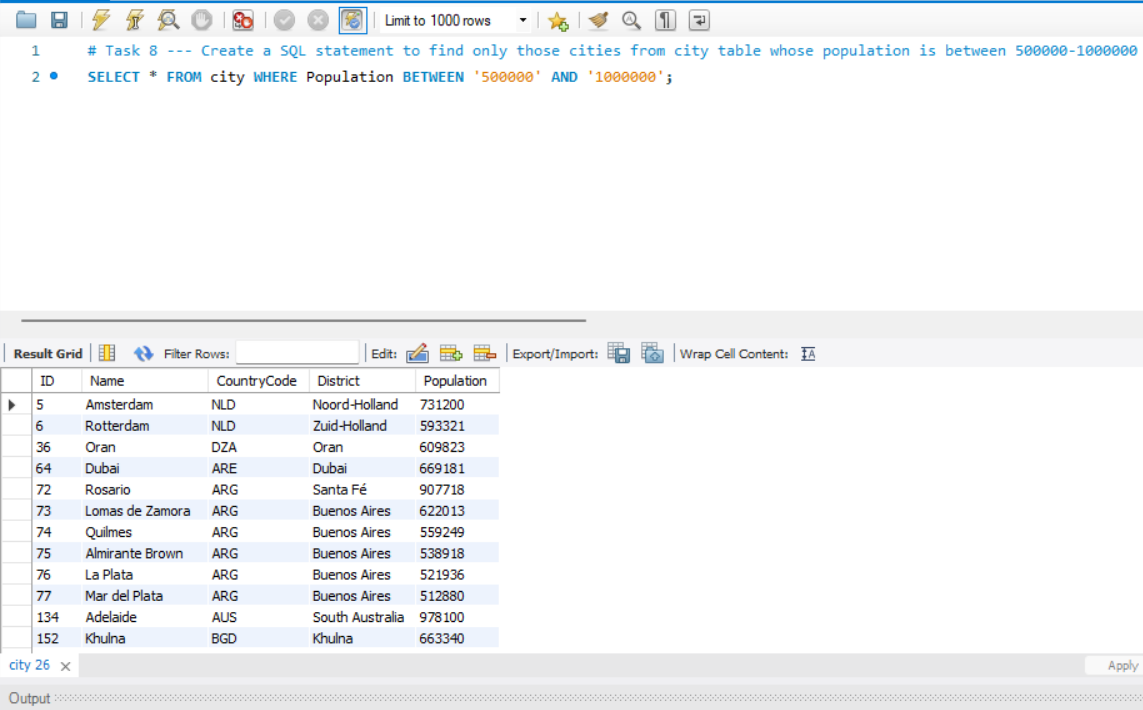
\*

FROM

city

WHERE

Population BETWEEN '500000' AND '1000000';



# Task 9

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

SELECT

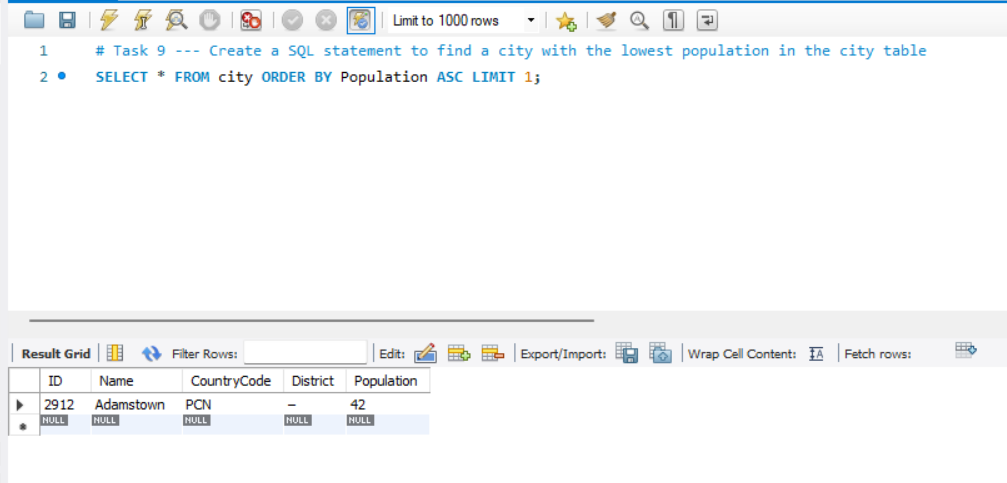
\*

FROM

city

ORDER BY Population ASC

LIMIT 1;



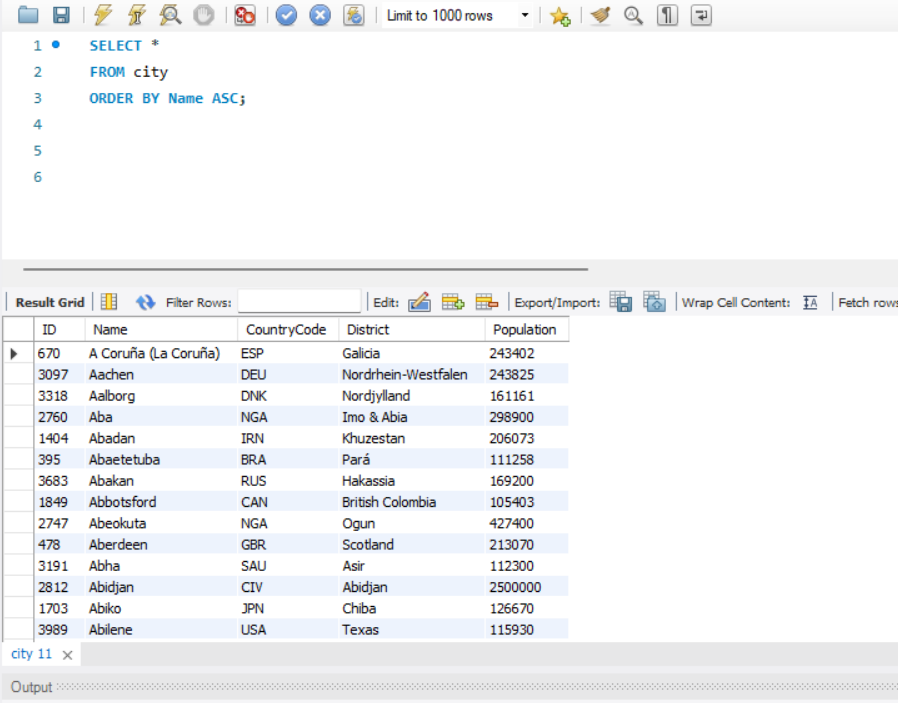
# Task 10

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

SELECT \*

FROM city

ORDER BY Name ASC;



# Task 11

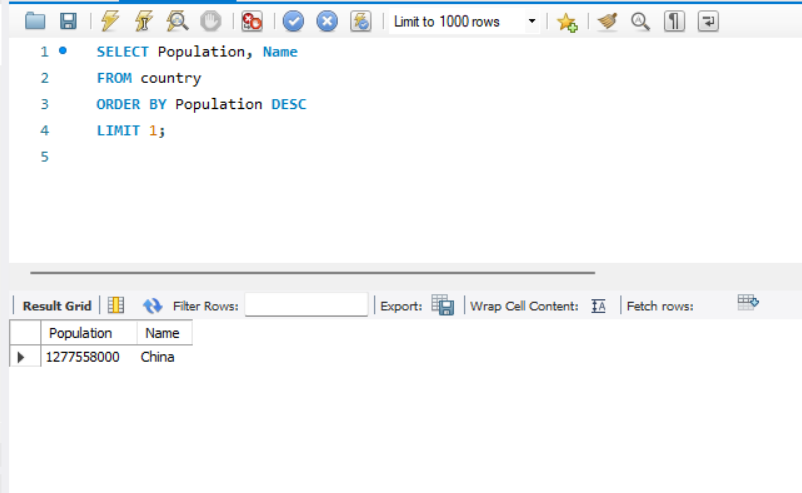
**Create a SQL statement to find a country with the largest population in the country table.**

SELECT Population, Name

FROM country

ORDER BY Population DESC

LIMIT 1;

****

OPTIONAL TASKS

# Task 1

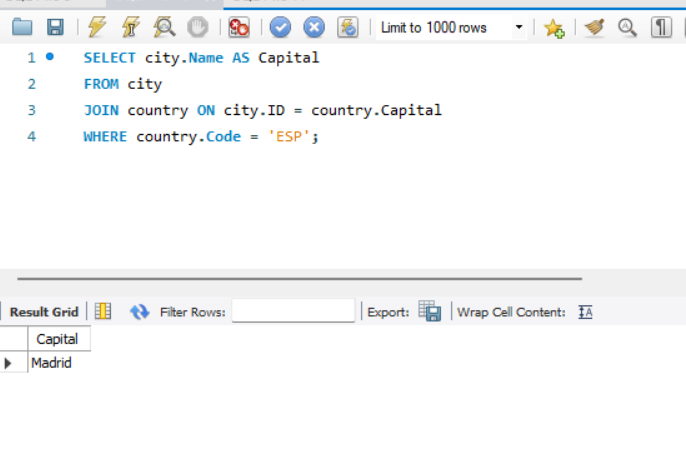
**Create a SQL statement to find the capital of Spain (ESP).**

SELECT city.Name AS Capital

FROM city

JOIN country ON city.ID = country.Capital

WHERE country.Code = 'ESP';



# Task 2

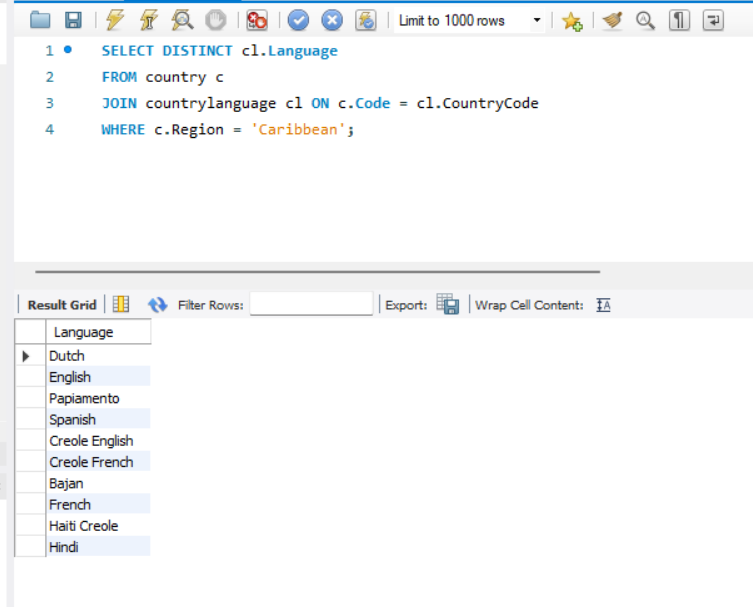
**Create a SQL statement to list all the languages spoken in the Caribbean region.**

SELECT DISTINCT cl.Language

FROM country c

JOIN countrylanguage cl ON c.Code = cl.CountryCode

WHERE c.Region = 'Caribbean';



# Task 3

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

SELECT Name AS City

FROM city

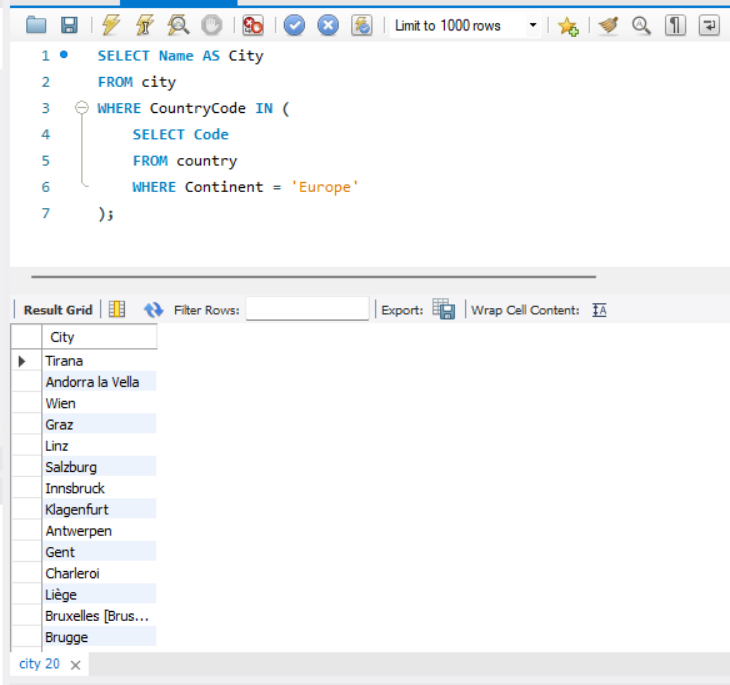
WHERE CountryCode IN (

SELECT Code

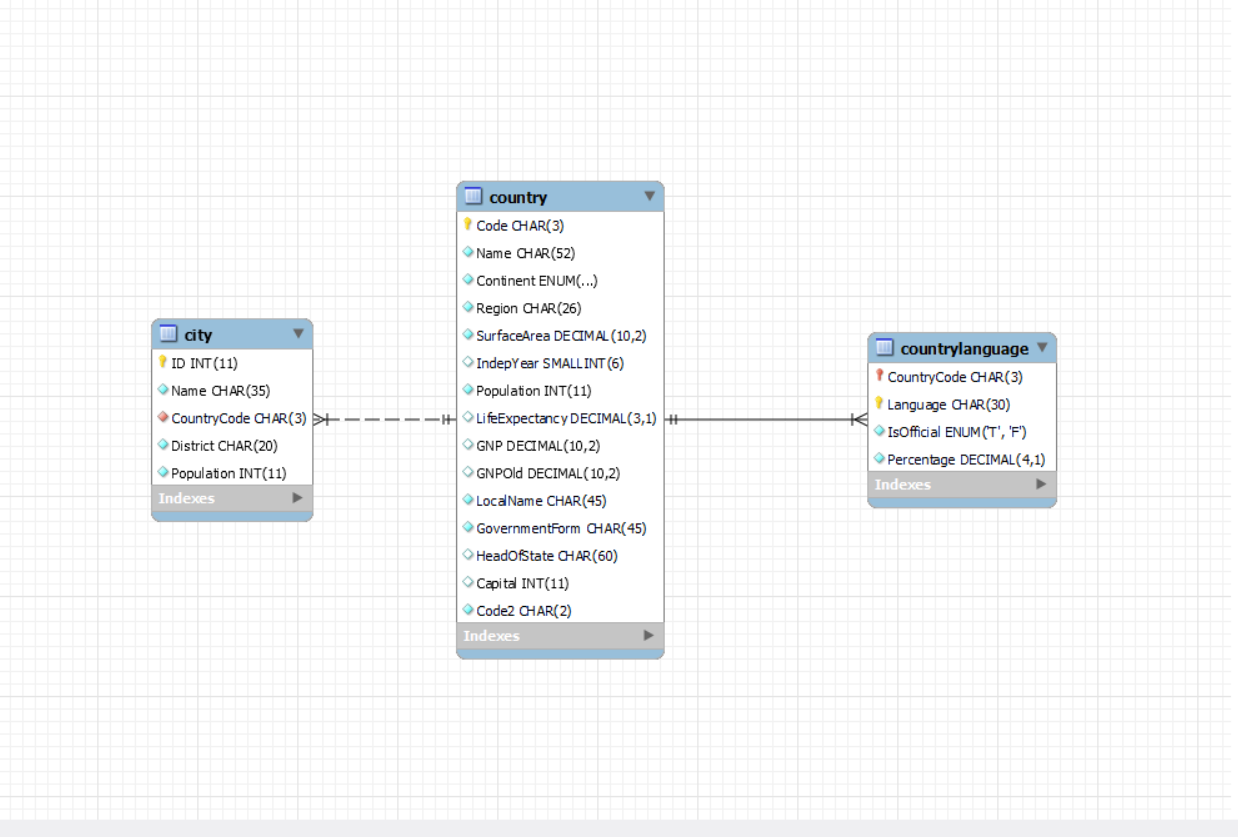
FROM country

WHERE Continent = 'Europe'

);



# Entity-Relationship Diagram (ERD)

****

## Explanation:

The Entity-Relationship Diagram (ERD) depicts three main tables: city, country, and countrylanguage. Here's a summary of the ERD:

**City Table:** This table holds data about cities, featuring columns like ID (serving as the primary key), Name, CountryCode (a foreign key connecting to the Code column in the country table), District, and Population.

**Country Table:** It encompasses various attributes regarding countries such as Code (the primary key), Name, Continent, Region, SurfaceArea, IndepYear, Population, LifeExpectancy, GNP, GNPOld, LocalName, GovernmentForm, HeadOfState, Capital, and Code2.

**Countrylanguage Table**: This table stores information concerning languages spoken in different countries. Its columns include CountryCode (a foreign key referencing the Code column in the country table), Language, IsOfficial, and Percentage. The primary key is a combined key involving CountryCode and Language.

**Regarding relationships:**

A one-to-many relationship exists between the **country** table and the **city** table, signifying that a country can have multiple cities, but each city pertains to only one country.

Similarly, there's a one-to-many relationship between the **country** table and the **countrylanguage** table, implying that a country can have multiple languages spoken, yet each language is affiliated with only one country.

**As for primary keys:**

In the **city** table, the primary key is the **ID** column.

In the **country** table, it's the **Code** column.

In the **countrylanguage** table, the primary key is a composite key, combining the **CountryCode** and **Language** columns.

Overall, the ERD offers a visual representation of the database schema, elucidating the relationships among tables and the unique identifiers for records in each table.

Interview Questions - Part 1

**Q1. What is Query?**

**Ans:** A query is a request for information or action made to a database management system, typically written in SQL, to retrieve, manipulate, or perform operations on data stored within a database.

**Q2. What is SELECT statement?**

**Ans:** The SELECT statement is a fundamental SQL command used to retrieve data from a database. It allows users to specify which columns of data they want to retrieve and from which table or tables. Additionally, it provides options for filtering, sorting, and grouping the results returned by the database. The SELECT statement is versatile and can be customized to meet specific data retrieval needs, making it a powerful tool in database querying.

**Q3. What is the WHERE clause?**

**Ans:** The WHERE clause is a fundamental component of SQL queries that allows users to filter rows of data based on specific conditions. When included in a SELECT statement, the WHERE clause specifies criteria that must be met for a row to be included in the query result.

**Q4. What is the Primary Key?**

**Ans:** The primary key is a fundamental concept in relational databases that uniquely identifies each record (or row) in a table. It serves as a unique identifier for each record within the table and ensures that there are no duplicate or null values for this key.

Key characteristics of a primary key include:

1. **Uniqueness:** Each value in the primary key column must be unique within the table. This uniqueness constraint prevents duplicate records from being inserted into the table.
2. **Non-nullability:** The primary key column cannot contain null values. Every record in the table must have a valid value for the primary key column.
3. **Single-column or composite:** A primary key can consist of one or multiple columns. In the case of a composite primary key, the combination of values across the columns must be unique.

**Q5. What is the Database?**

**Ans:** A database is a structured collection of data that is organised and stored in a computer system. It is designed to efficiently manage, manipulate, and retrieve data according to specific requirements. Databases are used to store various types of information, such as customer records, product details, financial transactions, and more. They provide mechanisms for data integrity, security, and concurrency control to ensure reliable and consistent access to data. Databases can range from simple flat file systems to complex relational database management systems (RDBMS) like MySQL, PostgreSQL, Oracle, and SQL Server, among others.

Interview Questions – Part 2

**Q1.** **List the different types of relationships in SQL and give examples.**

**Ans:** In SQL databases, the following types of relationships commonly exist:

1. **One-to-One (1:1) Relationship:**

**Example:** Consider a database where each employee is assigned a unique employee ID. Each employee may have one and only one office location assigned to them, and each office location is assigned to only one employee.

1. **One-to-Many (1:N) Relationship:**

**Example:** In a database of students and their courses, each student can enrol in multiple courses. However, each course is typically taught by only one instructor.

1. **Many-to-One (N:1) Relationship:**

**Example:** In an e-commerce database, multiple orders can be placed by different customers. However, each order is associated with only one customer.

1. **Many-to-Many (N:M) Relationship:**

**Example:** A database modelling students and courses in a university where each student can enrol in multiple courses, and each course can have multiple students enrolled.

**Q2. What is Normalization?**

**Ans:** Normalization is the process of organizing data in a database efficiently. This process involves reducing data redundancy and dependency by dividing large tables into smaller ones and defining relationships between them. The main objective of normalization is to eliminate data anomalies and ensure data integrity.

There are several normal forms (1NF, 2NF, 3NF, BCNF, etc.), each with specific rules for achieving a particular level of normalization. These normal forms aim to ensure that each table in the database has a clear purpose, with minimal redundancy and dependency.

Normalization helps improve database efficiency, reduces the likelihood of data anomalies such as update anomalies, insertion anomalies, and deletion anomalies, and facilitates easier database maintenance and modification. It is an essential concept in database design and is widely used to optimize database performance and maintain data consistency.

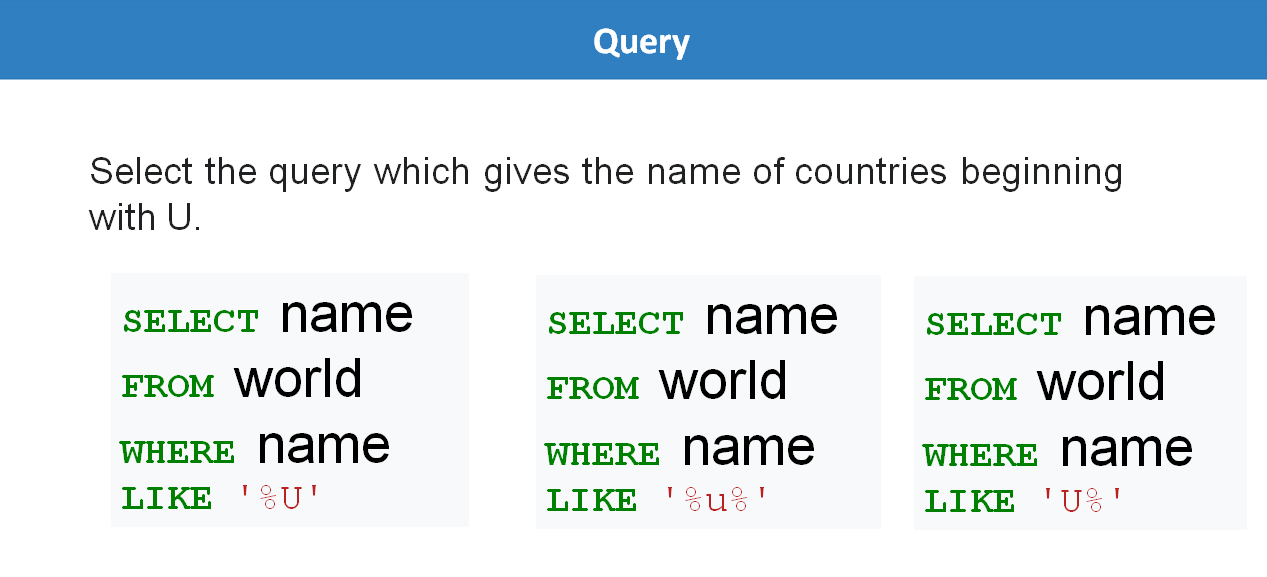
**Q3. Modify query to show the population of Germany.**

SELECT population FROM world

WHERE name = 'France'

**Ans:** SELECT population FROM world

WHERE name = 'Germany';

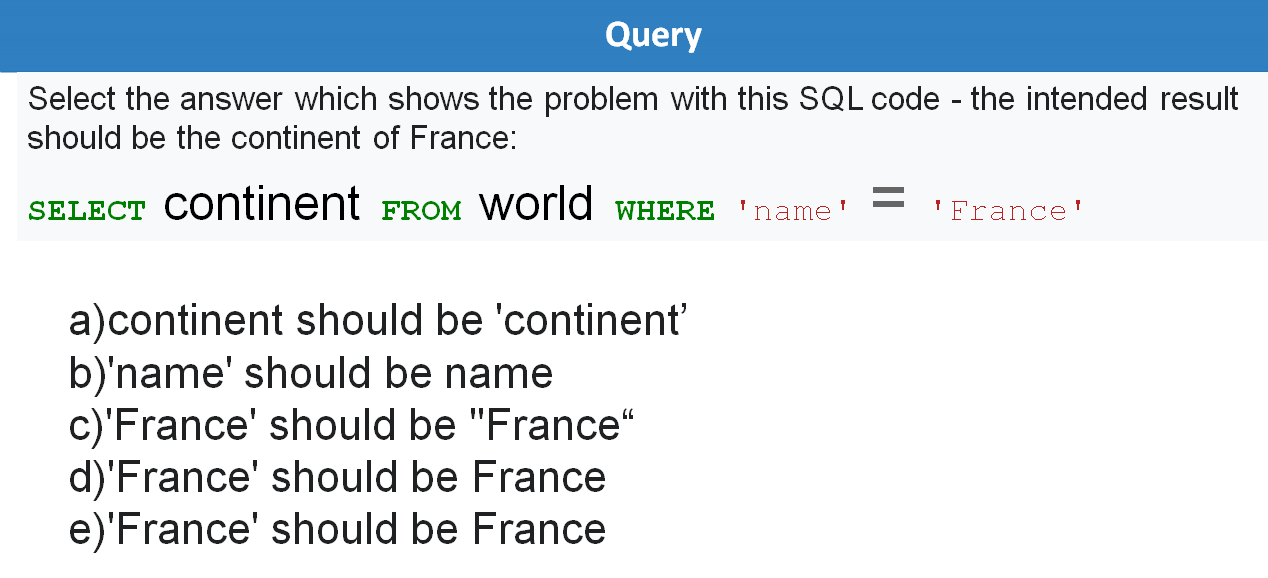
**Q4.** 

**Ans:** SELECT name

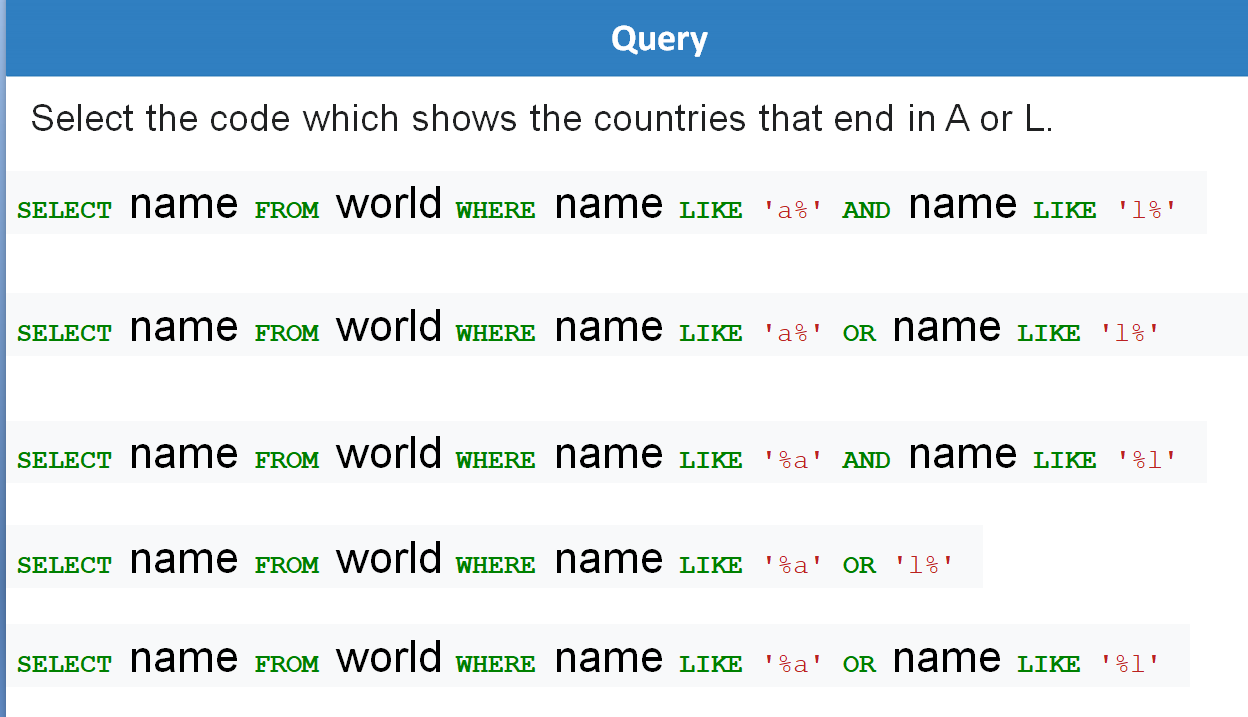
FROM world

WHERE name

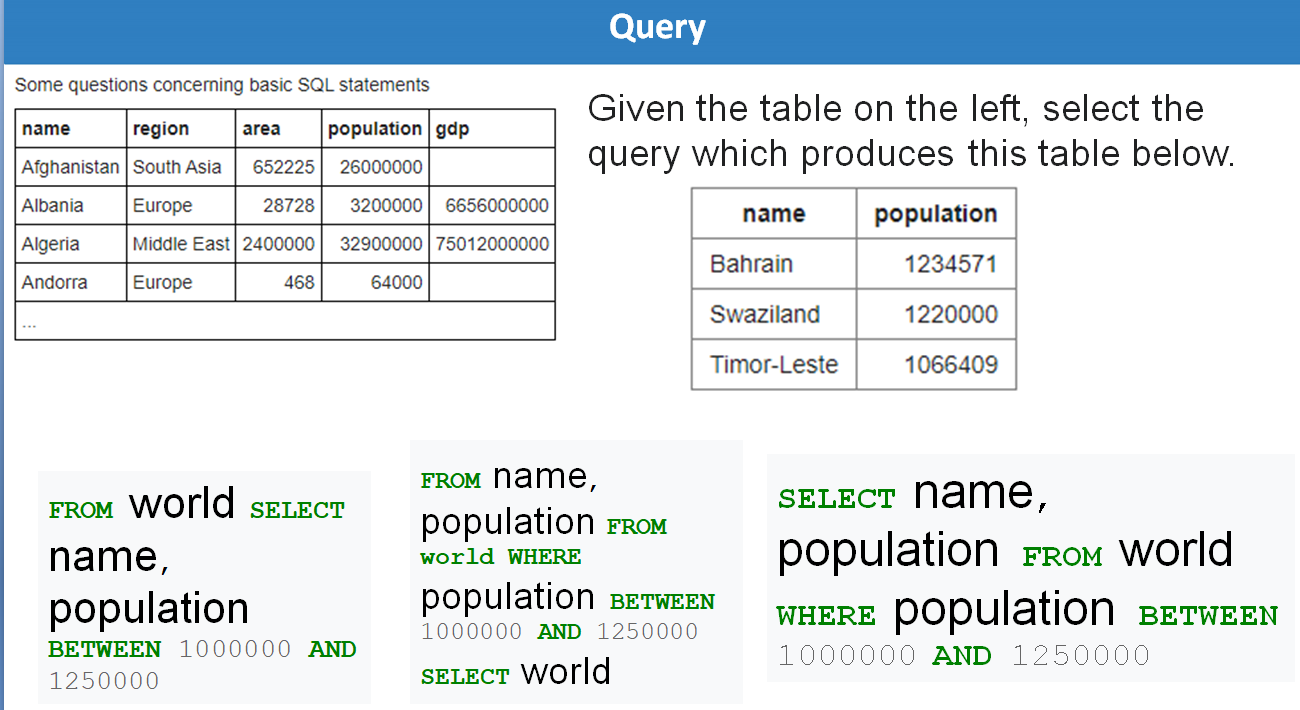
LIKE ‘U%’;

**Q5. **

**Ans:** b

**Q6:** 

**Ans:** SELECT name FROM world WHERE name LIKE ‘%a’ OR name LIKE ‘%l’;

**Q7. **

**Ans:** SELECT name, population

FROM world

WHERE population

BETWEEN 10000000 AND 1250000