Just IT

B2W group

Apprenticeships | Training | Recruitment

Data Technician

Name: Rebecca Osewa

Course Date: 18/12/2025

Table of contents

Day 1: Task 1

Please research and complete the below questions relating to key concepts of databases.

What is a primary key?	A primary key is a column, or a combination of columns, in a database table that uniquely identifies each record in that table. It ensures that no two rows have the same value in the primary key field, maintaining the integrity of the data. A primary key must always contain a unique value and cannot be left empty, as every record must be identifiable. For example, in a student's table, a Student ID column could serve as the primary key, as each student would have a unique ID that distinguishes them from all others. In essence, the primary key acts as the main identifier for each record in a table, helping to organise and retrieve data efficiently.
How does this differ from a secondary key?	A secondary key is a column or set of columns used to access or organise data in a table but does not uniquely identify each record. Unlike a primary key, a secondary key can have duplicate values and may contain nulls. It is primarily used for searching, sorting, or creating indexes to improve query performance. Essentially, while a primary key ensures uniqueness, a secondary key is used to make data retrieval more efficient.
How are primary and foreign keys related?	A foreign key is a column, or set of columns, in one table that refers to the primary key in another table. This relationship links the two tables and ensures integrity, meaning that values in the foreign key column must match existing values in the primary key column of the referenced table. The primary key uniquely identifies a record in its own table, while the foreign key allows other tables to reference that record. This connection is fundamental for establishing relationships between tables in a relational database.
Provide a real-world example of a one-to-one relationship	A real-world example of a one-to-one relationship is the relationship between a person and their passport**.** Each person can have only one passport**,** and each passport is assigned to only one person. In a database, this could be represented with a person table and a passport table, where the Person ID serves as the primary key in the person table and as a foreign key in the passport table. This ensures that each passport record is linked to exactly one person, and no person has more than one passport in the system.
Provide a real-world example of a one-to-many relationship	A real-world example of a one-to-many relationship is the relationship between a teacher and their students. One teacher can teach many students, but each student is assigned to only one teacher in that context. In a database, this could be represented with a teacher table and a student table, teacher ID is the primary key in the teacher table and serves as a foreign key in the student table. This setup allows multiple student records to be associated with a single teacher while maintaining the link to the correct teacher.

Provide a real-world example of a many-to-many relationship	A real-world example of a many-to-many relationship is the relationship between students and courses at a school or university. A student can enrol in multiple courses, and each course can have many students. In a database, this is typically handled with a junction table (or linking table) called something like student courses, which contains foreign keys referencing both the students table and the courses table. This allows the database to track which students are enrolled in which courses without duplicating data in either table.

Day 1: Task 2

Please research and complete the below questions relating to key concepts of databases.

What is the difference between a relational and non-relational database?	A relational database organises data into tables, also called relations, which consist of rows and columns. Each row represents a record, and each column represents a field of data. Relationships between tables are defined using keys, allowing for complex queries and data integrity. Relational databases use Structured Query Language (SQL) for managing and querying data. They are best suited for structured data where consistency and relationships are important. A non-relational database, also known as a NoSQL database, stores data in formats other than tables, such as documents, key-value pairs, graphs, or wide-column stores. These databases are designed for flexibility, scalability, and handling large volumes of unstructured or semi-structured data. They do not rely on SQL, and relationships between data are often managed differently, if at all. Non-relational databases are ideal for applications that require rapid development, horizontal scaling, and handling diverse data types
What type of data would benefit off the non-relational model? Why?	A non-relational (NoSQL) model is typically unstructured Examples include: Social media content: Posts, comments, likes, and multimedia that vary in format. IoT sensor data: Continuous streams of readings that may not have a fixed structure. E-commerce product catalogues: Items with varying attributes, such as colour, size, specifications. User-generated content: Reviews, messages, or logs that differ in structure between users. Big data analytics: Large datasets that require horizontal scaling and fast write/read operations. Non-relational databases are flexible and do not require a fixed schema, which makes them ideal for data that changes frequently or has inconsistent formats. They can scale horizontally, handling very large volumes of data efficiently. Additionally, they allow for fast read

and write operations, which is essential for real-time applications like social media feeds, analytics dashboards, and IoT systems.

Day 2: Task 1

You are a junior data analyst at **Northwind Traders**. Your manager has assigned you a series of real-world business tasks. For each scenario:

1. **Write an SQL query** to solve the request. Write an SQL query to solve the request.
2. Run the query in MySQL Workbench using the Northwind database.
3. Paste your query in the box below each question as evidence of completion.

---

1. 📘 *Retrieve Full Customer Data*

Your manager has asked you to export the full list of all customer details into a report.

➡️ **Write a SQL query to retrieve all columns from the Customers table.**

```
Select*

From Customers;
```

---

2. 📘 *Customer Names and Cities for Marketing*

A marketing analyst is targeting a campaign based on customer names and their cities.

➡️ **Write a SQL query to retrieve only the CustomerName and City columns from the Customers table.**

```
SELECT CustomerName, City

FROM Customers;
```

---

3. 📘 *Unique Cities for Delivery Network Expansion*

The logistics team wants to know all the different cities where customers are located (no duplicates).

➡️ **Write a SQL query to retrieve distinct values from the City column in the Customers table.**

```
SELECT DISTINCT City

FROM Customers;
```

4. 📘 *High-Value Products Report*

The product manager wants to analyse products priced over £50.

➡️ **Write a SQL query to retrieve all columns from the Products table where the Price is greater than 50.**

```
SELECT *
FROM Products
WHERE Price > 50;
```

---

5. 📘 *International Customers Targeting (USA & UK)*

The marketing team is preparing campaigns for customers in the USA and UK.

➡️ **Write a SQL query to retrieve all columns from the Customers table where the Country is either 'USA' or 'UK'.**

```
SELECT *
FROM Customers
WHERE Country IN ('USA', 'UK');
```

---

6. 📘 *Recent Orders Report*

Your team needs to analyze the latest orders.

➡️ **Write a SQL query to retrieve all columns from the Orders table, sorted by OrderDate in descending order.**

```
SELECT *
FROM Orders
ORDER BY OrderDate DESC;
```

---

7. 📘 *Mid-Range Products Listing*

You are preparing a product list for items priced between £20 and £50.

➡️ **Write a SQL query to retrieve all columns from the Products table where the Price is between 20 and 50, ordered by descending Price.**

```
SELECT*
FROM Products
WHERE Price BETWEEN 20 AND 50
ORDER BY Price DESC;
```

8. 📘 *Local Marketing in the US (Portland & Kirkland)*

You're focusing on customers from Portland and Kirkland in the USA.

➡️ **Write a SQL query to retrieve all columns from the Customers table where the Country is 'USA' and City is either 'Portland' or 'Kirkland', ordered by ascending CustomerName.**

```
SELECT *
FROM Customers
WHERE Country = 'USA'
  AND City IN ('Portland', 'Kirkland')
ORDER BY CustomerName ASC;
```

---

9. 📘 *Customers from the UK or London*

Prepare a list for promotional emails targeting customers from the UK or those based in London.

➡️ **Write a SQL query to retrieve all columns from the Customers table where the Country is 'UK' or City is 'London', ordered by descending CustomerName.**

```
SELECT *
FROM Customers
WHERE Country = 'UK'
   OR City = 'London'
ORDER BY CustomerName DESC;
```

---

10. 📘 *Product Inventory for Selected Categories*

The inventory team needs products from Category 1 or 2, sorted alphabetically.

➡️ **Write a SQL query to retrieve all columns from the Products table where the CategoryID is 1 or 2, ordered by ascending ProductName.**

```
SELECT *
FROM Products
WHERE CategoryID IN (1, 2)
ORDER BY ProductName ASC;
```

Day 3: Task 1

Please research the below 'JOIN' types, explain what they are and provide an example of the types of data it would be used on.

Self-join	When it comes to Self-join this is when a table is joined with itself. This becomes useful when you want to compare rows within the same table. Question: Which employee reports to which manager, when both employees and managers are stored in the same table Select e.name AS employee, m.name as manager from employees e left join employees m
Right join	A Right Join returns all records from the right table, and the matched records from the left table. If there's no match, null appears for the left table's columns. SELECT Orders.order_id, Customers.name FROM Orders

	RIGHT JOIN Customers ON Orders.customer_id = Customers.id;
Full join	A full outer join returns all records from both tables, with null for missing matches on either side. Example SELECT e.name AS Employee, c.name AS Contractor FROM Employees e FULL JOIN Contractors c ON e.id = c.id;
Inner join	When it comes to Inner Join it returns only the records when there is a match in both tables. Example SELECT Orders.order_id, Customers.name FROM Orders INNER JOIN Customers ON Orders.customer_id = Customers.id;
Cross join	Cross Join returns the Cartesian product of two tables-all possible combinations of rows. Example SELECT Products.name, Discounts.discount_code FROM Products CROSS JOIN Discounts;

Left join	Left join returns all records from the left table, and the matched records from the right table. If there's no match, again the word null will appear for the right table's columns. Example SELECT Customers.name, Orders.order_id FROM Customers LEFT JOIN Orders ON Customers.id = Orders.customer_id;

Day 3: Task 2

You are a junior data analyst at **Northwind Traders**. Your manager has assigned you a series of real-world business tasks. For each scenario:

1. **Write an SQL query** to solve the request. Write an SQL query to solve the request.
2. Run the query in MySQL Workbench using the Northwind database.
3. Paste your query in the box below each question as evidence of completion.

1. 📘 *Linking Products to Suppliers*

Management wants to know which supplier provides each product in the inventory.

➡️ **Write a SQL query to find the supplier of each product.**

```
SELECT
    p.ProductName,
    s.CompanyName AS SupplierName
FROM
    Products p
JOIN
    Suppliers s ON p.SupplierID = s.SupplierID
ORDER BY
    p.ProductName;
```

---

2. 📘 *Classifying Products by Category*

The category manager is reviewing how products are organized.

➡️ **Write a SQL query to find the category of each product.**

```
SELECT
    p.ProductName,
    c.CategoryName
FROM
    Products p
JOIN
    Categories c ON p.CategoryID = c.CategoryID
ORDER BY
    p.ProductName;
```

---

3. 📘 *Category-Specific Product Report: Meat/Poultry*

The food department wants to view all items in the *Meat/Poultry* category.

➡️ **Write a SQL query to retrieve all products belonging to the Meat/Poultry category.**

```
SELECT
    p.ProductName,
    c.CategoryName
FROM
    Products p
JOIN
    Categories c ON p.CategoryID = c.CategoryID
WHERE
    c.CategoryName = 'Meat/Poultry'
ORDER BY
    p.ProductName;
```

---

4. 📘 *Complete Order Overview*

The business team wants to see a detailed order list with customer and employee information.

➡️ **Write a SQL query to retrieve the Order ID, Order Date, Customer Name, and Employee Name for all orders.**

```
SELECT
    o.OrderID,
    o.OrderDate,
    c.CompanyName AS CustomerName,
    CONCAT(e.FirstName, ' ', e.LastName) AS EmployeeName
FROM
    Orders o
JOIN
```

```
    Customers c ON o.CustomerID = c.CustomerID
JOIN
    Employees e ON o.EmployeeID = e.EmployeeID
ORDER BY
    o.OrderID;
```

---

5. 📘 *Supply Chain Overview Report*

Your manager wants to see the product name, its category, and the name of its supplier all in one report.

➡️ **Write a SQL query to retrieve the Product Name, Category Name, and Supplier Name for all products.**

```
SELECT
    p.ProductName,
    c.CategoryName,
    s.CompanyName AS SupplierName
FROM
    Products p
JOIN
    Categories c ON p.CategoryID = c.CategoryID
JOIN
    Suppliers s ON p.SupplierID = s.SupplierID
ORDER BY
    p.ProductName;
```

---

6. 📘 *Yearly Order Summary – 1996*

The team is auditing customer orders made in 1996.

➡️ **Write a SQL query to create a report for all the orders of 1996 and their customers.**

```
WHERE YEAR(o.OrderDate) = 1996
```

---

7. 📘 *Product Count by Category*

The product team wants to know how many products exist under each category.

➡️ **Write a SQL query to retrieve all categories along with the number of products in each category.**

```
SELECT
    c.CategoryName,
    COUNT(p.ProductID) AS ProductCount
FROM Categories c
LEFT JOIN Products p
    ON c.CategoryID = p.CategoryID
GROUP BY c.CategoryName
ORDER BY ProductCount DESC;
```

---

8. 📘 *Sales Volume Breakdown*

The sales department wants to analyze how much of each product was ordered and at what price.

➡️ **Write a SQL query to retrieve all products with their prices and the quantity ordered for each product.**

```
SELECT
    p.ProductID,
    p.ProductName,
    p.Price,
    SUM(od.Quantity) AS TotalQuantityOrdered
FROM Products p
LEFT JOIN OrderDetails od
    ON p.ProductID = od.ProductID
GROUP BY
    p.ProductID,
    p.ProductName,
    p.Price
ORDER BY TotalQuantityOrdered DESC;
```

Day 4: Task 1: SQL Practical

You are working as a **junior data analyst** for an international research organization analyzing countries, cities, and languages from around the world. Your job is to write and run SQL queries that provide useful global insights.
For each task:

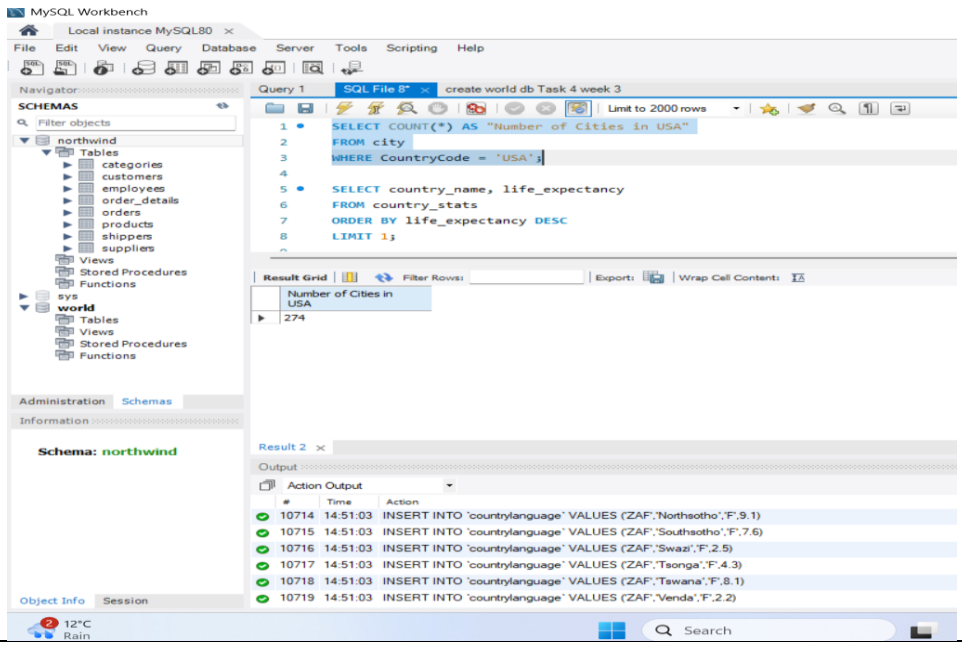
1. **Write the appropriate SQL query** to meet the scenario's objective.
2. **Run your query** using **MySQL Workbench** and the **world** database.
3. **Take a screenshot** that clearly shows both your SQL code and the result table.
4. **Paste the screenshot in the box provided for each question as evidence of completion.**

Setting up the database:

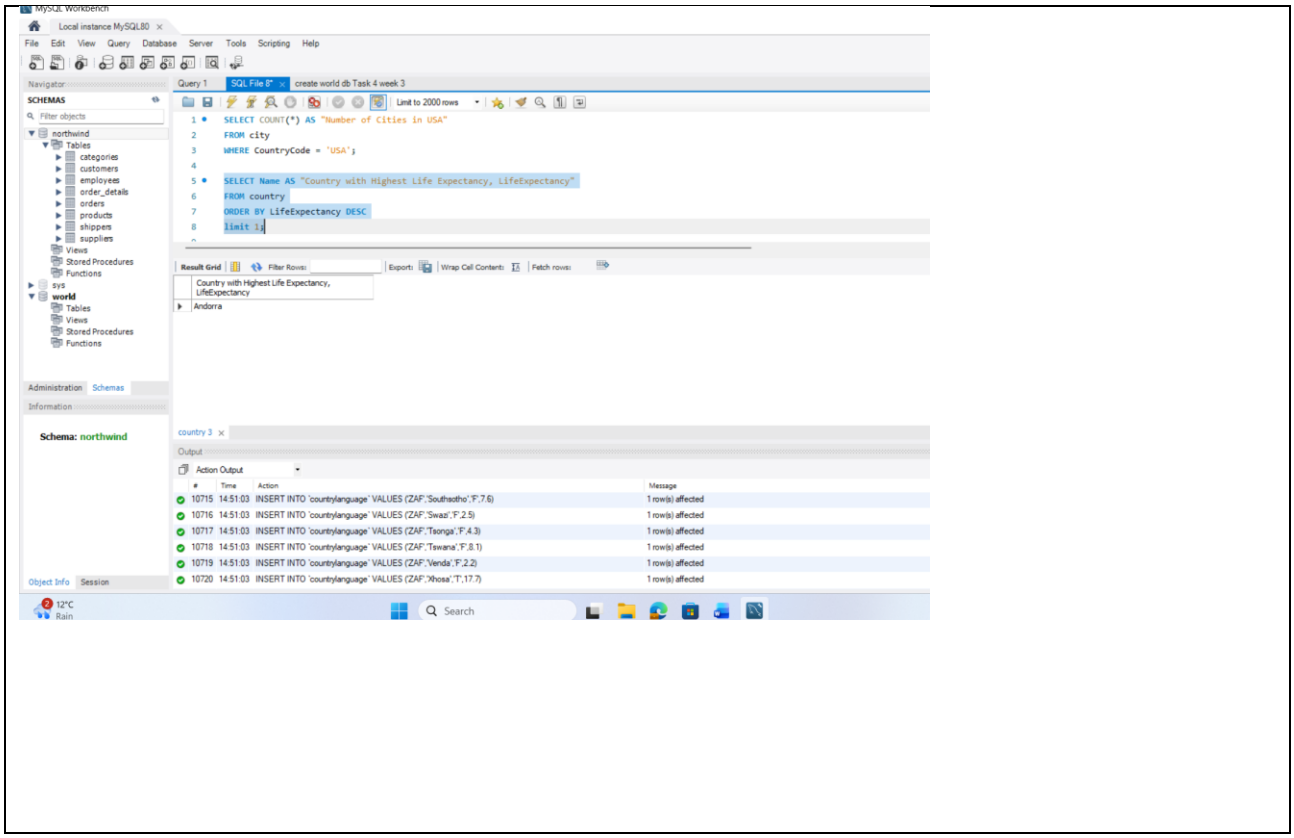
1. **Download world_db here**
2. **Follow each step to create your database here**

1. **Count Cities in USA:** *Scenario:* You've been tasked with conducting a demographic analysis of cities in the United States. Your first step is to determine the total number of cities within the country to provide a baseline for further analysis.

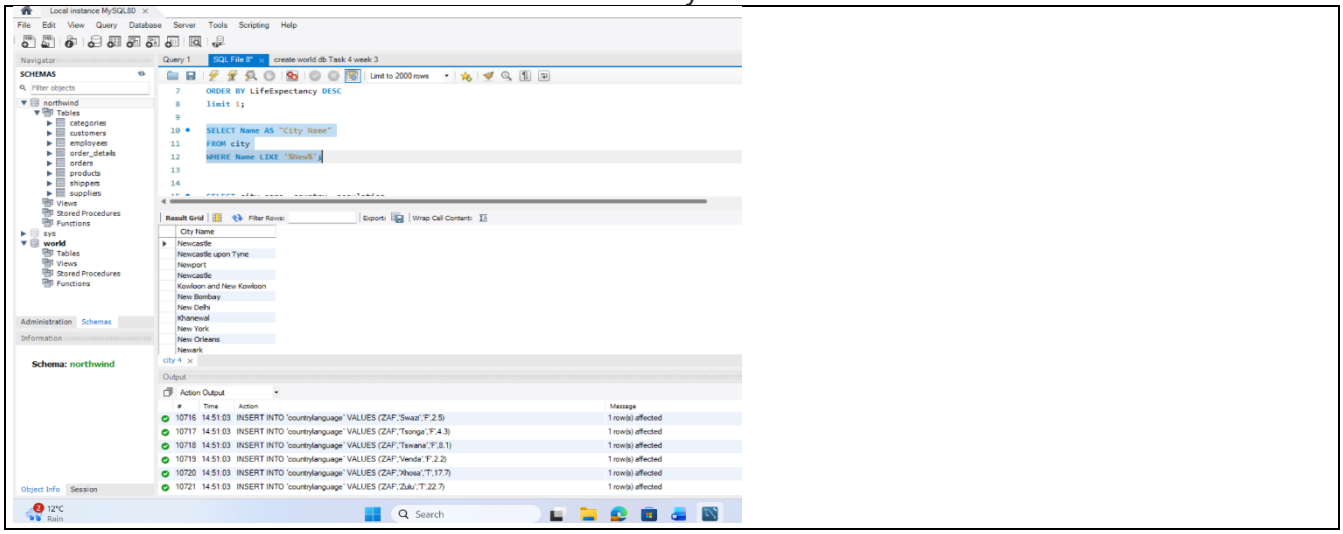
The highlighted my SQL formula for each question I didn't remember to screenshot until I completed the questions. I decided to highlight my answers for each question.



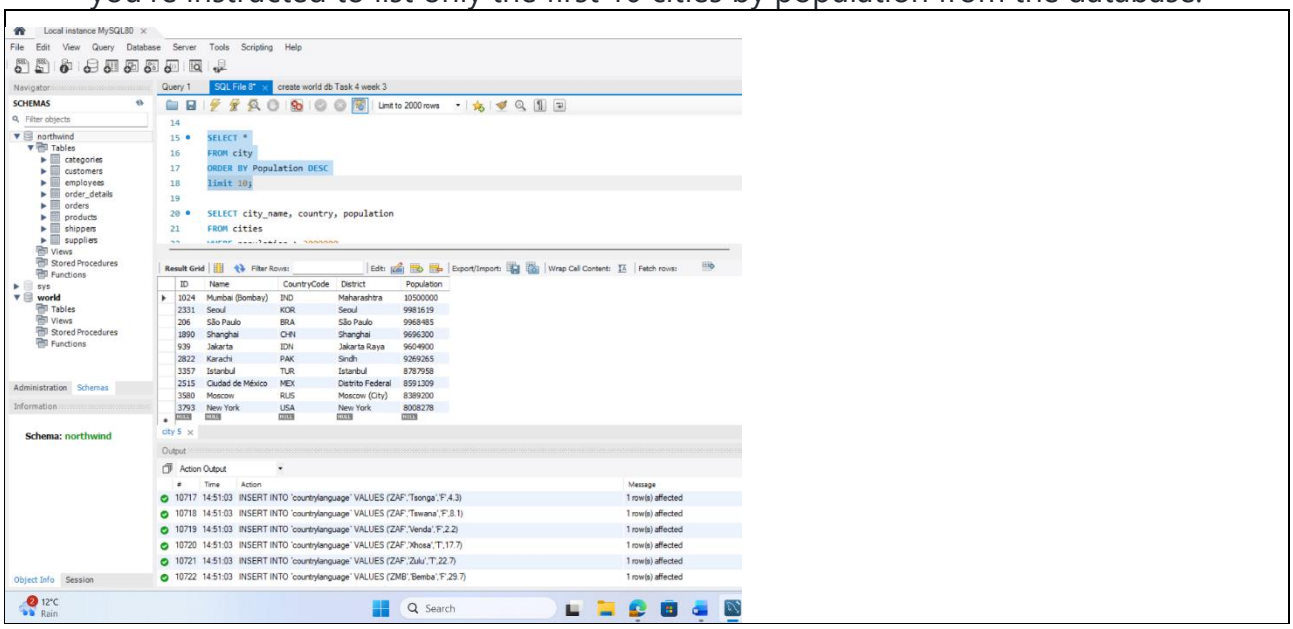
2. **Country with Highest Life Expectancy:** *Scenario:* As part of a global health initiative, you've been assigned to identify the country with the highest life expectancy. This information will be crucial for prioritising healthcare resources and interventions.



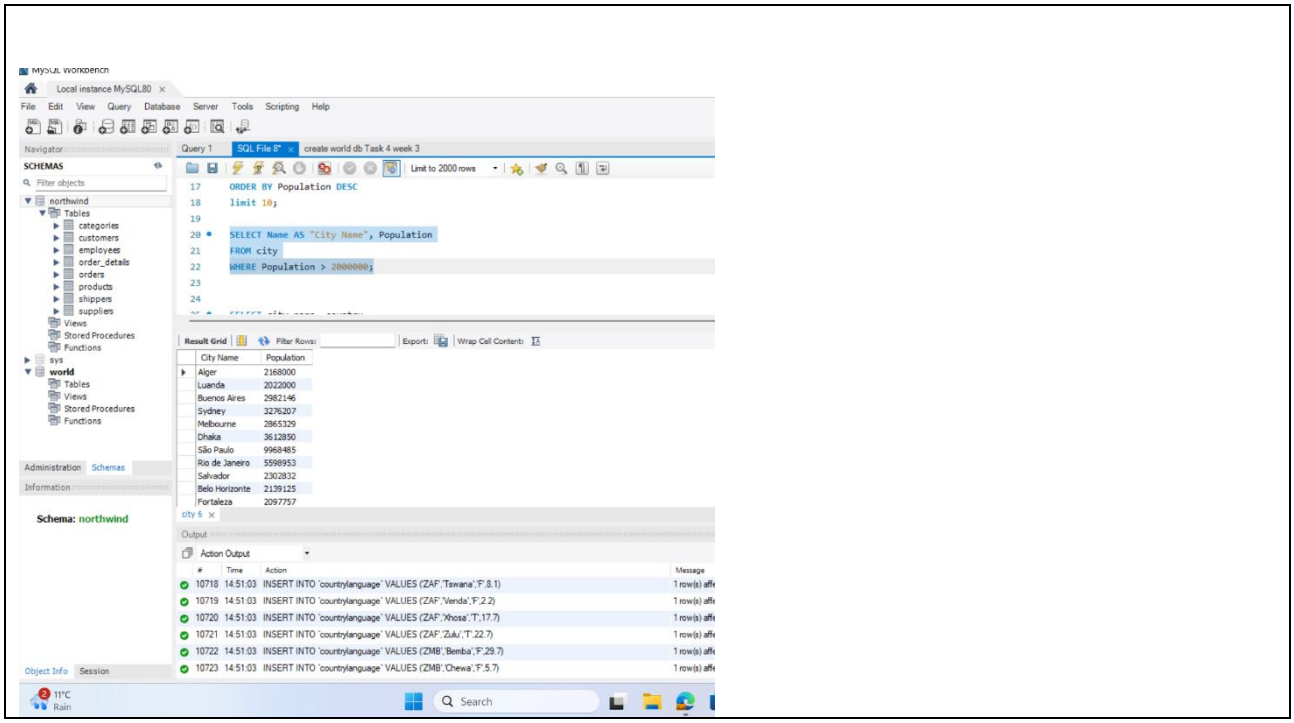
3. **"New Year Promotion: Featuring Cities with 'New :** *Scenario:* In anticipation of the upcoming New Year, your travel agency is gearing up for a special promotion featuring cities with names including the word 'New'. You're tasked with swiftly compiling a list of all cities from around the world. This curated selection will be essential in creating promotional materials and enticing travellers with exciting destinations to kick off the New Year in style.



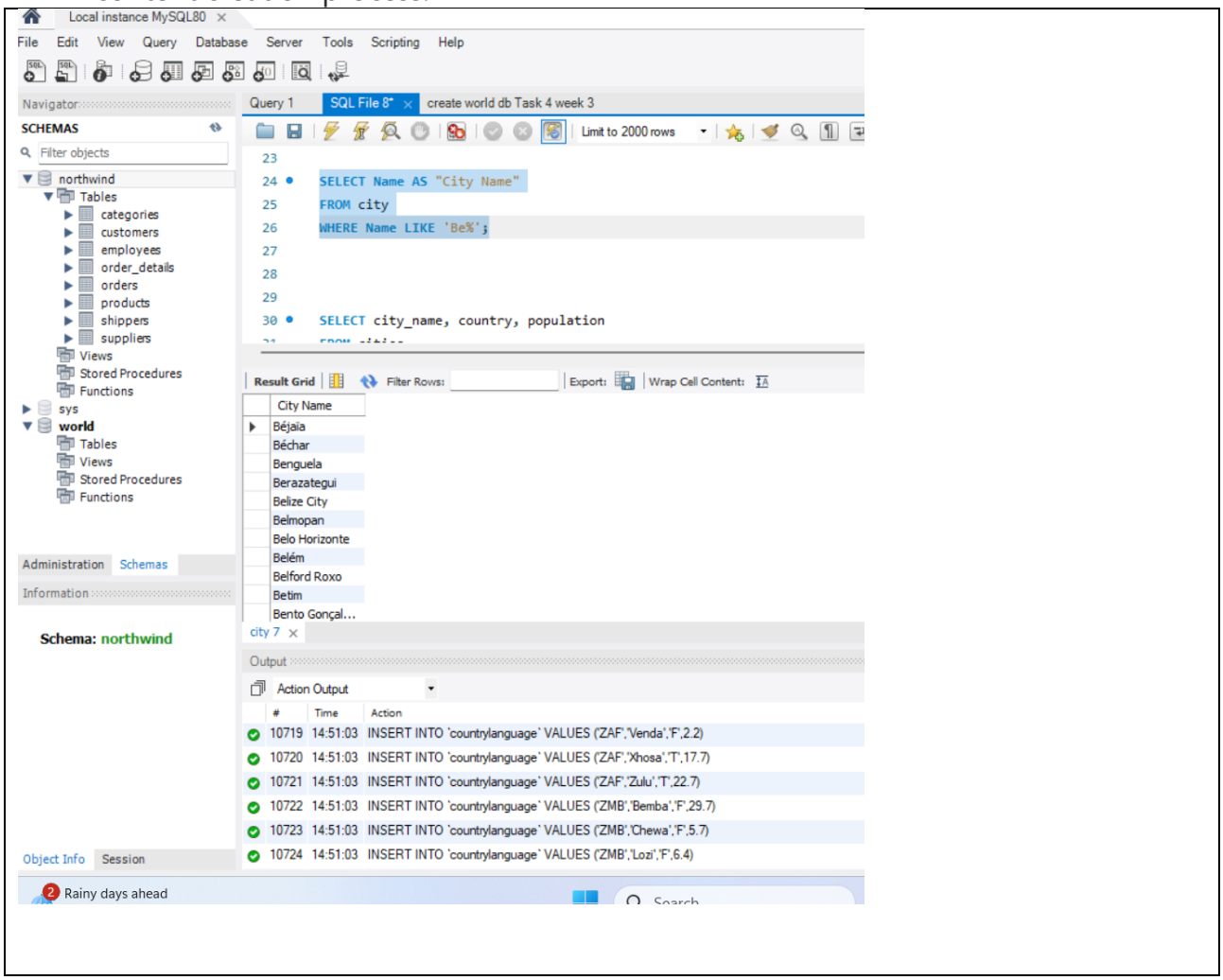
4. **Display Columns with Limit (First 10 Rows):** *Scenario:* You're tasked with providing a brief overview of the most populous cities in the world. To keep the report concise, you're instructed to list only the first 10 cities by population from the database.



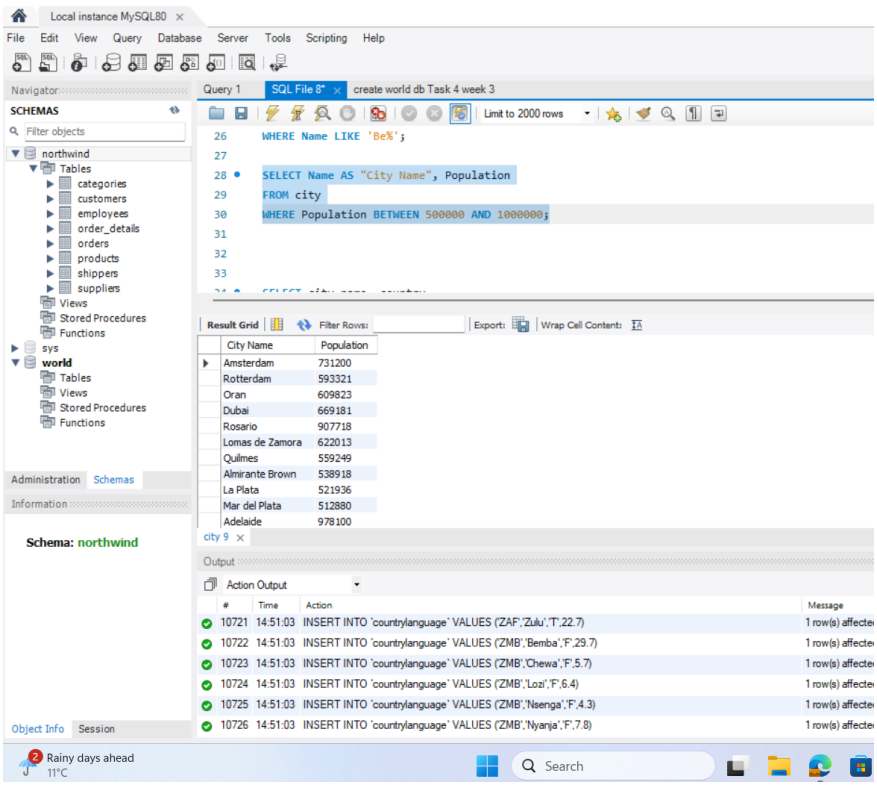
5. **Cities with Population Larger than 2,000,000:** *Scenario:* A real estate developer is interested in cities with substantial population sizes for potential investment opportunities. You're tasked with identifying cities from the database with populations exceeding 2 million to focus their research efforts.



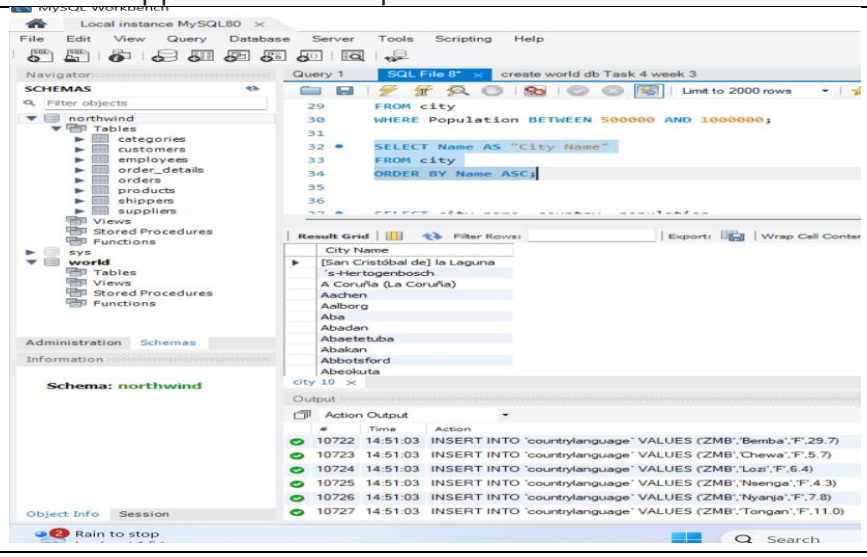
6. **Cities Beginning with 'Be' Prefix:** *Scenario:* A travel blogger is planning a series of articles featuring cities with unique names. You're tasked with compiling a list of cities from the database that start with the prefix 'Be' to assist in the blogger's content creation process.



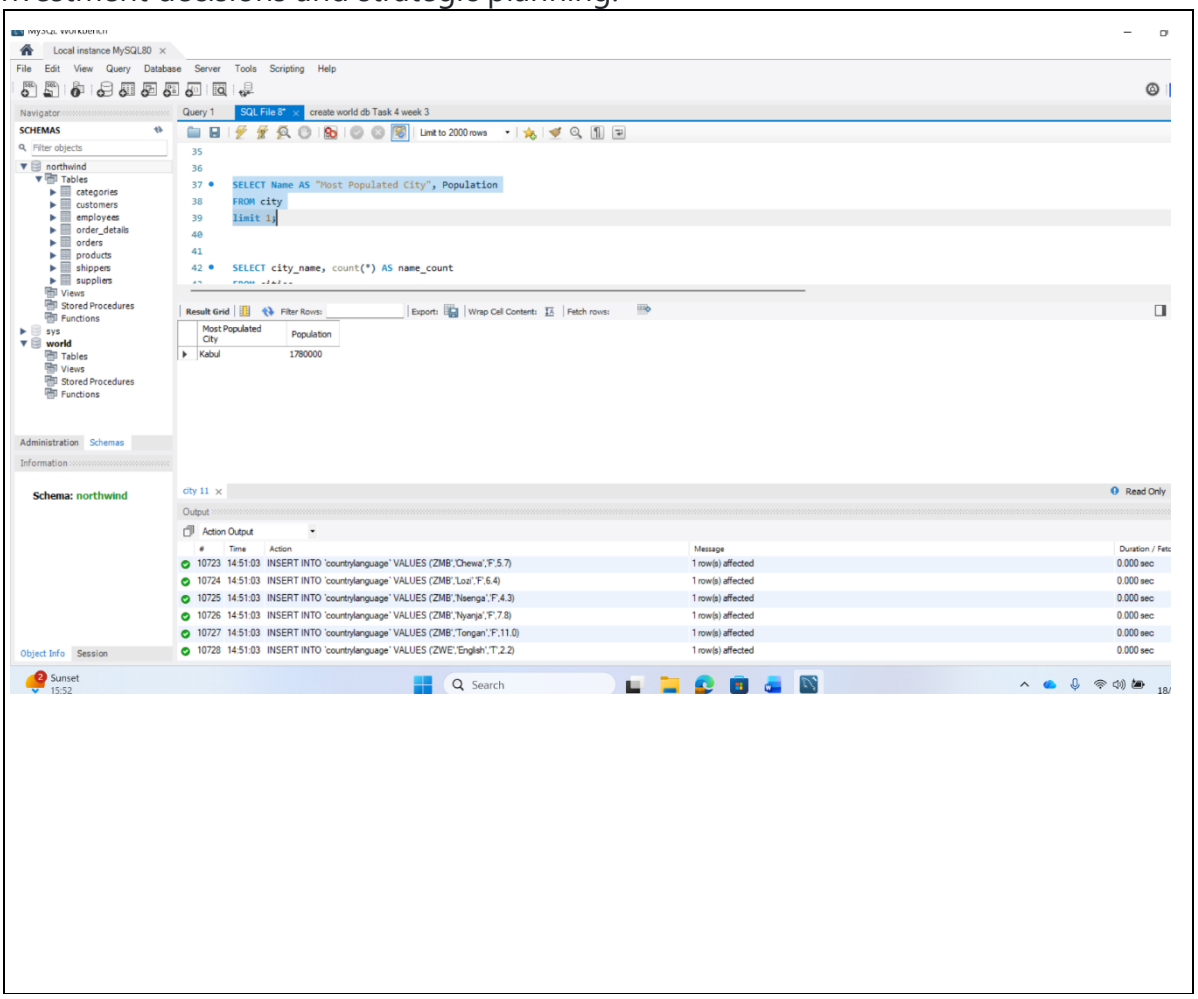
7. **Cities with Population Between 500,000-1,000,000:** *Scenario:* An urban planning committee needs to identify mid-sized cities suitable for infrastructure development projects. You're tasked with identifying cities with populations ranging between 500,000 and 1 million to inform their decision-making process.



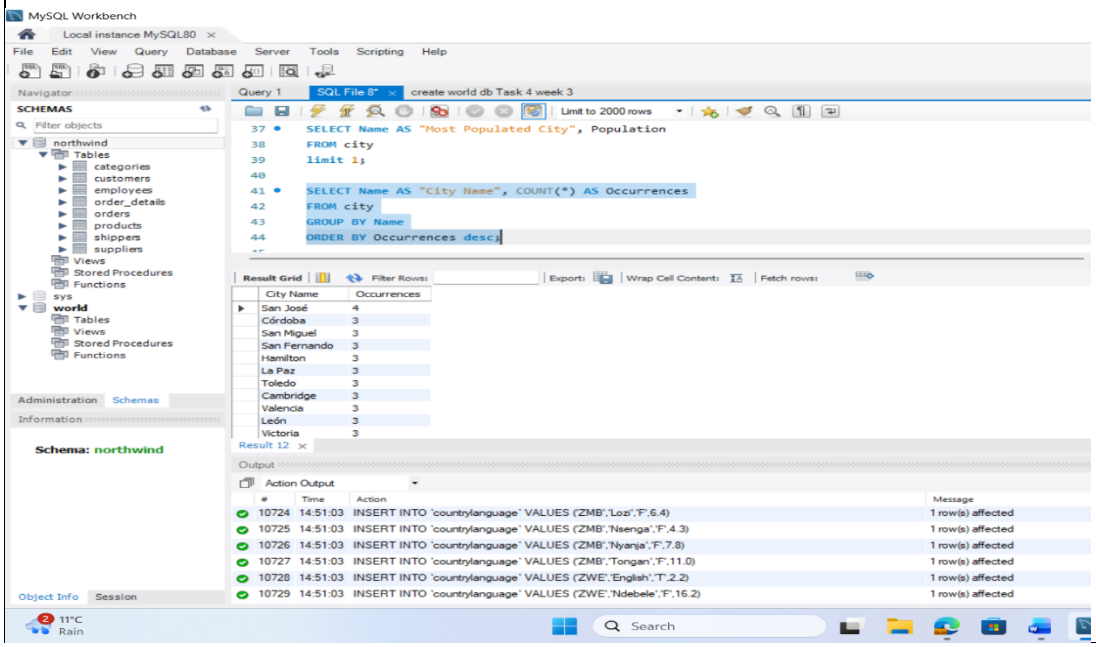
8. **Display Cities Sorted by Name in Ascending Order:** *Scenario:* A geography teacher is preparing a lesson on alphabetical order using city names. You're tasked with providing a sorted list of cities from the database in ascending order by name to support the lesson plan.



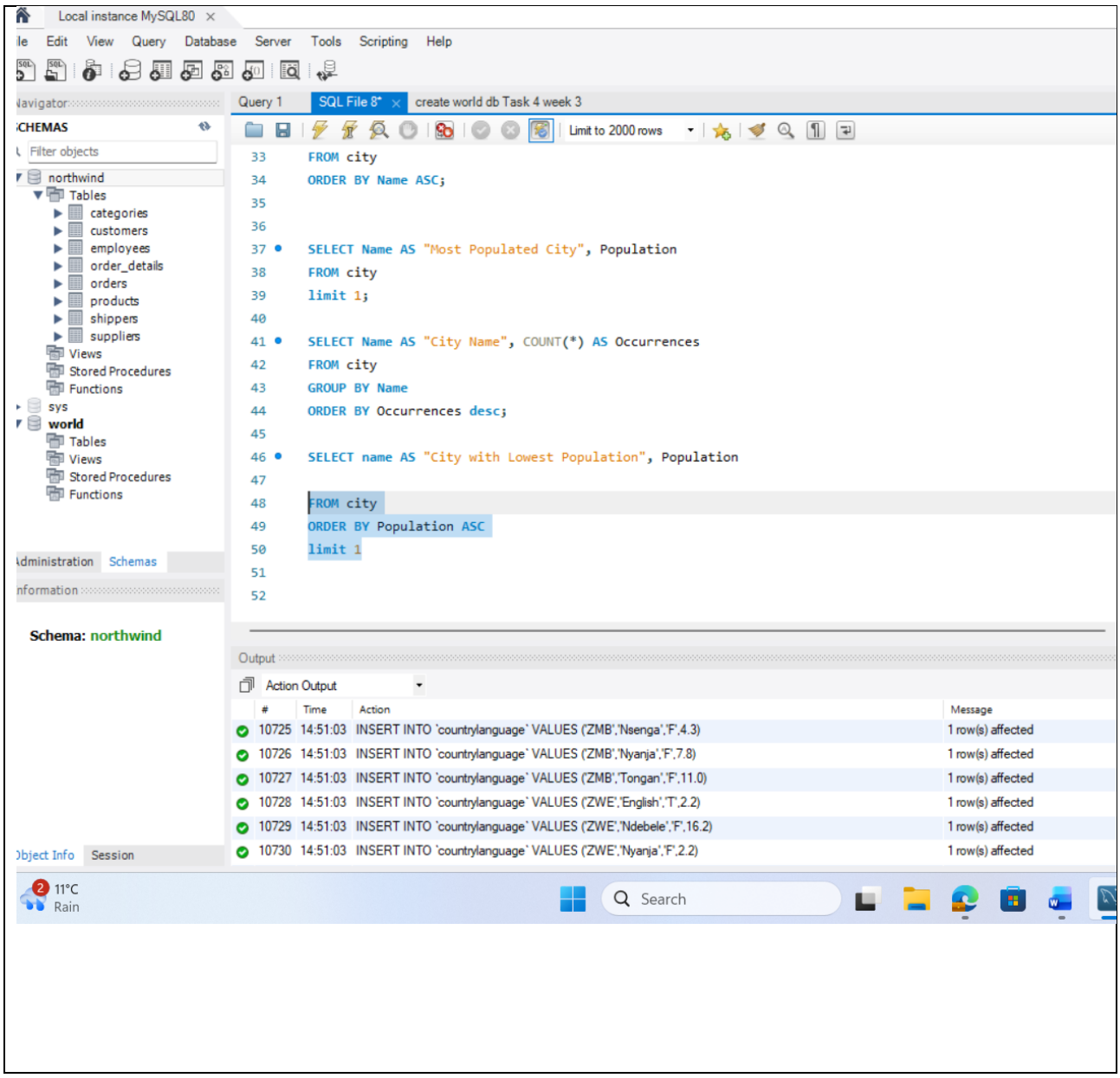
9. **Most Populated City:** *Scenario:* A real estate investment firm is interested in cities with significant population densities for potential development projects. You're tasked with identifying the most populated city from the database to guide their investment decisions and strategic planning.



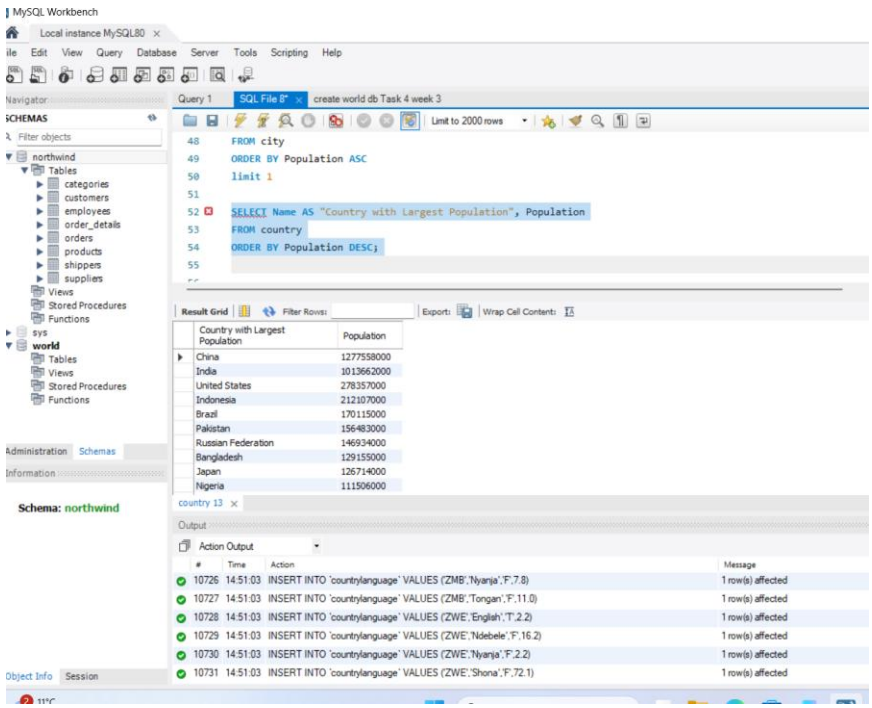
10. **City Name Frequency Analysis: Supporting Geography Education** *Scenario*: In a geography class, students are learning about the distribution of city names around the world. The teacher, in preparation for a lesson on city name frequencies, wants to provide students with a list of unique city names sorted alphabetically, along with their respective counts of occurrences in the database. You're tasked with this sorted list to support the geography teacher.



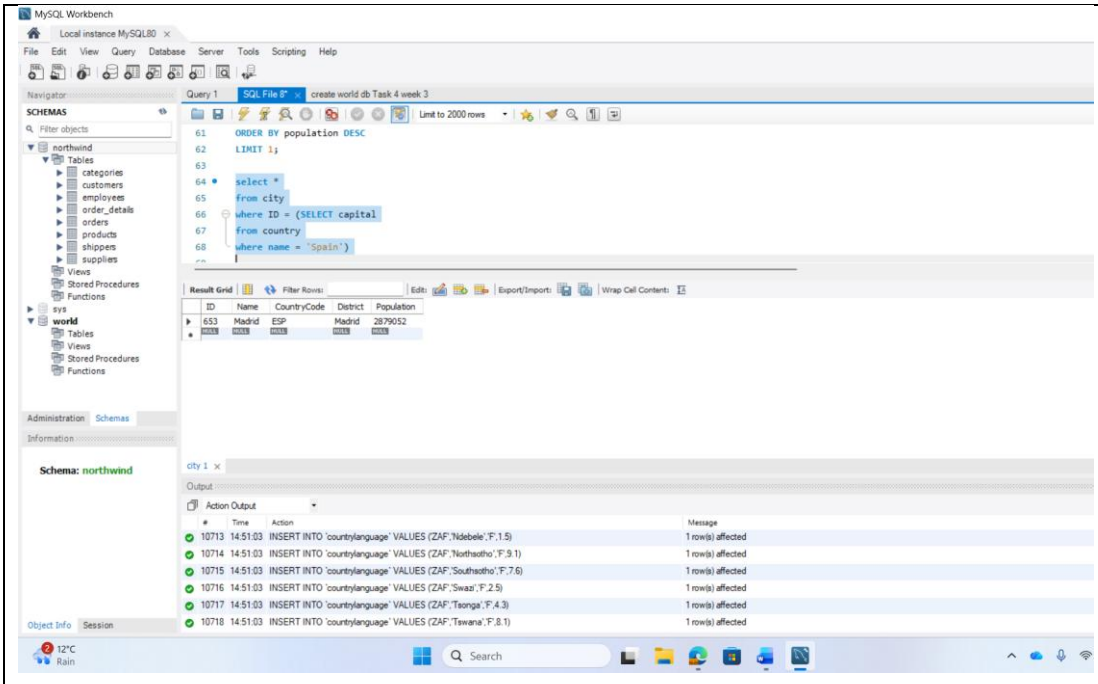
11. **City with the Lowest Population:** *Scenario:* A census bureau is conducting an analysis of urban population distribution. You're tasked with identifying the city with the lowest population from the database to provide a comprehensive overview of demographic trends.


Local instance MySQL80
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
Filter objects
northwind
Tables
categories
customers
employees
order_details
orders
products
shippers
suppliers
Views
Stored Procedures
Functions
sys
world
Tables
Views
Stored Procedures
Functions
Administration Schemas
Information
Schema: northwind
Object Info Session
Query 1 SQL File 8* create world db Task 4 week 3
Limit to 2000 rows
33 FROM city
34 ORDER BY Name ASC;
35
36
37 SELECT Name AS "Most Populated City", Population
38 FROM city
39 limit 1;
40
41 SELECT Name AS "City Name", COUNT(*) AS Occurrences
42 FROM city
43 GROUP BY Name
44 ORDER BY Occurrences desc;
45
46 SELECT name AS "City with Lowest Population", Population
47
48 FROM city
49 ORDER BY Population ASC
50 limit 1
51
52
Output
Action Output
Time Action Message
10725 14:51:03 INSERT INTO 'countrylanguage' VALUES ('ZMB','Nsenga','F',4.3) 1 row(s) affected
10726 14:51:03 INSERT INTO 'countrylanguage' VALUES ('ZMB','Nyanja','F',7.8) 1 row(s) affected
10727 14:51:03 INSERT INTO 'countrylanguage' VALUES ('ZMB','Tongan','F',11.0) 1 row(s) affected
10728 14:51:03 INSERT INTO 'countrylanguage' VALUES ('ZWE','English','T',2.2) 1 row(s) affected
10729 14:51:03 INSERT INTO 'countrylanguage' VALUES ('ZWE','Ndebele','F',16.2) 1 row(s) affected
10730 14:51:03 INSERT INTO 'countrylanguage' VALUES ('ZWE','Nyanja','F',2.2) 1 row(s) affected
11°C
Rain
Search

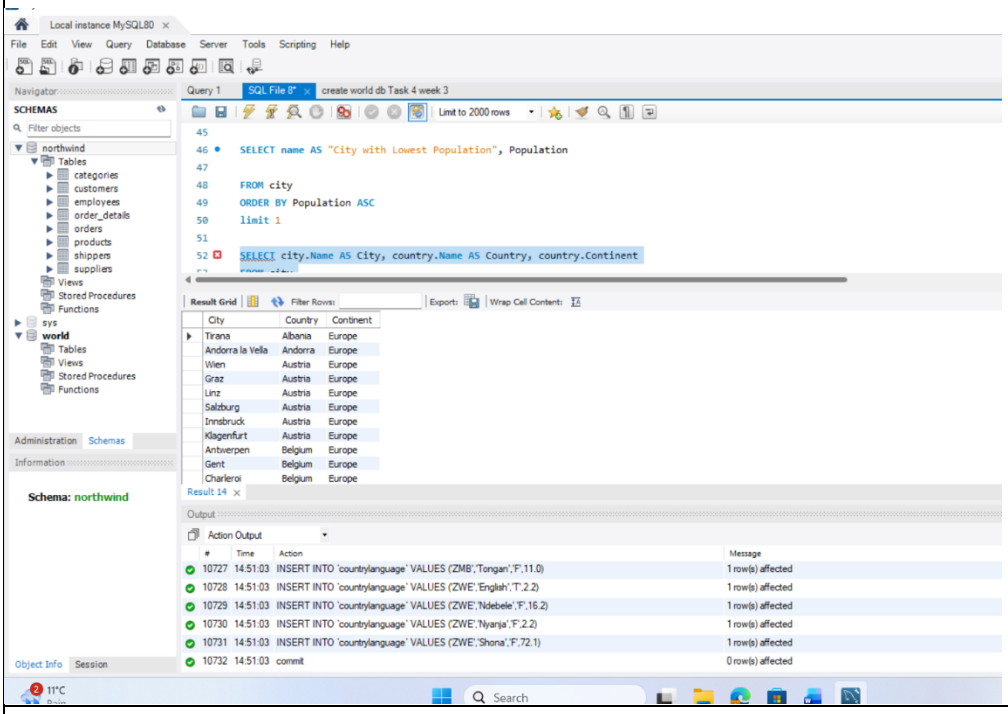
12. **Country with Largest Population:** *Scenario:* A global economic research institute requires data on countries with the largest populations for a comprehensive analysis. You're tasked with identifying the country with the highest population from the database to provide valuable insights into demographic trends.



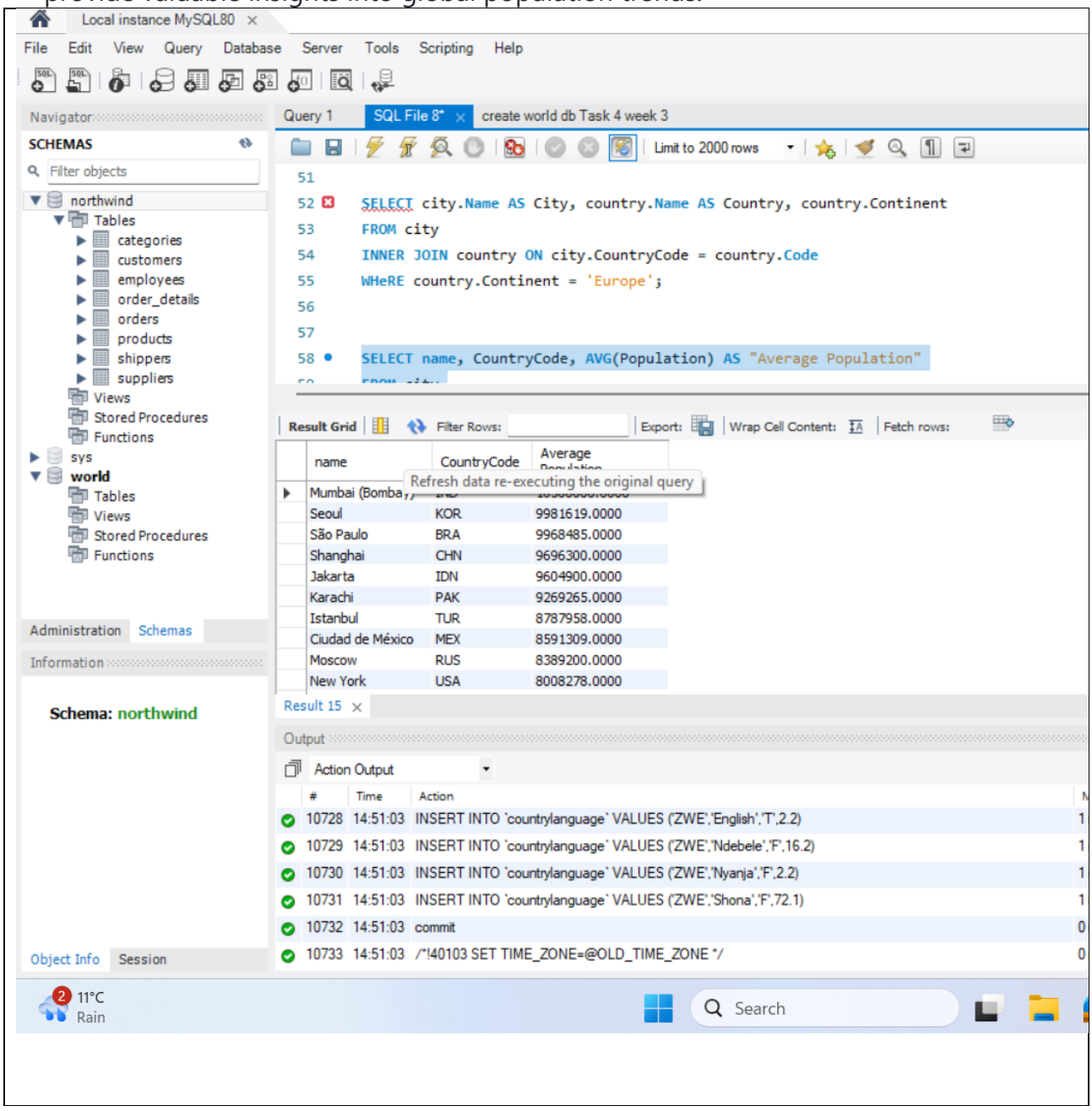
13. **Capital of Spain:** *Scenario:* A travel agency is organising tours across Europe and needs accurate information on capital cities. You're tasked with identifying the capital of Spain from the database to ensure itinerary accuracy and provide travellers with essential destination information.



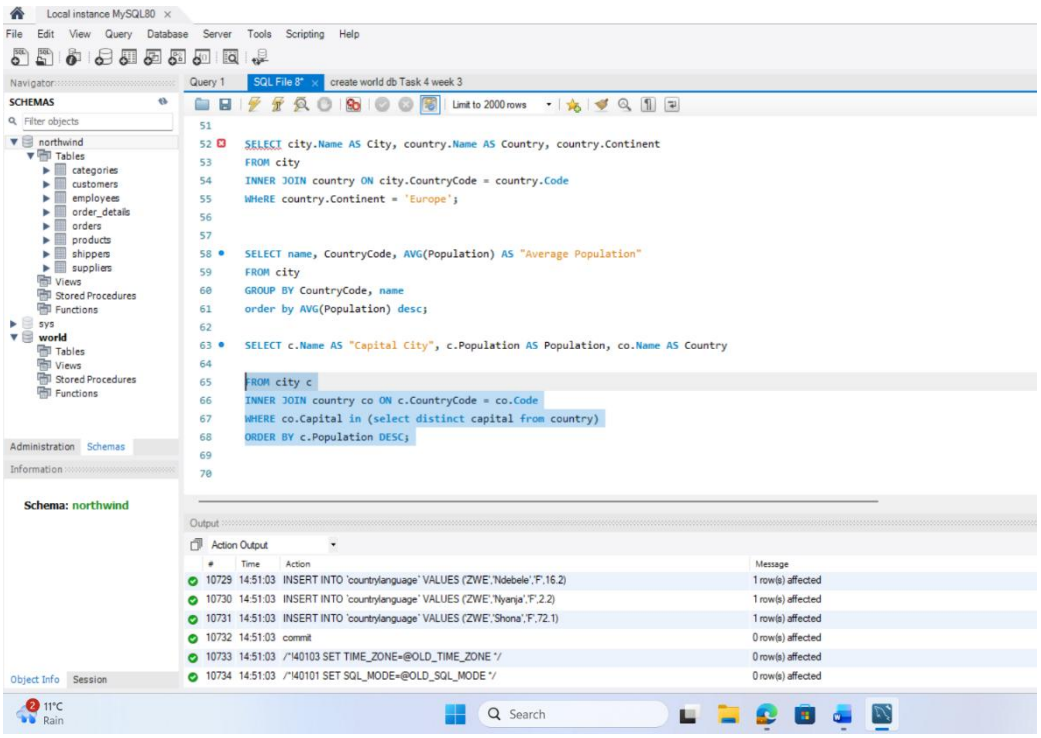
14. **Cities in Europe:** *Scenario:* A European cultural exchange program is seeking to connect students with cities across the continent. You're tasked with compiling a list of cities located in Europe from the database to facilitate program planning and student engagement.



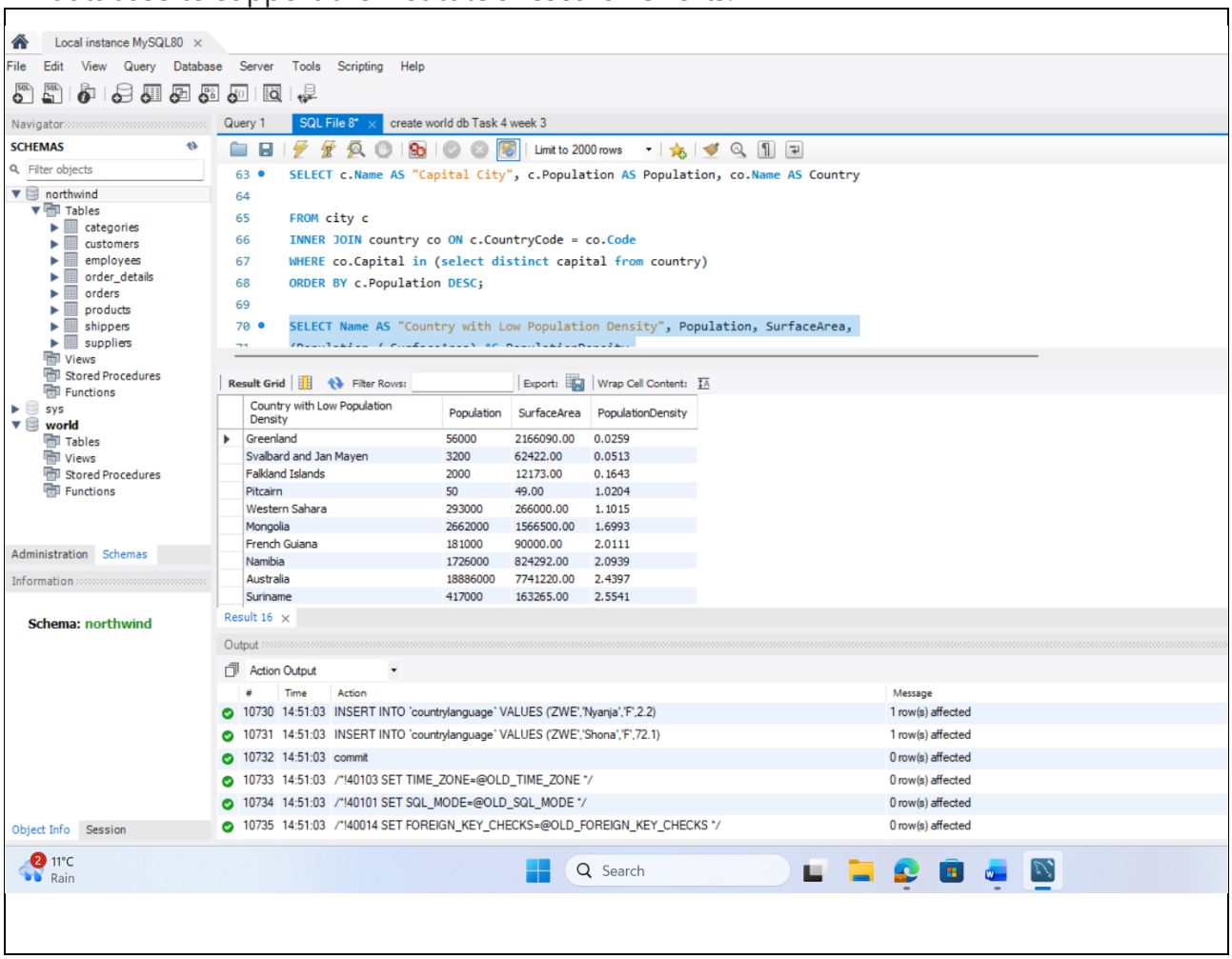
15. **Average Population by Country:** *Scenario:* A demographic research team is conducting a comparative analysis of population distributions across countries. You're tasked with calculating the average population for each country from the database to provide valuable insights into global population trends.



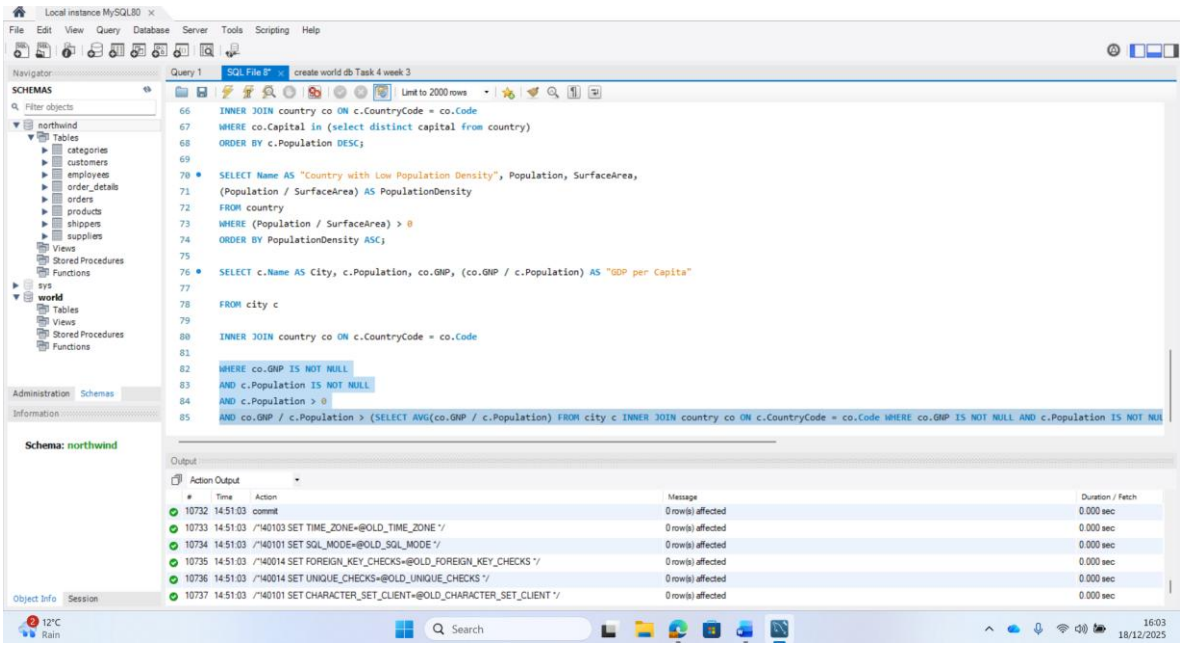
16. **Capital Cities Population Comparison:** *Scenario:* A statistical analysis firm is examining population distributions between capital cities worldwide. You're tasked with comparing the populations of capital cities from different countries to identify trends and patterns in urban demographics.



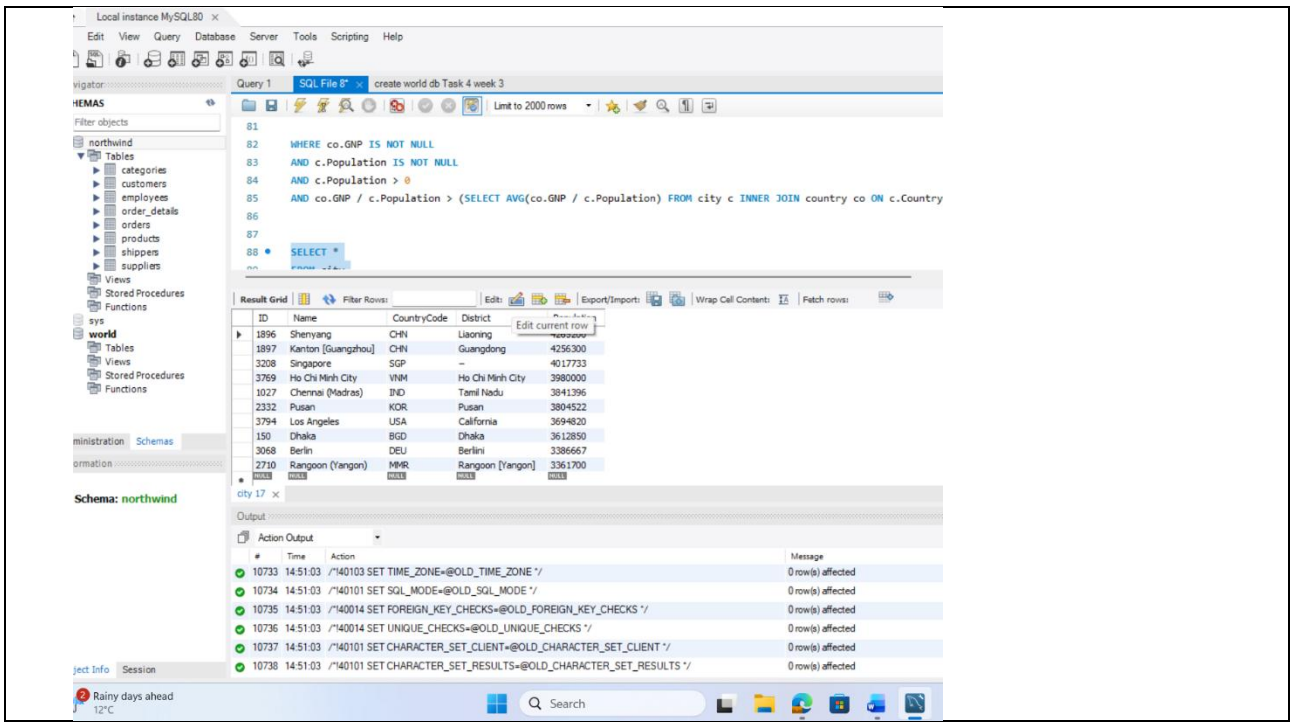
17. **Countries with Low Population Density:** *Scenario:* An agricultural research institute is studying countries with low population densities for potential agricultural development projects. You're tasked with identifying countries with sparse populations from the database to support the institute's research efforts.



18. **Cities with High GDP per Capita:** *Scenario:* An economic consulting firm is analysing cities with high GDP per capita for investment opportunities. You're tasked with identifying cities with above-average GDP per capita from the database to assist the firm in identifying potential investment destinations.



19. **Display Columns with Limit (Rows 31-40):** *Scenario:* A market research firm requires detailed information on cities beyond the top rankings for a comprehensive analysis. You're tasked with providing data on cities ranked between 31st and 40th by population to ensure a thorough understanding of urban demographics.



Course Notes

It is recommended to take notes from the course, use the space below to do so, or use the revision guide shared with the class:

Additional Resources

1) SQLBolt - Learn SQL - Introduction to SQL : Welcome to SQLBolt, a series of interactive lessons and exercises designed to help you quickly learn SQL right in your browser.
2) MySQL Basics Tutorial Series - YouTube: MySQL Basics Tutorial Series
3) Hackerrank Challenges: How to use hackerrank challanges.odt
4) Sakila DB Exercises:
    a. Use this file to create the schema: create sakila db.sql
    b. Access exercises here: sakila db exercises combined.sql

You can watch the video on how to open sql file:How to create a schema in workbench.webm

We have included a range of additional links to further resources and information that you may find useful, these can be found within your revision guide.

END OF WORKBOOK

Please check through your work thoroughly before submitting and update the table of contents if required.

Please send your completed work booklet to your trainer.