

Data Technician

Name:

Course Date:

Table of contents

Day 1: Task 1	3
Day 1: Task 2	5
Day 3: Task 1	6
Day 4: Task 1: Written.....	8
Day 4: Task 2: SQL Practical.....	13
Course Notes.....	30
Additional Information.....	31



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 unique identifier for each record in a table. For example student ID in a student table.
How does this differ from a secondary key?	A secondary key is another field used for searching or sorting, but may not be unique. For example last name.
How are primary and foreign keys related?	<p>A foreign key is a field in one table that links to the primary key in another key in another table. This creates a relationship between the two tables.</p> <p>For example: student table- student ID (primary key) Courses table – course ID (primary key) Enrolment table- student ID (foreign key) + course ID (Foreign Key)</p>
Provide a real-world example of a one-to-one relationship	Person – passport: each person has one passport, and each passport belongs to one person.
Provide a real-world example of a one-to-many relationship	Teacher-student: one teacher can teach many students but each student has only one main teacher.
Provide a real-world example of a many-to-many relationship	Students-courses: students can enroll in many courses, and each course can have many students.



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 (SQL) stores data in structured tables with predefined schemas, enforces relationships through keys, and uses SQL for complex queries and transactions making it ideal for consistent, structured data like financial records. A non-relational database (NoSQL) stores data in flexible formats such as documents, key-value pairs, or graphs, often without a fixed schema, and is designed for scalability, fast access, and handling unstructured or rapidly changing data commonly used in social media, IoT, or real-time analytics. In short, relational databases emphasize structure and consistency, while non-relational databases prioritize flexibility and scalability.
What type of data would benefit off the non-relational model? Why?	Data that benefits from a non-relational model is typically unstructured, semi-structured, or fast-changing, such as social media posts, product catalogs with varying attributes, IoT sensor streams, real-time analytics, multimedia content, and complex relationships like social networks or recommendations. These scenarios need flexible schemas and horizontal scalability, making non-relational databases ideal when data doesn't fit neatly into rows and columns or must handle massive, rapidly growing workloads.



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	<p>is when a table is joined with itself, treating it as if it were two separate tables (using aliases). It's useful for comparing rows within the same table.</p> <p>Example: In an Employees table, a self join can link each employee to their manager by joining the table on EmployeeID and ManagerID.</p>
Right join	<p>Returns all rows from the right table, plus matching rows from the left.</p> <p><i>Example:</i> All orders, even if the customer record is missing.</p>
Full join	<p>Returns all rows from both tables, with nulls where there's no match.</p> <p><i>Example:</i> All customers and all orders, showing unmatched records from either side.</p>
Inner join	<p>Returns rows that match in both tables.</p> <p><i>Example:</i> Customers who have placed an order (only those appearing in both the Customers and Orders tables).</p>
Cross join	<p>Returns every combination of rows from both tables.</p> <p><i>Example:</i> Pairing every product with every promotion to test possible deals.</p>



Left join

LEFT JOIN: Returns all rows from the left table, plus matching rows from the right (null if no match).
Example: All customers, including those who haven't placed an order.



Day 4: Task 1: Written

In your groups, discuss and complete the below activity. You can either nominate one writer or split the elements between you. Everyone however must have the completed work below:

Imagine you have been hired by a small retail business that wants to streamline its operations by creating a new database system. This database will be used to manage inventory, sales, and customer information. The business is a small corner shop that sells a range of groceries and domestic products. It might help to picture your local convenience store and think of what they sell. They also have a loyalty program, which you will need to consider when deciding what tables to create.

Write a 500-word essay explaining the steps you would take to set up and create this database. Your essay should cover the following points:

1. **Understanding the Business Requirements:**
 - a. What kind of data will the database need to store?
 - b. Who will be the users of the database, and what will they need to accomplish?
2. **Designing the Database Schema:**
 - a. How would you structure the database tables to efficiently store inventory, sales, and customer information?
 - b. What relationships between tables are necessary (e.g., how sales relate to inventory and customers)?
3. **Implementing the Database:**
 - a. What SQL commands would you use to create the database and its tables?
 - b. Provide examples of SQL statements for creating tables and defining relationships between them.
4. **Populating the Database:**
 - a. How would you input initial data into the database? Give examples of SQL INSERT statements.
5. **Maintaining the Database:**
 - a. What measures would you take to ensure the database remains accurate and up to date?
 - b. How would you handle backups and data security?

Your essay should include specific examples of SQL commands and explain why each step is necessary for creating a functional and efficient database for the retail business.



When setting up a database for a small corner shop, the first thing I'd do is figure out what the business actually needs from it. The shop sells groceries and little household items, so the database has to keep track of things like product names, categories, prices, and how much stock is left. It also needs to hold customer info, like names, emails, and loyalty points, since the shop has a reward system. On top of that, it should store sales transactions things like what was bought, when, how much it cost, and which customer bought it. The main people using the database would be the shop workers at the checkout, who need to quickly record sales, and the manager, who will want to check reports like which items are popular, which ones are running low, and how customers are doing with loyalty points.

After that, the next step is designing the database schema, which basically means planning out the tables. For example, there would be a Products table with product ID, name, category, price, and stock quantity. A Customers table would keep customer details like their ID, name, email, and loyalty points. Then you'd have a Sales table to store sales records, and a SaleItems table to connect each product to a specific sale. The relationships are important: one customer can make many sales, and each sale can include many products. Having it set up this way keeps everything organised and makes it easier to answer questions later, like "how many times has this customer bought bread?" or "which product is selling the most?"

Next comes actually creating the database using SQL commands. First, I'd make the database itself using something like `CREATE DATABASE RetailShop;`. Then I'd create the tables with commands like `CREATE TABLE Products` and so on. For example, the Products table would have columns for product ID, name, category, price, and stock. The Customers table would have customer ID, name, email, phone number, and loyalty points. In the Sales table, I'd include sale ID, sale date, customer ID, and total amount, while also linking the customer ID to the Customers table with a foreign key.

The SaleItems table would connect sales and products by including sale ID, product ID, and quantity. These relationships are what keep all the data linked together so it makes sense.

After setting it up, the database needs data put into it. This can be done with SQL `INSERT` statements. For example:

```
INSERT INTO Products (ProductName, Category, Price, StockQuantity)
VALUES ('Milk', 'Dairy', 1.50, 10);
```

For customers:

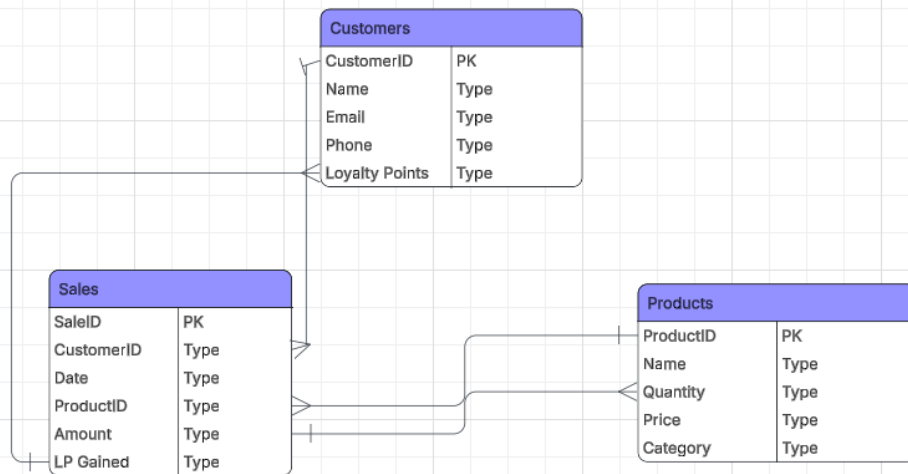
```
INSERT INTO Customers (Name, Email, LoyaltyPoints) VALUES ('Jane Doe',
'jane.doe@example.com', 50);
```

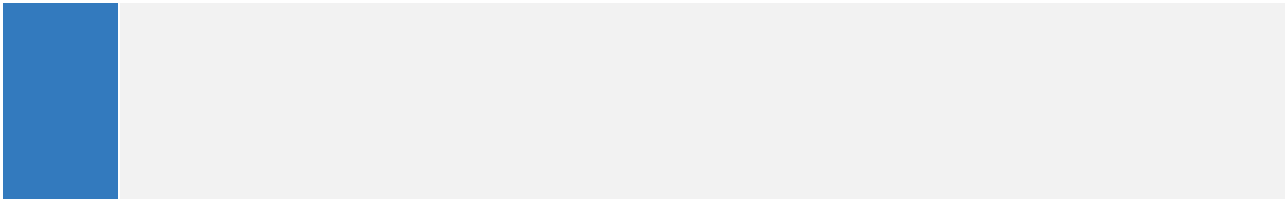
Sales can be added in the same way. If there's already a lot of data in Excel or CSV files, it can be imported straight into the database, which is much faster than typing everything out manually.

Finally, once the database is running, it has to be looked after. This means doing regular backups in case something goes wrong, encrypting them so the data is secure, and testing that the backups actually work.

Maintenance also includes things like cleaning up duplicate data, updating the software for security, and checking performance with indexing. If problems are caught early, the database will keep running smoothly.

Overall, the process goes step by step: figure out what data is needed, design the tables and relationships, create them with SQL, add the data, and then keep the database maintained. Doing all this makes sure the shop can manage stock, track sales, and run the loyalty program without any hassle.





Day 4: Task 2: SQL Practical

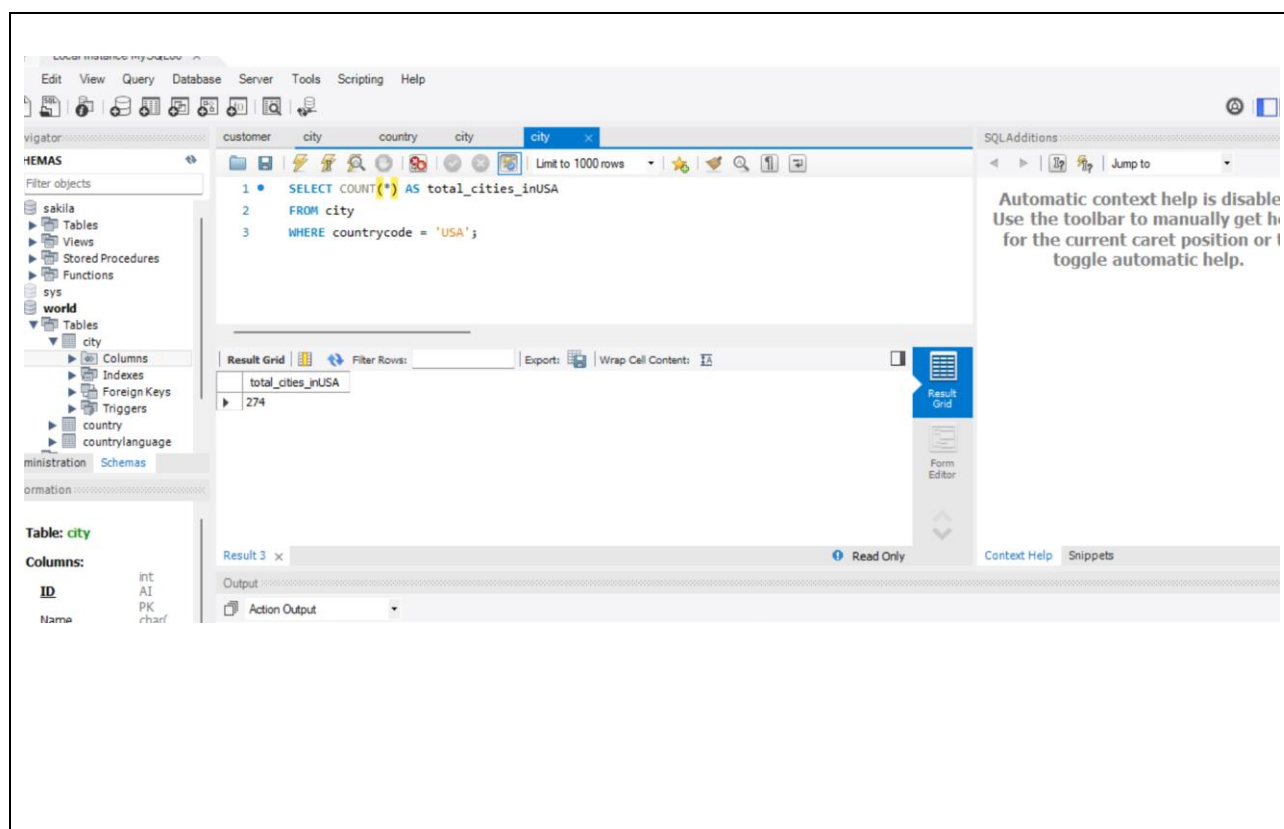
In your groups, work together to answer the below questions. It may be of benefit if one of you shares your screen with the group and as a team answer / take screen shots from there.

Setting up the database:

1. Download world_db(1)
2. Follow each step to create your database

For each question I would like to see both the syntax used and the output.

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 screenshot displays the SQL Enterprise Manager interface. On the left, the 'Server Enterprise Explorer' shows a tree view of database objects, including 'Tables', 'Views', 'Stored Procedures', 'Functions', and 'Triggers'. The 'city' table is selected under the 'world' database. The main window shows the 'Query Editor' with the following SQL query:

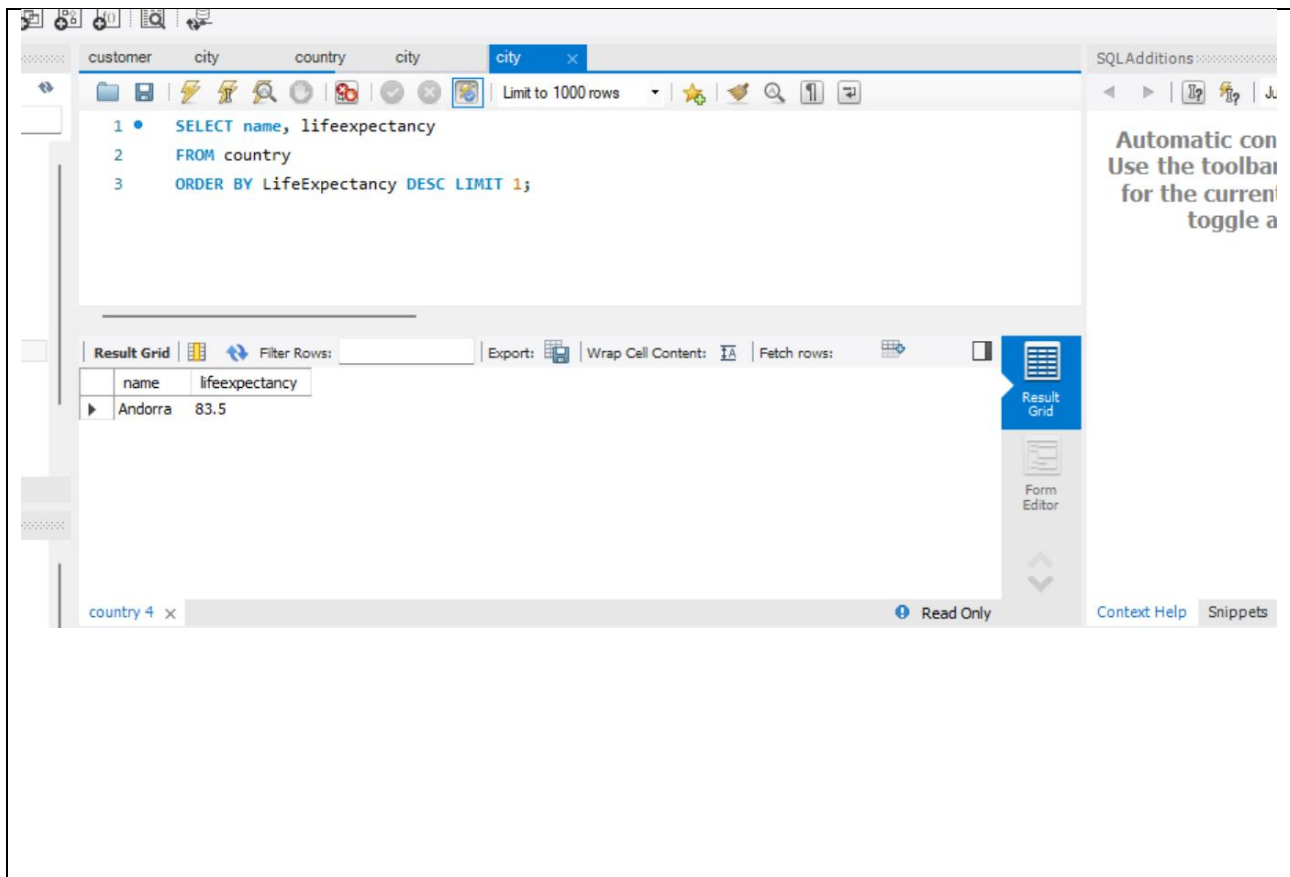
```
1 SELECT COUNT(*) AS total_cities_inUSA
2 FROM city
3 WHERE countrycode = 'USA';
```

The 'Results' pane shows the output of the query:

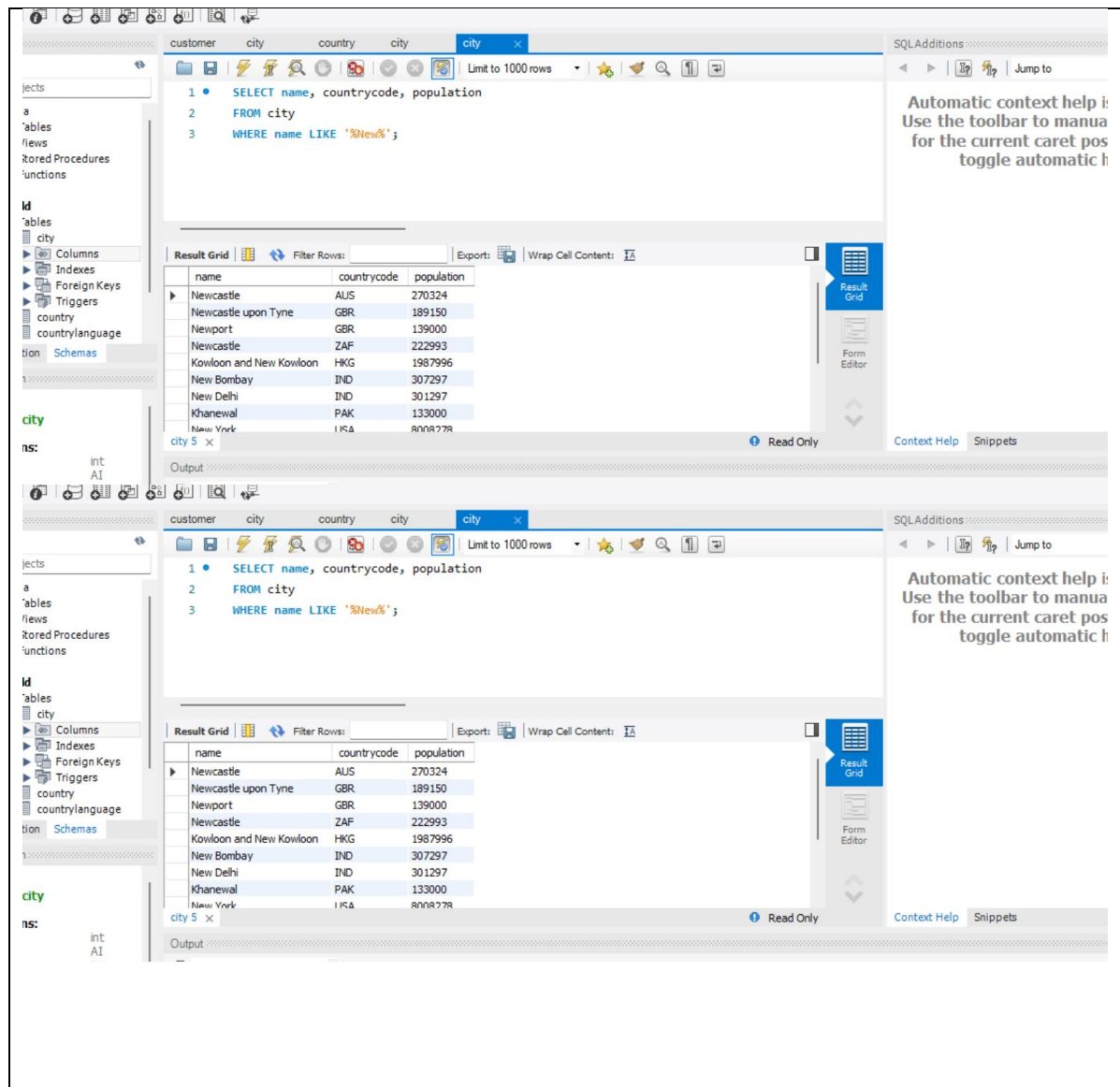
total_cities_inUSA
274

On the right, the 'SQL Additions' pane displays a message: 'Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.'

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.



- 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.

SQL query:

```
1 SELECT name, countrycode, population
2 FROM city
3 ORDER BY population DESC LIMIT 10;
```

Results:

name	countrycode	population
Mumbai (Bombay)	IND	10500000
Seoul	KOR	9981619
São Paulo	BRA	9968485
Shanghai	CHN	9696300
Jakarta	IDN	9604900
Karachi	PAK	9269265
Istanbul	TUR	8787958
Ciudad de México	MEX	8591309
Moscow	RUS	8380700

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.

SQL query:

```
1 SELECT name, countrycode, population
2 FROM city
3 WHERE population > 2000000;
```

Results:

name	countrycode	population
Alger	DZA	2168000
Luanda	AGO	2022000
Buenos Aires	ARG	2982146
Sydney	AUS	3276207
Melbourne	AUS	2865329
Dhaka	BGD	3612850
São Paulo	BRA	9968485
Rio de Janeiro	BRA	5598953
Salvador	BRA	2707877

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.

The screenshot shows a database application window with a tab labeled 'city'. The SQL editor contains the following query:

```
1 SELECT name, countrycode, population
2 FROM city
3 WHERE name LIKE 'B%';
```

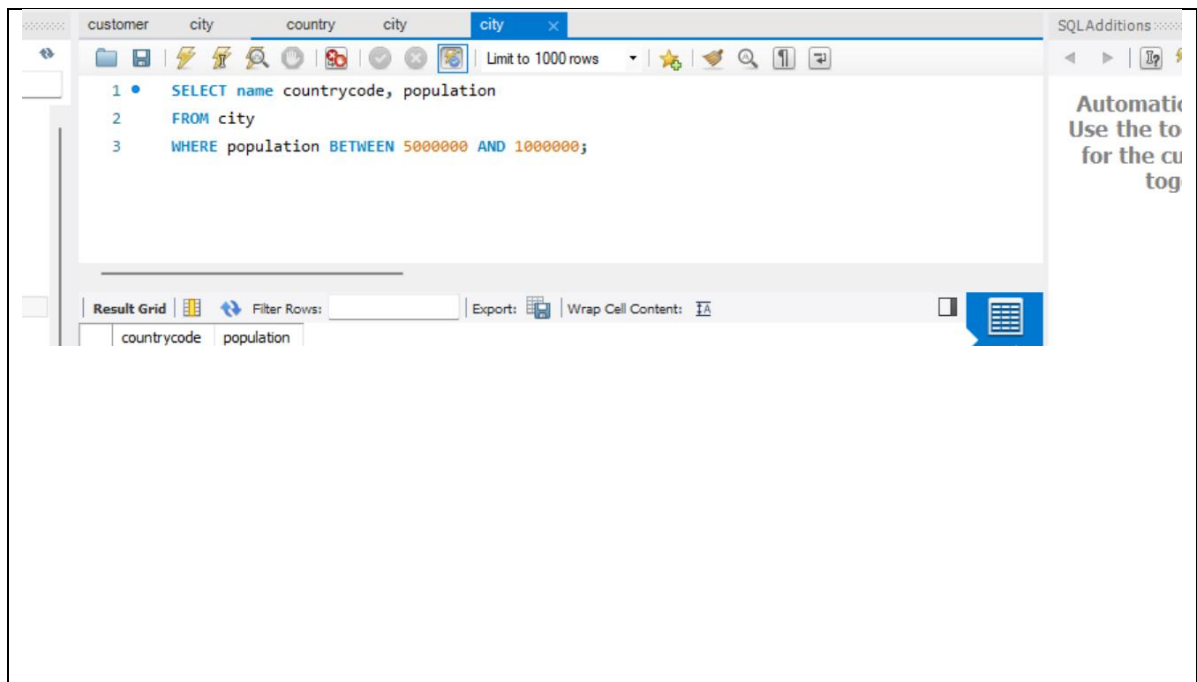
The 'Result Grid' displays the following data:

name	countrycode	population
Breda	NLD	160398
Batna	DZA	183377
Biskra	DZA	128281
Blida (el-Boulaida)	DZA	127284
Béjaia	DZA	117162
Béchar	DZA	107311
Benguela	AGO	128300
Buenos Aires	ARG	2982146
Brazzaville	AGO	276916

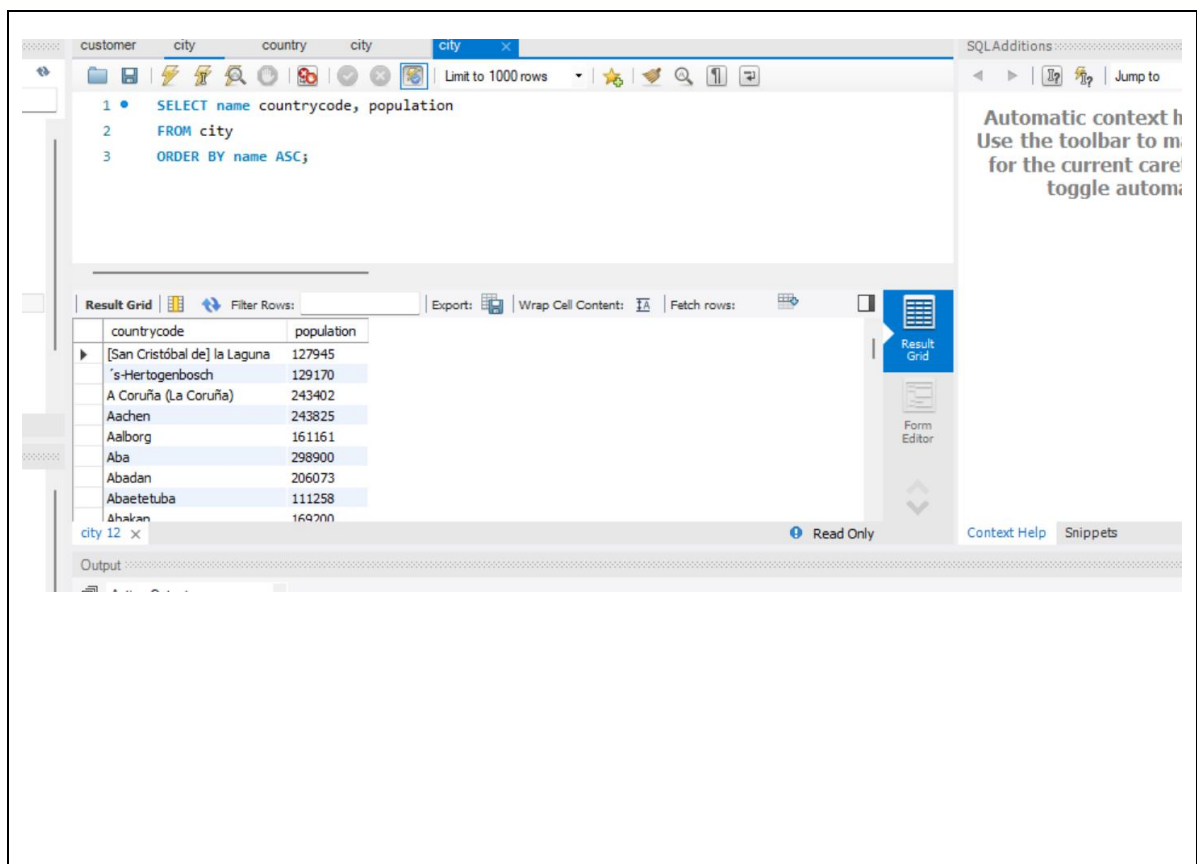
The 'Output' pane at the bottom shows the following message:

```
12 14:30:52 SELECT name, countrycode, population FROM city ORDER BY population DESC ... 10 row(s) returned
```

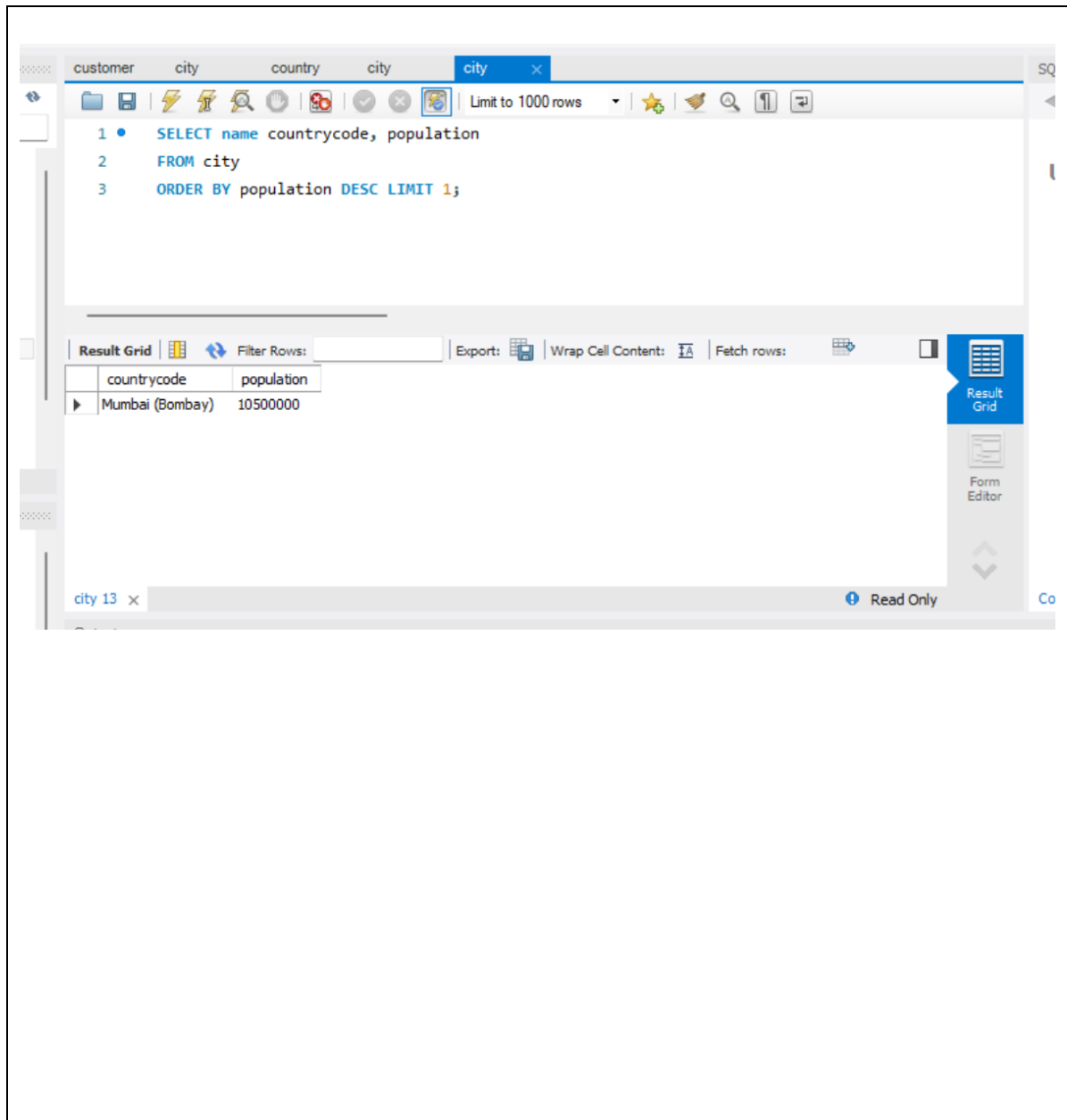
7. **Cities with Population Between 500,000-1,000,000:** *Scenario:* An urban planning committee needs to identify mid-sized cities suitable for infrastructure development projects. You're tasked with identifying cities with populations ranging between 500,000 and 1 million to inform their decision-making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



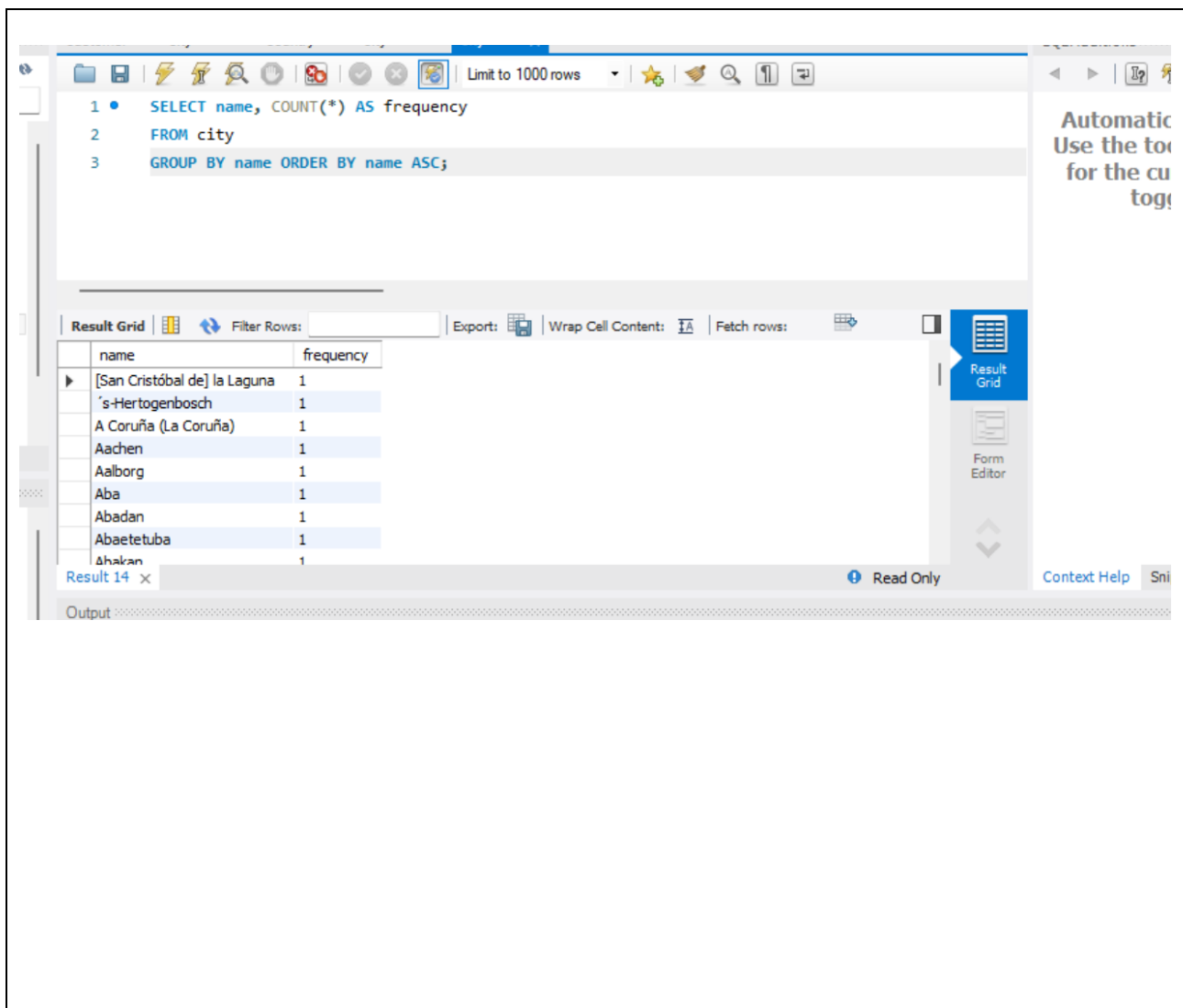
8. **Display Cities Sorted by Name in Ascending Order:** *Scenario:* A geography teacher is preparing a lesson on alphabetical order using city names. You're tasked with providing a sorted list of cities from the database in ascending order by name to support the lesson plan.



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.

customer city country city city

Limit to 1000 rows

```
1 • SELECT name, countrycode, population
2 FROM city
3 ORDER BY population ASC LIMIT 1;
```

Result Grid

	name	countrycode	population
▶	Adamstown	PCN	42

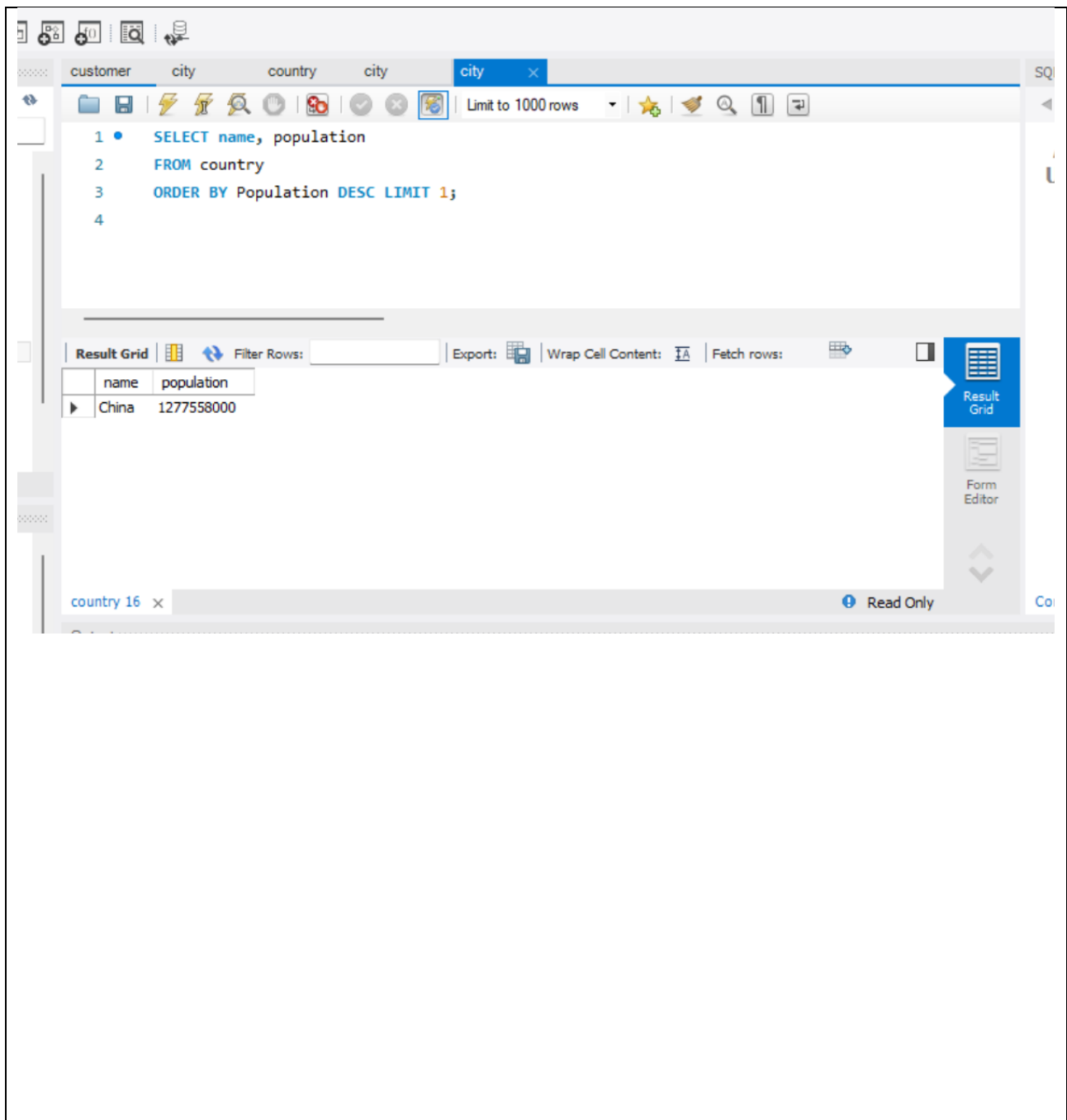
city 15 x

Read Only

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.

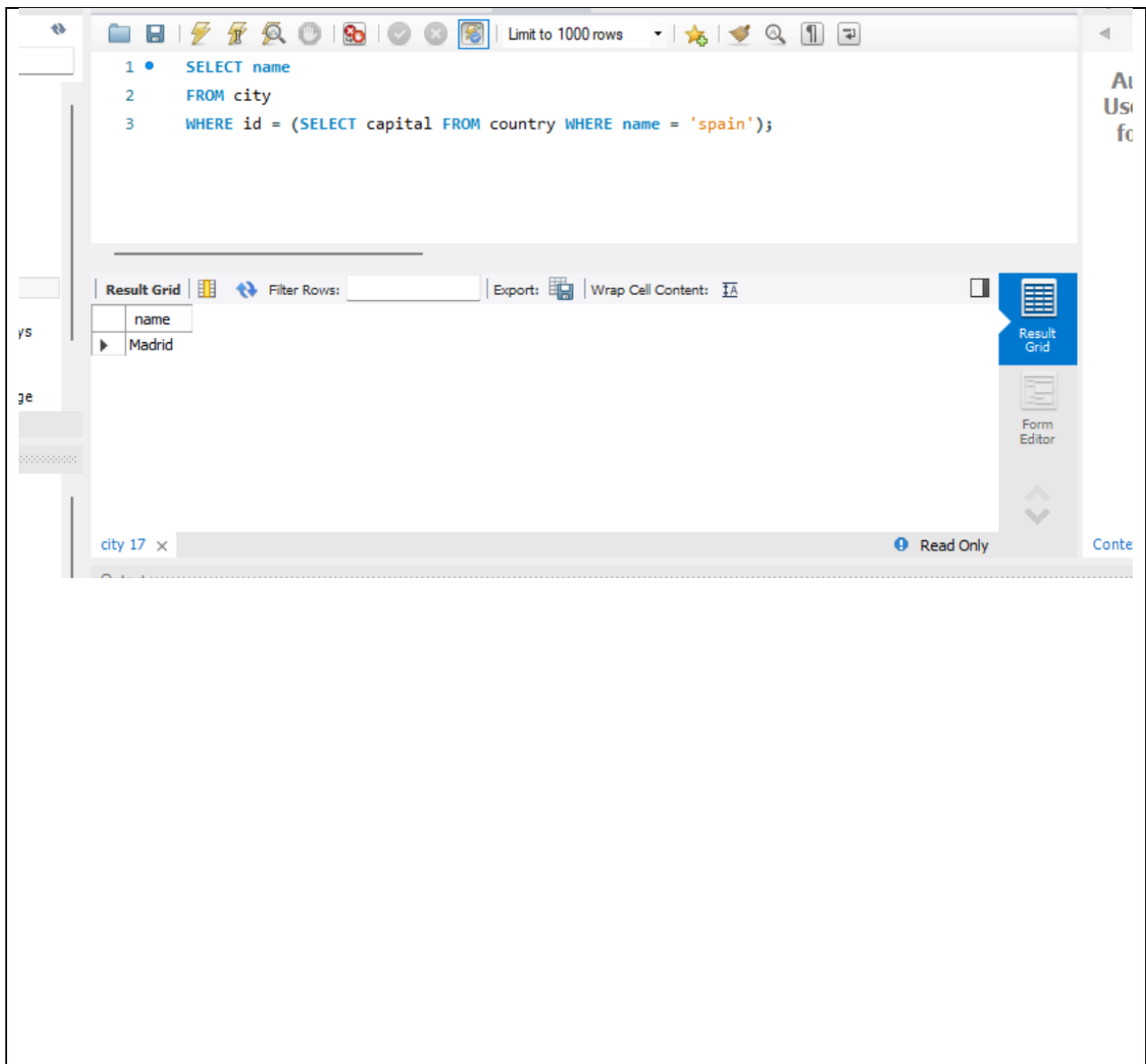
--





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. **Country with Shortest Life Expectancy:** *Scenario:* A healthcare foundation is conducting research on global health indicators. You're tasked with identifying the country with the highest life expectancy from the database to inform their efforts in improving healthcare systems and policies.

The screenshot shows a database query editor with a tab labeled 'city'. The SQL query is as follows:

```

1 • SELECT name, LifeExpectancy
2   FROM country
3  WHERE LifeExpectancy IS NOT NULL
4  ORDER BY LifeExpectancy ASC
5  LIMIT 1;

```

Below the query editor, the 'Result Grid' is displayed with the following data:

name	LifeExpectancy
Zambia	37.2

The interface includes a toolbar with icons for file operations, a 'Limit to 1000 rows' dropdown, and a 'Filter Rows' input field. On the right side, there are buttons for 'Result Grid' and 'Form Editor'. At the bottom, a status bar indicates 'country 18' and 'Read Only'.

15. **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.

The screenshot shows a database query editor with a tab labeled 'city'. The SQL query is as follows:

```

1 • SELECT ci.name, ci.countrycode
2   FROM city ci
3  JOIN country co ON ci.countrycode = co.Code
4  WHERE co.Continent = 'Europe';

```

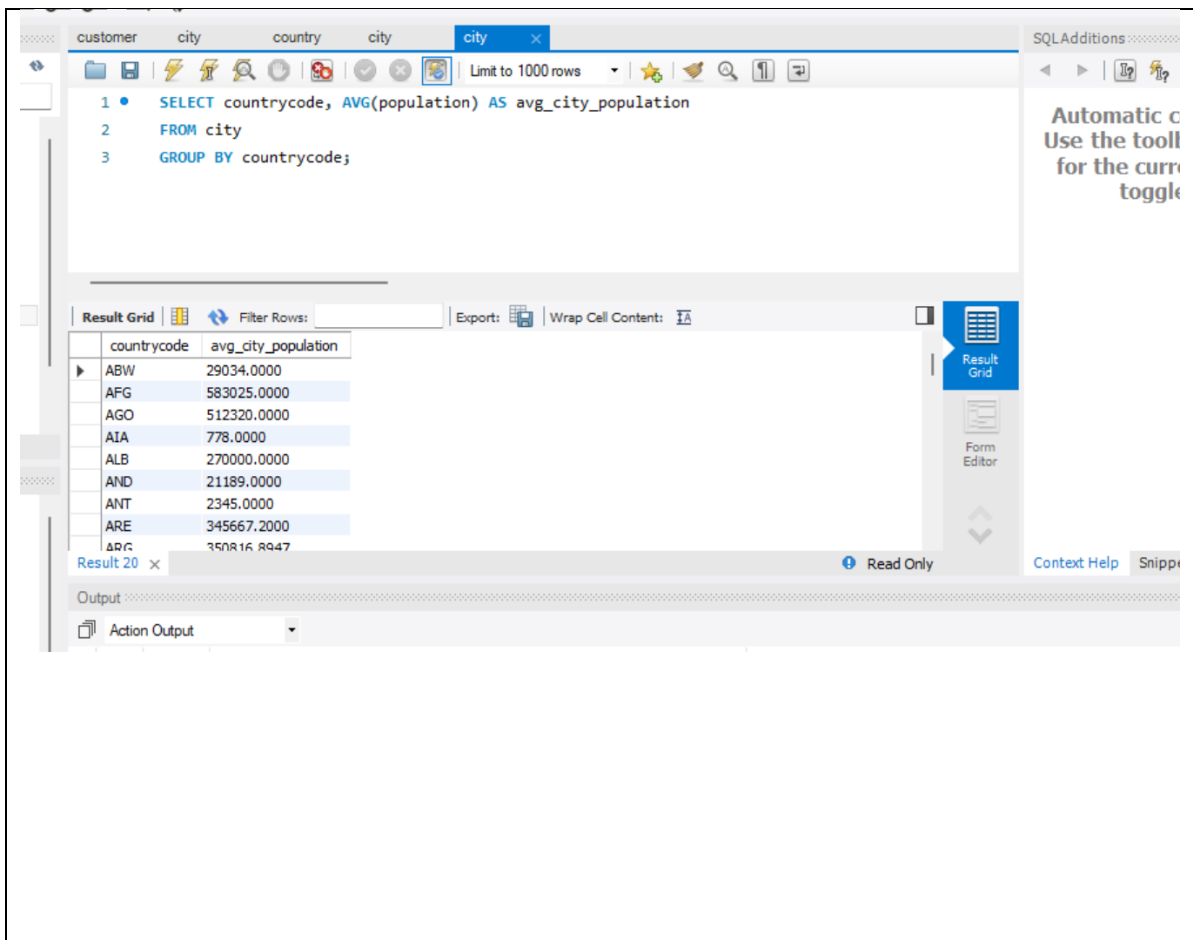
Below the query editor, the 'Result Grid' is displayed with the following data:

name	countrycode
Tirana	ALB
Andorra la Vella	AND
Wien	AUT
Graz	AUT
Linz	AUT
Salzburg	AUT
Innsbruck	AUT
Klagenfurt	AUT
Antwerpen	BEI

The interface includes a toolbar with icons for file operations, a 'Limit to 1000 rows' dropdown, and a 'Filter Rows' input field. On the right side, there are buttons for 'Result Grid' and 'Form Editor'. At the bottom, a status bar indicates 'Result 19' and 'Read Only'. A 'Context Help' button is also visible.



16. **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.



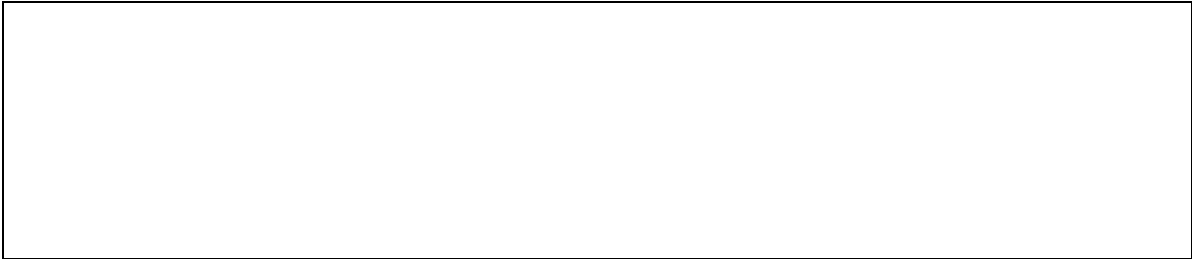
The screenshot displays a database management interface with a SQL query editor and a results grid. The query is as follows:

```
1 SELECT countrycode, AVG(population) AS avg_city_population
2 FROM city
3 GROUP BY countrycode;
```

The results grid shows the following data:

countrycode	avg_city_population
ABW	29034.0000
AFG	583025.0000
AGO	512320.0000
AIA	778.0000
ALB	270000.0000
AND	21189.0000
ANT	2345.0000
ARE	345667.2000
ARG	350216.8947

The interface includes a toolbar with various icons, a 'Limit to 1000 rows' dropdown, and a 'SQLAdditions' panel on the right. The results grid has a 'Filter Rows' section and an 'Export' button. The bottom of the interface shows an 'Output' section with an 'Action Output' dropdown.



17. **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.

The screenshot shows a database application interface. At the top, there are tabs for 'customer', 'city', 'country', 'city', and 'city'. Below the tabs is a toolbar with various icons. The main area displays an SQL query:

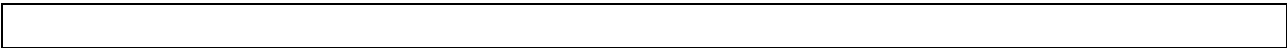
```
1 SELECT co.name AS country, ci.name AS capital, ci.population
2 FROM country co
3 JOIN city ci ON co.capital = ci.id
4 ORDER BY ci.population DESC;
```

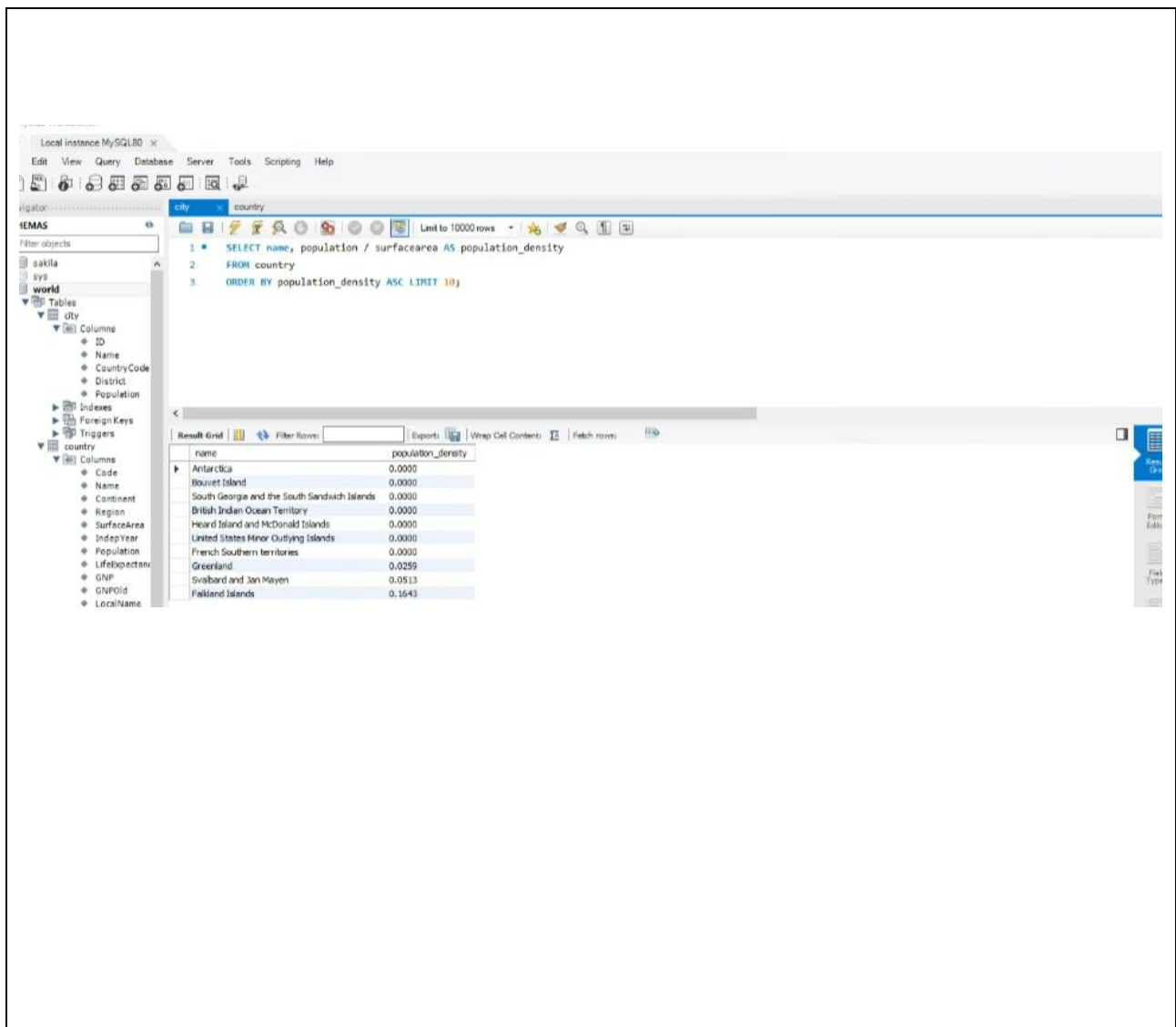
Below the query is a 'Result Grid' showing the results of the query. The grid has three columns: 'country', 'capital', and 'population'. The results are as follows:

country	capital	population
South Korea	Seoul	9981619
Indonesia	Jakarta	9604900
Mexico	Ciudad de México	8591309
Russian Federation	Moscow	8389200
Japan	Tokyo	7980230
China	Peking	7472000
United Kingdom	London	7285000
Egypt	Cairo	6789479
Iran	Teheran	6758845

On the right side of the interface, there is a sidebar with a 'Result Grid' button and a 'Form Editor' button. At the bottom, there is an 'Output' section.

18. **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.





19. **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.

The screenshot shows the MySQL Workbench interface. On the left, the 'Schemas' pane is open, showing the 'world' database. The 'country' table is selected. The main editor displays the following SQL query:

```

1 SELECT name, GNP/population AS gdp_per_capita
2 FROM country
3 WHERE (GNP/population) > (
4   SELECT AVG(GNP/population) FROM country
5 );

```

The 'Results' pane at the bottom shows the output of the query, which is a table with two columns: 'name' and 'gdp_per_capita'. The table contains 10 rows of data, including Aruba, Anguilla, Andorra, Netherlands Antilles, United Arab Emirates, Argentina, Antigua and Barbuda, Australia, Austria, and Belgium.

name	gdp_per_capita
Aruba	0.008039
Anguilla	0.007900
Andorra	0.020897
Netherlands Antilles	0.008945
United Arab Emirates	0.015553
Argentina	0.009388
Antigua and Barbuda	0.009000
Australia	0.018595
Austria	0.026182
Belgium	0.024388

20. **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.



MySQL Workbench
Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigation

city country

Limit to 10000 rows

SCHEMAS

Filter objects

sakila
 sys
 world
 Tables
 city
 Columns
 ID
 Name
 CountryCode
 District
 Population
 Indices
 ForeignKeys
 Triggers
 country
 Columns
 Code
 Name
 Continent
 Region
 SurfaceArea
 IndepYear
 Population
 LifeExpectancy
 GNP
 GNPPold
 LocalName
 Government
 HeadOfState
 Capital
 Code2
 Indices

1
2
3

SELECT name, countrycode, population
FROM city
ORDER BY population DESC LIMIT 31 OFFSET 40;

Result Grid

Filter Rows

Export

Wrap Cell Contents

Fetch rows

name	countrycode	population
Chengdu	CHN	3361500
Jokohama [Yokohama]	JPN	3339594
Alexandria	EGY	3328196
Riyadh	SAU	3324000
Sydney	AUS	3276307
Ankara	TUR	3038159
Buenos Aires	ARG	2902146
Hyderabad	IND	2564638
Casablanca	MAR	2940623
Chicago	USA	2896016
Madrid	ESP	2879052
Ahmedabad	IND	2876710
Nanking [Nanjing]	CHN	2870300
Melbourne	AUS	2865329
Chungking	CHN	2812000

SQL Additions
Automatic conl
to manually get
to

2



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:



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.

