Just IT

B2W group

Apprenticeships | Training | Recruitment

Data Technician

Name: Tetiana Semenchuk

Course Date: 29/05/2025

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 **unique identifier** for each row in a table. It ensures that **no two rows have the same value** in that column and that the value is **never NULL**.
How does this differ from a secondary key?	A **secondary key** is **any non-primary column** that is used to **look up data**. • It may or may not be unique. • It's mainly used to **speed up searches** or for querying/filtering. **Example:** In the same student table, you might frequently search by Name, so Name can be set as a **secondary key (index)**.
How are primary and foreign keys related?	• A **primary key** uniquely identifies records in its own table. • A **foreign key** is a field in one table that refers to the **primary key** in another table. It **links two tables together** and ensures **referential integrity**.

Provide a real-world example of a one-to-one relationship	1. **Person and Passport** • One person has **one unique passport**. • One passport belongs to **one person**. **2. User and UserProfile** • Each user has **one profile**. • Each profile belongs to **one user**. **3. Employee and WorkLaptop** • Each employee is assigned **one work laptop**. • Each work laptop is assigned to **only one employee**. **4. Vehicle and Registration** • One vehicle has **one registration record**. • Each registration record belongs to **one vehicle**. **5. Country and Capital City (simplified)** • One country has **one capital**. • A capital city is the capital of **one country**.

Provide a real-world example of a one-to-many relationship	***1. Author and Books*** • One author can write **many books**. • Each book has **only one author** (in a simple model). ***2. Department and Employees*** • One department has **many employees**. • Each employee belongs to **one department**. ***3. Country and Cities*** • One country can have **many cities**. • Each city is located in **one country**. ***4. Blog Post and Comments*** • One blog post can have **many comments**. • Each comment is associated with **one post**.
Provide a real-world example of a many-to-many relationship	1. **Students and Courses** • One student can take **many courses**. • One course can have **many students**. ***2. Doctors and Patients*** • A doctor may have **many patients**. • A patient may visit **multiple doctors**. ***3. Movies and Actors*** • A movie can feature **many actors**. • An actor can act in **many movies**. ***4. Teachers and Subjects*** • A teacher can teach **multiple subjects**. • A subject can be taught by **multiple teachers**.

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?

🆚 Relational vs. Non-Relational Databases

Feature	Relational DB (RDBMS)	Non-Relational DB (NoSQL)
Data Structure	Tables with rows and columns	Flexible formats: documents, key-value, graphs, wide-columns
Schema	Fixed schema (must define columns first)	Flexible or dynamic schema (can vary by record)
Relationships	Strong support for joins and relations	Usually no joins (use embedding or referencing instead)
Examples	MySQL, PostgreSQL, Oracle, SQL Server	MongoDB, Redis, Cassandra, Firebase, Neo4j
Best For	Structured data, complex queries	Semi-structured/unstructured data, scalability
ACID Compliance	Strong support (transactions, consistency)	Varies (some are eventually consistent)

What type of data would benefit off the non-relational model?

Why?

What type of data benefits from the Non-Relational model?

Non-relational databases work best for:

1. Unstructured or Semi-Structured Data

- JSON, XML, text, images, video, logs
- Example: Chat messages, product catalogs, social media posts

2. Fast-changing or Evolving Data

- Where the structure may vary between records or change frequently.
- Example: User profiles in a social media app — different users may have different fields.

3. Large-scale, high-speed applications

- High volumes of real-time data across many servers.
- Example: IoT sensor data, analytics logs, game state storage

4. Distributed systems that need scalability

- Easily scale horizontally (across servers)
- Example: Global content delivery networks, online stores

Why Non-Relational Databases for These Use Cases?

- **Flexibility**: No need to predefine a schema, making it easy to adapt to new data structures.
- **Performance**: Faster for certain operations (e.g., retrieving entire documents).

- **Scalability**: Designed to work well across many machines (great for big data or cloud systems).
- **Availability**: Often optimized for distributed systems and high availability.

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

A **self join** is when a table is joined with itself. It's used when rows in the same table are related to each other.

Example Use:

An **employees** table where each employee has a `manager_id` that refers to another employee in the same table.

```sql
SELECT A.name AS Employee, B.name AS Manager
FROM employees A
JOIN employees B ON A.manager_id = B.employee_id;
```

Right join

A **RIGHT JOIN** returns all rows from the **right table**, and matching rows from the **left table**. If there is no match, NULLs are returned for the left table.

Example Use:

Finding all **orders** and showing their related **customers**, even if some customers don't exist (e.g. deleted customers).

```sql
SELECT orders.order_id, customers.name
FROM orders
RIGHT JOIN customers ON orders.customer_id = customers.id;
```

Full join

A **FULL JOIN** returns all rows from both tables, with NULLs where there is no match.

Example Use:

Compare two lists, like `students_2024` and `students_2025`, to see:

- Who stayed
- Who left
- Who joined

```sql
SELECT a.name AS '2024', b.name AS '2025'
FROM students_2024 a
FULL JOIN students_2025 b ON a.student_id = b.student_id;
```

Inner join

An **INNER JOIN** returns only rows where there is a **match in both tables**. It's the most common join.

Example Use:

Get all **customers** who have placed **orders**.

```sql
SELECT customers.name, orders.order_id
FROM customers
INNER JOIN orders ON customers.id = orders.customer_id;
```

Cross join

A **CROSS JOIN** returns the **cartesian product** of two tables — every row from the first table is combined with every row from the second table.

Example Use:

Generate all combinations of **sizes** and **colors** for a product.

```sql
SELECT sizes.size, colors.color
FROM sizes
CROSS JOIN colors;
```

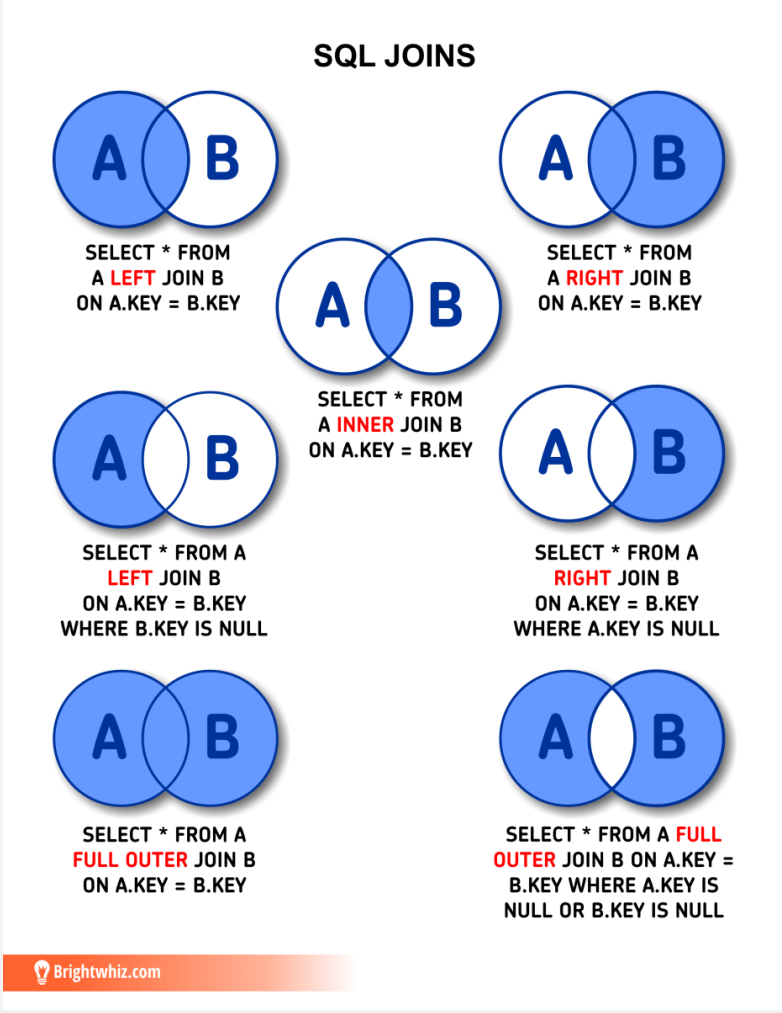
Left join

A **LEFT JOIN** returns all rows from the **left table**, and matching rows from the right table. If there's no match, NULLs are used.

Example Use:

List all **students** and any **classes** they are assigned to (including students who don't have a class yet).

```sql
SELECT students.name, classes.class_name
FROM students
LEFT JOIN classes ON students.class_id = classes.id;
```


SQL JOINS
A B
SELECT * FROM
A LEFT JOIN B
ON A.KEY = B.KEY
A B
SELECT * FROM
A RIGHT JOIN B
ON A.KEY = B.KEY
A B
SELECT * FROM
A INNER JOIN B
ON A.KEY = B.KEY
A B
SELECT * FROM A
LEFT JOIN B
ON A.KEY = B.KEY
WHERE B.KEY IS NULL
A B
SELECT * FROM A
RIGHT JOIN B
ON A.KEY = B.KEY
WHERE A.KEY IS NULL
A B
SELECT * FROM A
FULL OUTER JOIN B
ON A.KEY = B.KEY
A B
SELECT * FROM A FULL
OUTER JOIN B ON A.KEY =
B.KEY WHERE A.KEY IS
NULL OR B.KEY IS NULL
Brightwhiz.com

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.

Please write your 500-word essay here

Creating a Database for a Small Retail Business

To streamline operations for a small retail store that sells groceries and household items, a well-structured database is essential. This database will manage inventory, sales, customer information, and a loyalty program. Setting up this system involves understanding business needs, designing the schema, implementing the structure, populating data, and ensuring ongoing maintenance.

Understanding the Business Requirements

The database needs to store various types of data:
- Products: item name, category, quantity in stock, price.
- Sales: sale date, sold items, total amount.
- Customers: name, contact info, loyalty membership status.
- Loyalty Program: points earned, redeemed, and expiry.

The primary users of the database will be shop staff and possibly the manager or owner. Staff will use it to update inventory, process sales, and enroll customers into the loyalty program. Managers will use it to generate reports on sales and customer activity.

Designing the Database Schema

The database can be structured with the following tables:
1. Products(product_id, name, category, price, stock_qty)
2. Customers(customer_id, name, email, phone, loyalty_points)
3. Sales(sale_id, sale_date, customer_id, total_amount)
4. SaleItems(sale_item_id, sale_id, product_id, quantity, price)

Relationships:
- One customer can make many sales →1:N (Customers to Sales)
- One sale can have many products → 1:N (Sales to SaleItems)
- One product can be sold in many sales → 1:N (Products to

SaleItems)

Implementing the Database

To build the database, we use SQL CREATE TABLE commands. Here's an example:

```
CREATE TABLE Customers (
customer_id INT PRIMARY KEY,
name VARCHAR(100),
email VARCHAR(100),
phone VARCHAR(15),
loyalty_points INT DEFAULT 0
);

CREATE TABLE Products (
product_id INT PRIMARY KEY,
name VARCHAR(100),
category VARCHAR(50),
price DECIMAL(6,2),
stock_qty INT
);

CREATE TABLE Sales (
sale_id INT PRIMARY KEY,
sale_date DATE,
customer_id INT,
total_amount DECIMAL(8,2),
FOREIGN KEY (customer_id) REFERENCES
Customers(customer_id)
);

CREATE TABLE SaleItems (
sale_item_id INT PRIMARY KEY,
sale_id INT,
product_id INT,
quantity INT,
price DECIMAL(6,2),
```

```
FOREIGN KEY (sale_id) REFERENCES
Sales(sale_id),
FOREIGN KEY (product_id) REFERENCES
Products(product_id)
);
```

Populating the Database

To add data, we use INSERT INTO:

```
INSERT INTO Products VALUES (1, 'Milk', 'Dairy', 1.20, 50);
INSERT INTO Customers VALUES (1, 'Alice Brown',
'alice@email.com', '07123456789', 100);
INSERT INTO Sales VALUES (1, '2025-05-27', 1, 10.40);
INSERT INTO SaleItems VALUES (1, 1, 1, 2, 1.20);
```

Maintaining the Database

To keep data accurate, shop staff should regularly update stock and verify entries. You can use SQL UPDATE to reflect changes:

```
UPDATE Products SET stock_qty = stock_qty - 2
WHERE product_id = 1;
```

Backups should be scheduled daily using automated scripts or database tools. For security, access control is important: for example, allowing only the manager to delete or modify sales records.

Conclusion

A carefully designed and implemented database improves efficiency, tracks sales, supports marketing through loyalty points, and helps business decisions. With the right structure and ongoing maintenance, this small retail business can manage its operations smoothly and grow more effectively.

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.

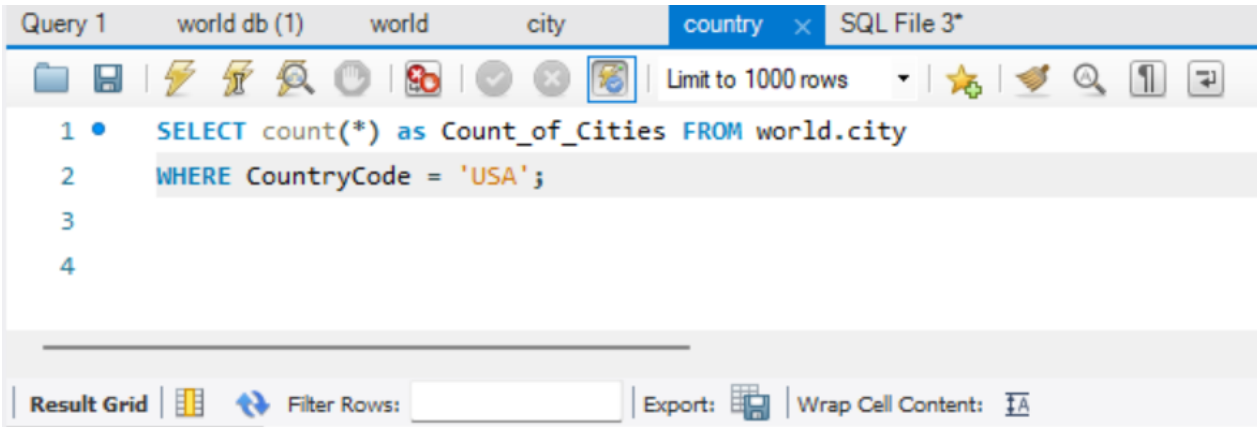
Setting up the database:

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

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

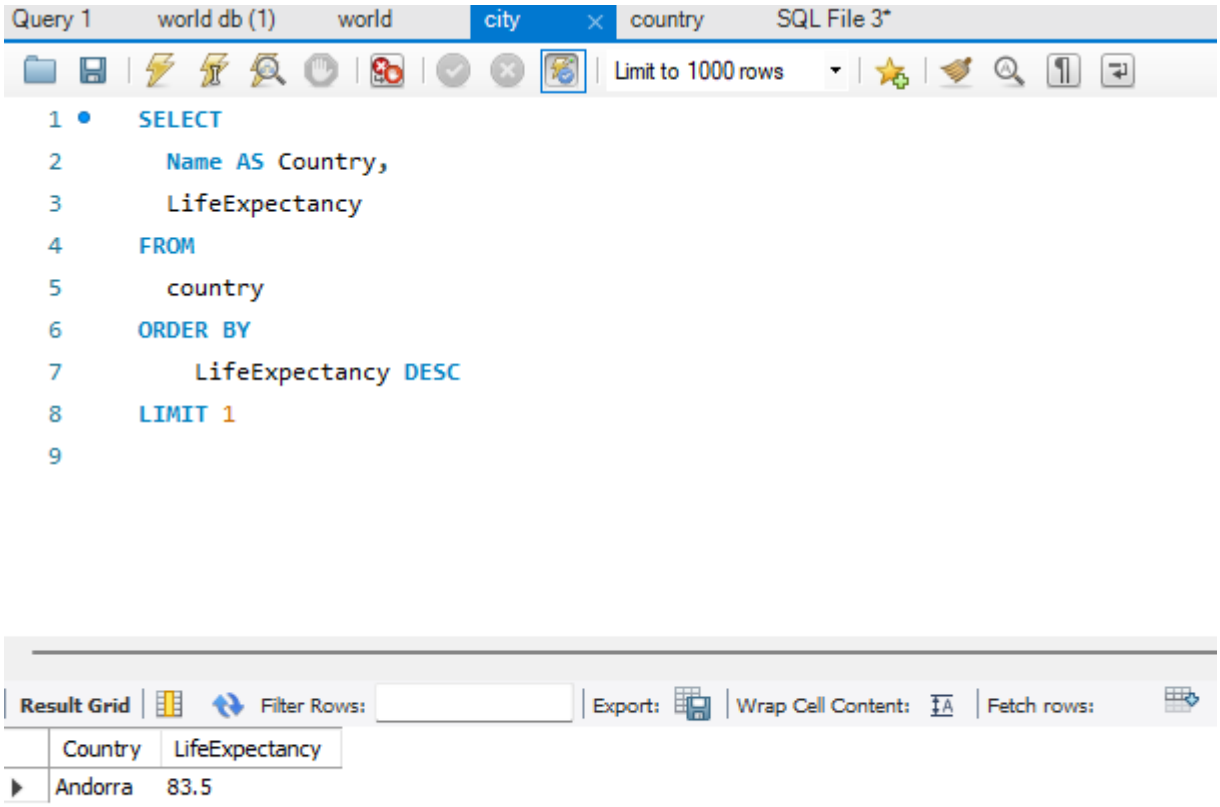
1. **Count Cities in USA:** *Scenario:* You've been tasked with conducting a demographic analysis of cities in the United States. Your first step is to determine the total number of cities within the country to provide a baseline for further analysis.

```
SELECT count(*) FROM world.city
WHERE CountryCode = 'USA';
```



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.

```
SELECT
  Name AS Country,
  LifeExpectancy
FROM
  country
ORDER BY
LifeExpectancy DESC
LIMIT 1
```



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.

```
SELECT
  ci.Name AS City,
  co.Name AS Country,
  ci.Population
FROM
  city ci
JOIN
  country co ON ci.CountryCode = co.Code
WHERE
ci.Name LIKE '%New%'
ORDER BY
  ci.Population DESC;
```

Query 1 | world db (1) | world | city | country | SQL File 3*

Limit to 1000 rows

```
1  SELECT
2    ci.Name AS City,
3    co.Name AS Country,
4    ci.Population
5  FROM
6    city ci
7  JOIN
8    country co ON ci.CountryCode = co.Code
9  WHERE
10     ci.Name LIKE '%New%'
11 ORDER BY
12   ci.Population DESC;
13
```

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

City	Country	Population
New York	United States	8008278
Kowloon and New Kowloon	Hong Kong	1987996
New Orleans	United States	484674
New Bombay	India	307297
New Delhi	India	301297
Newark	United States	273546
Newcastle	Australia	270324
Newcastle	South Africa	222993
Newcastle upon Tyne	United Kingdom	189150
Newport News	United States	180150
Newport	United Kingdom	139000
Khanewal	Pakistan	133000
New Haven	United States	123626
New Bedford	United States	94780

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.

```
SELECT
  ci.Name AS City,
  co.Name AS Country,
  ci.Population
FROM
  city ci
JOIN
  country co ON ci.CountryCode = co.Code
ORDER BY
  ci.Population DESC
LIMIT 10;
```

Query 1 | world db (1) | world | city | country | SQL File 3*

Limit to 1000 rows

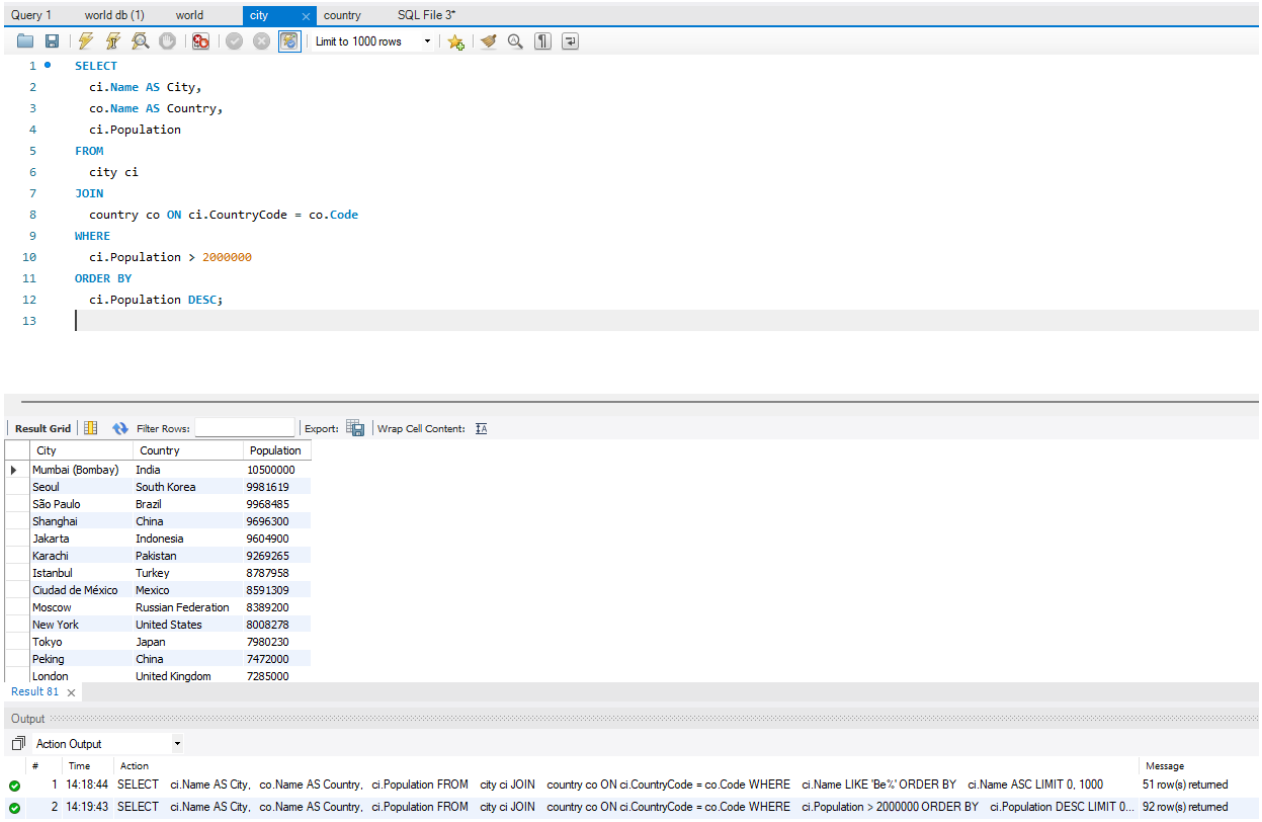
```
1  SELECT
2    ci.Name AS City,
3    co.Name AS Country,
4    ci.Population
5  FROM
6    city ci
7  JOIN
8    country co ON ci.CountryCode = co.Code
9  ORDER BY
10   ci.Population DESC
11 LIMIT 10;
12
```

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

City	Country	Population
Mumbai (Bombay)	India	10500000
Seoul	South Korea	9981619
São Paulo	Brazil	9968485
Shanghai	China	9696300
Jakarta	Indonesia	9604900
Karachi	Pakistan	9269265
Istanbul	Turkey	8787958
Ciudad de México	Mexico	8591309
Moscow	Russian Federation	8389200
New York	United States	8008278

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.

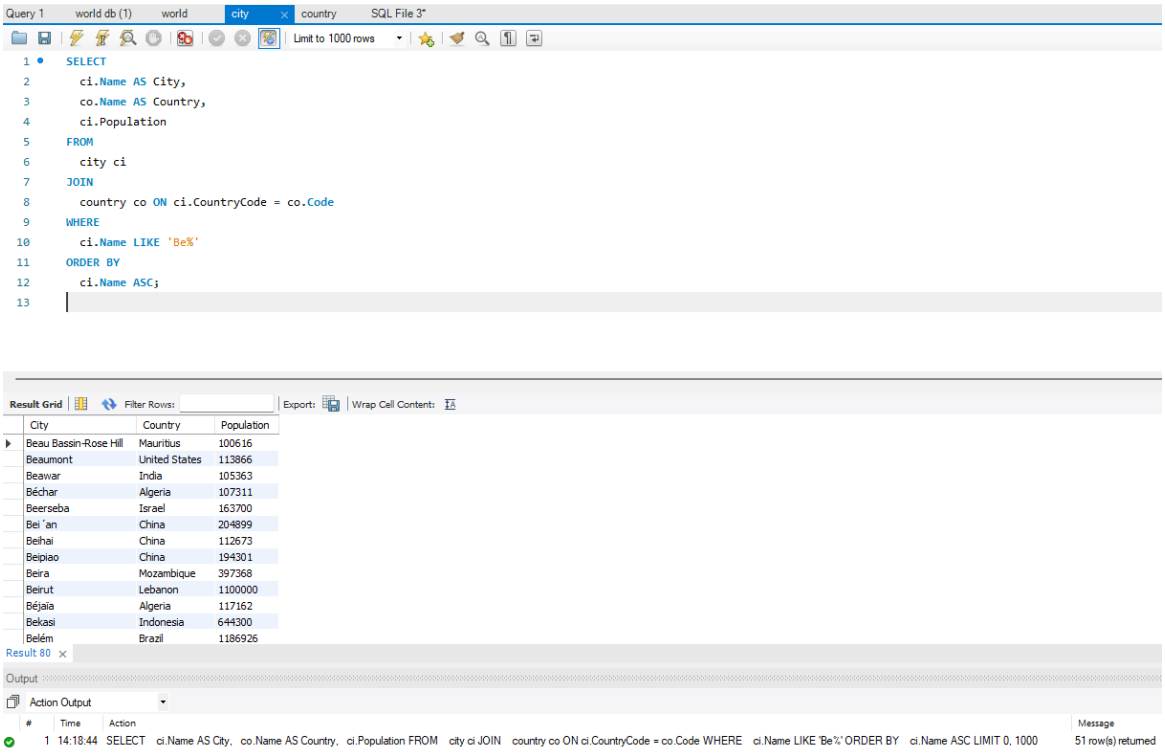
```
SELECT
  ci.Name AS City,
  co.Name AS Country,
  ci.Population
FROM
  city ci
JOIN
  country co ON ci.CountryCode = co.Code
WHERE
  ci.Population > 2000000
ORDER BY
  ci.Population DESC;
```



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.

```
SELECT
  ci.Name AS City,
  co.Name AS Country,
  ci.Population
FROM
  city ci
JOIN
  country co ON ci.CountryCode = co.Code
WHERE
  ci.Name LIKE 'Be%'
ORDER BY
  ci.Name ASC;
```



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.

```
SELECT
  ci.Name AS City,
  co.Name AS Country,
  ci.Population
FROM
  city ci
JOIN
  country co ON ci.CountryCode = co.Code
WHERE
  ci.Population BETWEEN 500000 AND 1000000
ORDER BY
  ci.Population ASC;
```

Query 1 | world db (1) | world | city | country | SQL File 3*

Limit to 1000 rows

```
1  SELECT
2    ci.Name AS City,
3    co.Name AS Country,
4    ci.Population
5  FROM
6    city ci
7  JOIN
8    country co ON ci.CountryCode = co.Code
9  WHERE
10   ci.Population BETWEEN 500000 AND 1000000
11 ORDER BY
12   ci.Population DESC
```

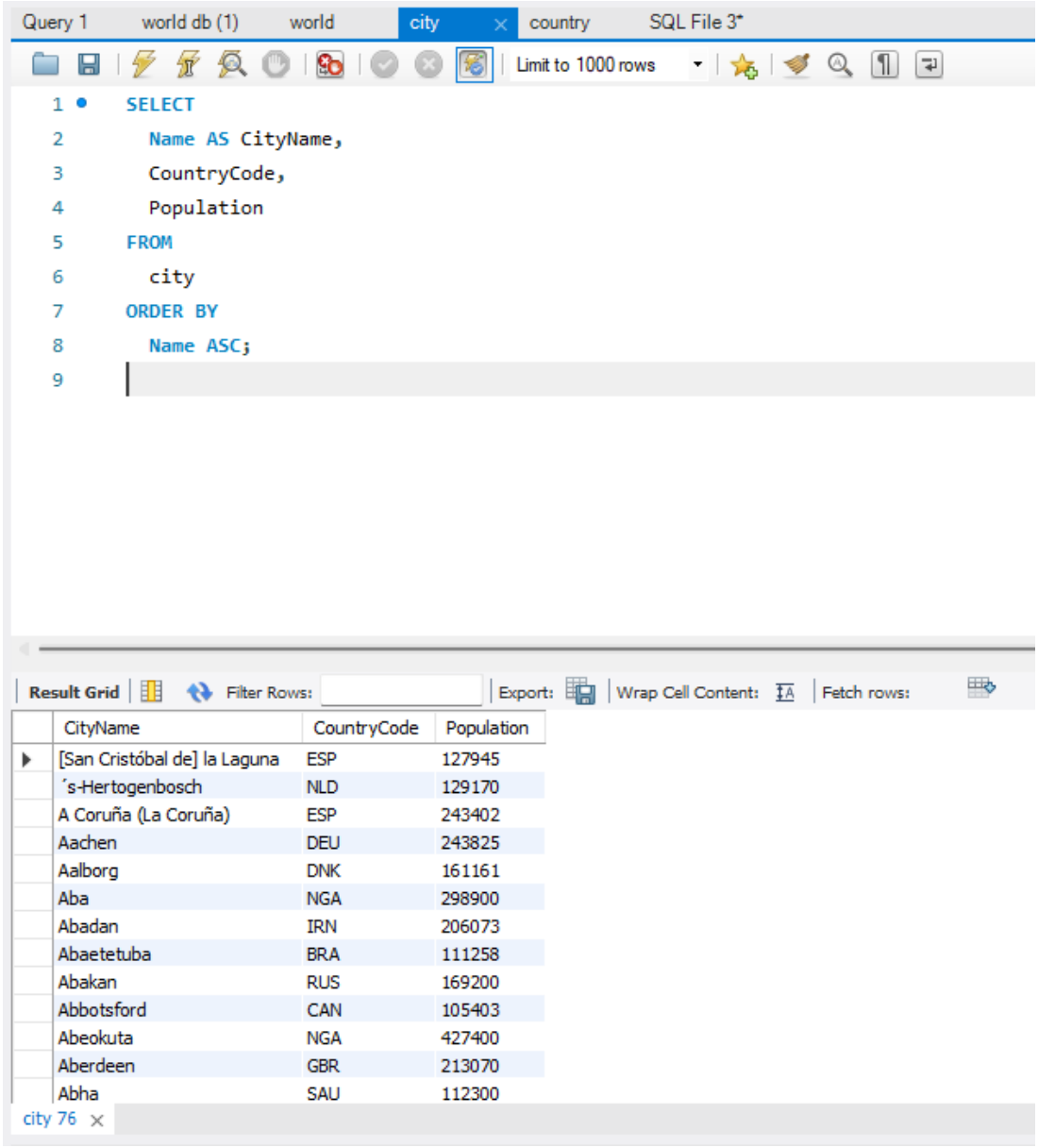
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

City	Country	Population
Amman	Jordan	1000000
Mogadishu	Somalia	997000
Volgograd	Russian Federation	993400
Sendai	Japan	989975
Peshawar	Pakistan	988005
Baotou	China	980000
Adelaide	Australia	978100
Madurai	India	977856
Mekka	Saudi Arabia	965700
Köln	Germany	962507
Managua	Nicaragua	959000
Detroit	United States	951270
Shenzhen	China	950500

Result 78

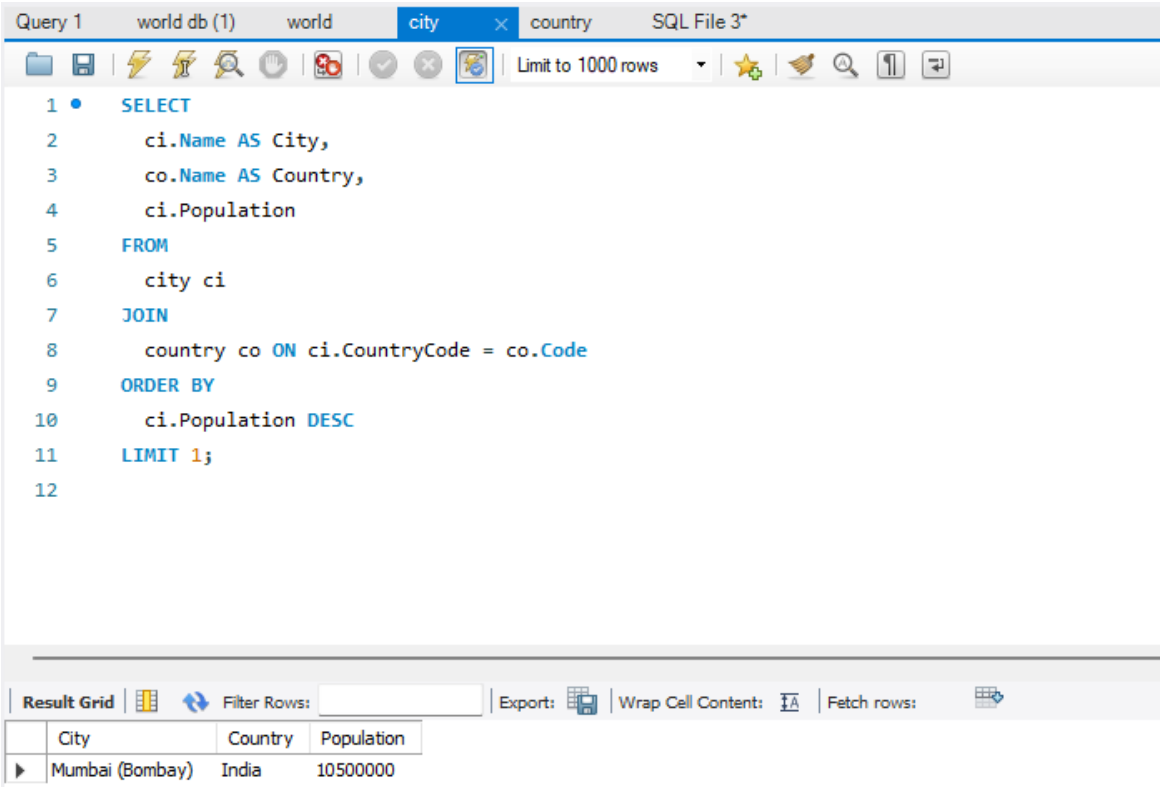
8. **Display Cities Sorted by Name in Ascending Order:** *Scenario:* A geography teacher is preparing a lesson on alphabetical order using city names. You're tasked with providing a sorted list of cities from the database in ascending order by name to support the lesson plan.

```
SELECT
  Name AS CityName,
  CountryCode,
  Population
FROM
  city
ORDER BY
  Name ASC;
```



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.

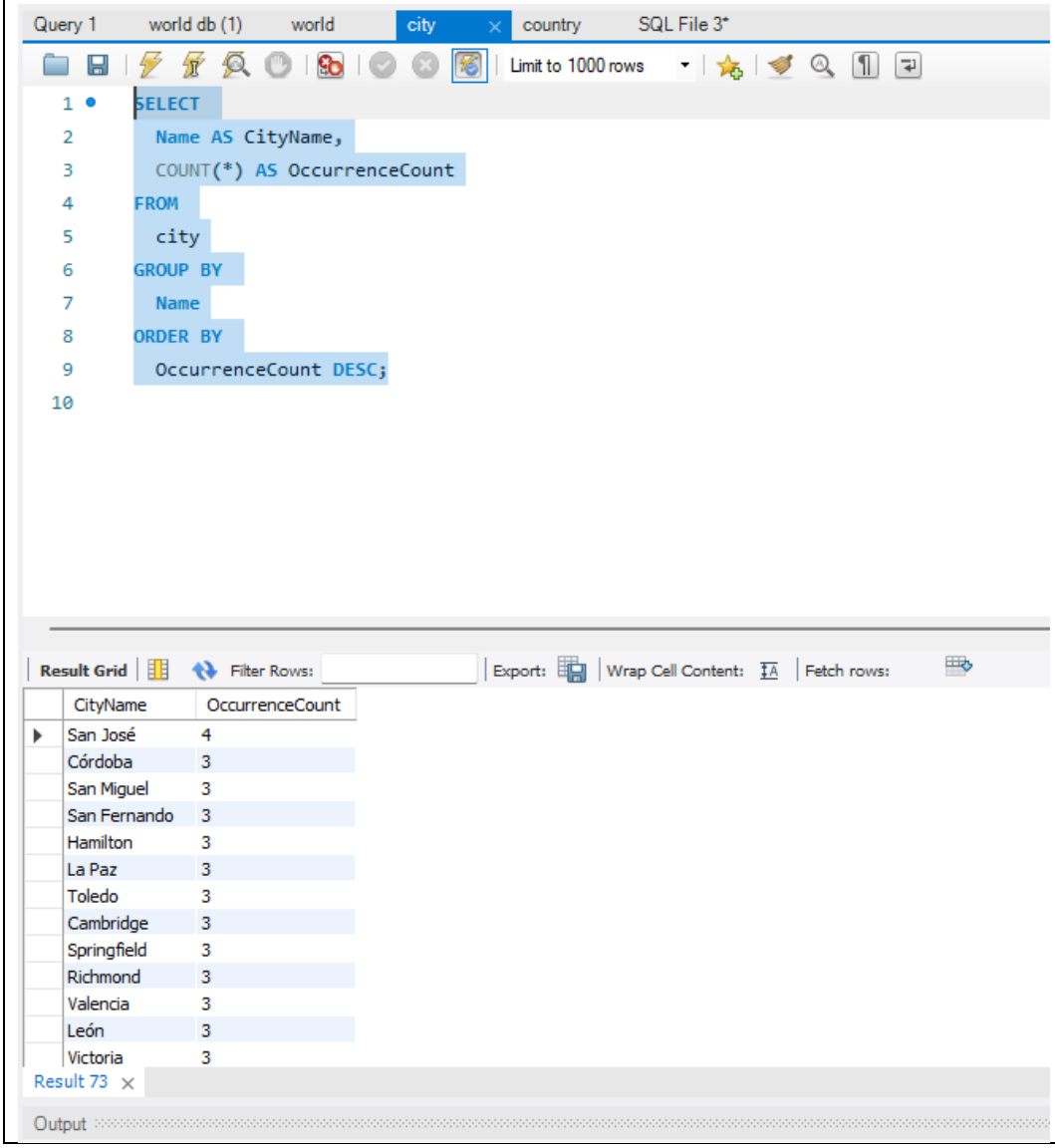
```
SELECT
  ci.Name AS City,
  co.Name AS Country,
  ci.Population
FROM
  city ci
JOIN
  country co ON ci.CountryCode = co.Code
ORDER BY
  ci.Population DESC
LIMIT 1;
```



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.

```
SELECT
  Name AS CityName,
  COUNT(*) AS OccurrenceCount
FROM
  city
GROUP BY
  Name
ORDER BY
  OccurrenceCount DESC;
```

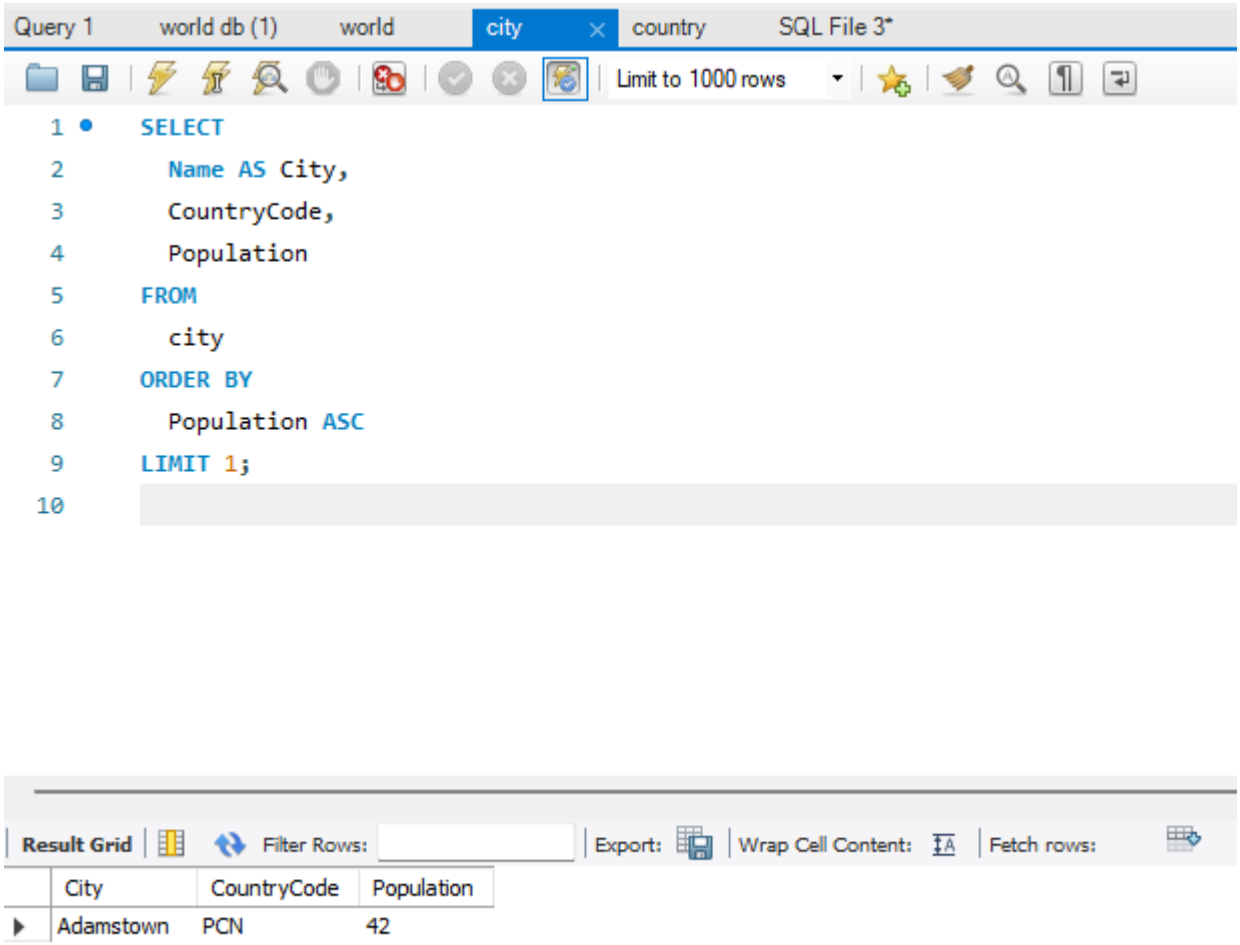


CityName	OccurrenceCount
San José	4
Córdoba	3
San Miguel	3
San Fernando	3
Hamilton	3
La Paz	3
Toledo	3
Cambridge	3
Springfield	3
Richmond	3
Valencia	3
León	3
Victoria	3

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

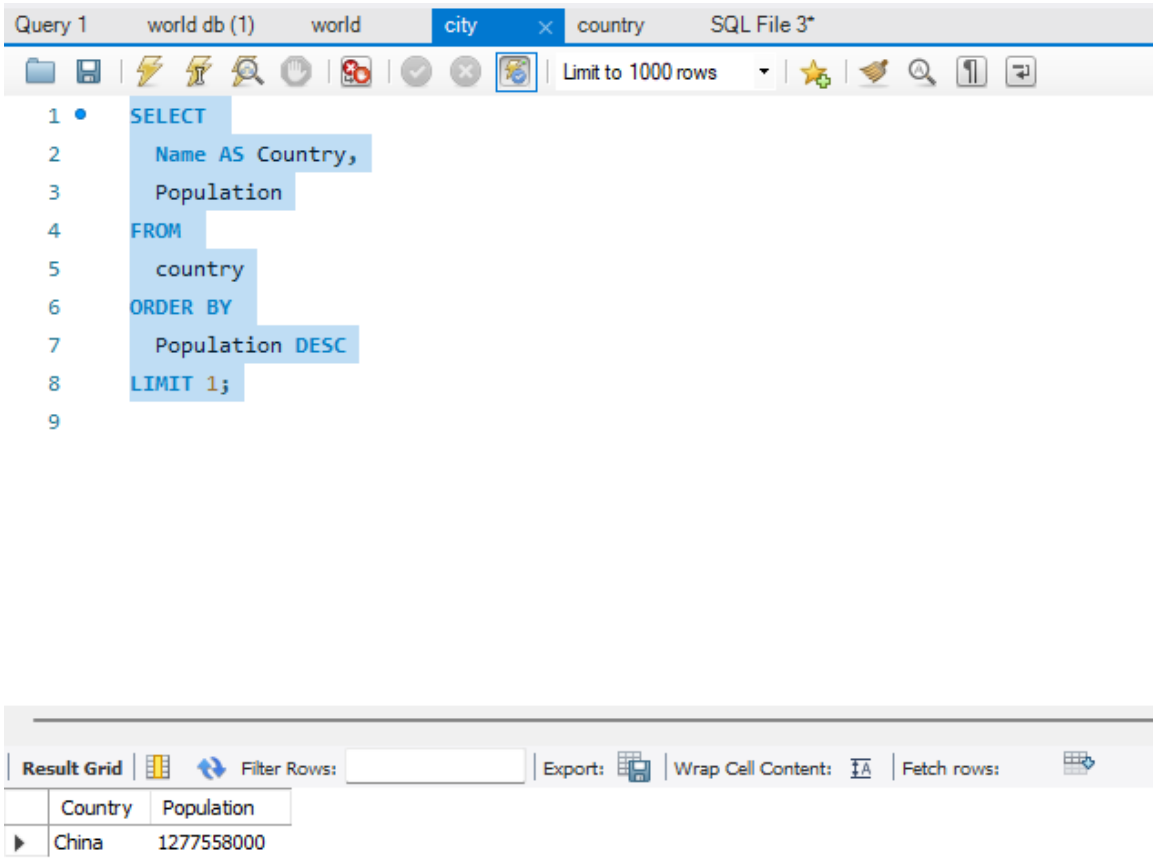
the lowest population from the database to provide a comprehensive overview of demographic trends.

```
SELECT
  Name AS City,
  CountryCode,
  Population
FROM
  city
ORDER BY
  Population ASC
LIMIT 1;
```



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.

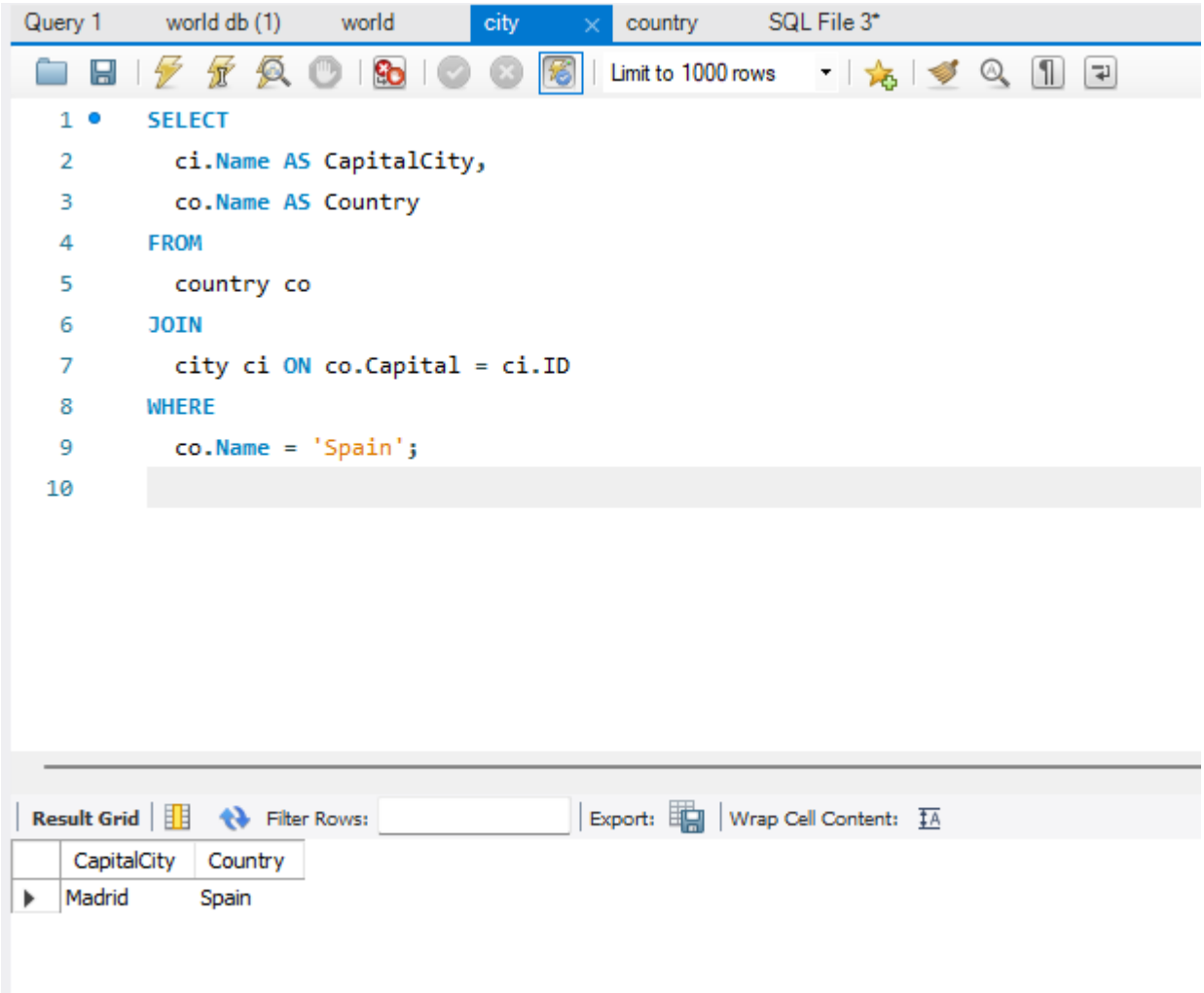
```
SELECT
  Name AS Country,
  Population
FROM
  country
ORDER BY
  Population DESC
LIMIT 1;
```



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

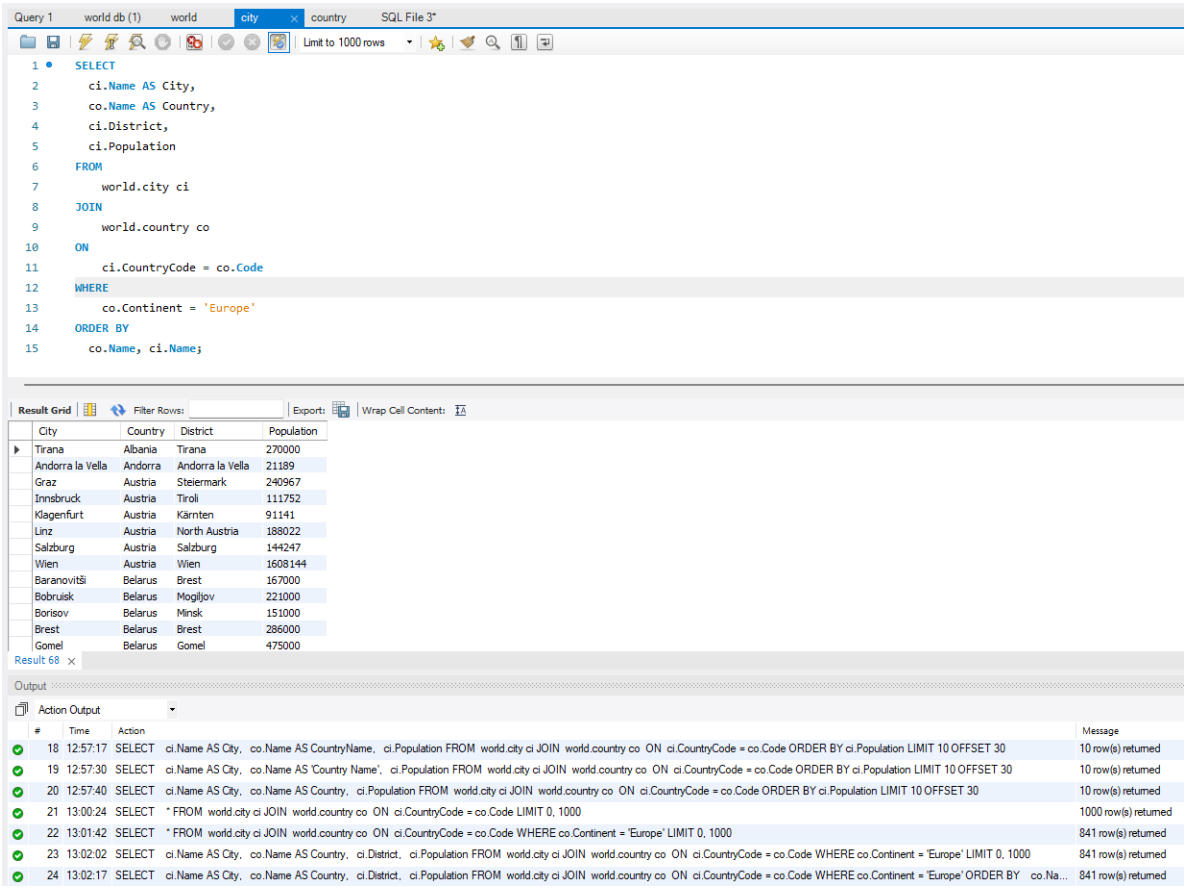
capital of Spain from the database to ensure itinerary accuracy and provide travellers with essential destination information.

```
SELECT
  ci.Name AS CapitalCity,
  co.Name AS Country
FROM
  country co
JOIN
  city ci ON co.Capital = ci.ID
WHERE
  co.Name = 'Spain';
```



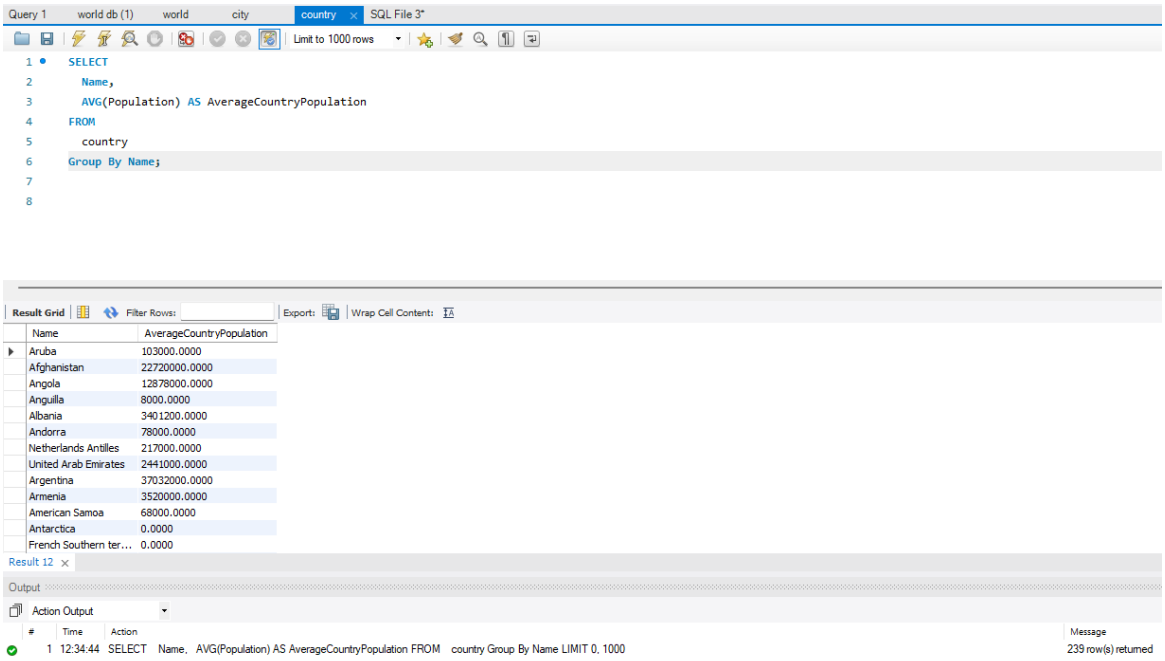
14. **Cities in Europe:** *Scenario:* A European cultural exchange program is seeking to connect students with cities across the continent. You're tasked with compiling a list of cities located in Europe from the database to facilitate program planning and student engagement.

```
SELECT
  ci.Name AS City,
  co.Name AS Country,
  ci.District,
  ci.Population
FROM
world.city ci
JOIN
world.country co
ON
ci.CountryCode = co.Code
WHERE
co.Continent = 'Europe'
ORDER BY
  co.Name, ci.Name;
```



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.

```
SELECT
  Name,
  AVG(Population) AS AverageCountryPopulation
FROM
  country
Group By Name;
```




```
SELECT
  c.Name AS Country,
  COUNT(ci.ID) AS NumberOfCities,
  AVG(ci.Population) AS AverageCityPopulation
FROM
  country c
JOIN
  city ci ON c.Code = ci.CountryCode
GROUP BY
  c.Code, c.Name
ORDER BY
  AverageCityPopulation DESC;
```

Query 1 | world db (1) | world | city | country | SQL File 3*

Limit to 1000 rows

```sql
SELECT
  c.Name AS Country,
  COUNT(ci.ID) AS NumberOfCities,
  AVG(ci.Population) AS AverageCityPopulation
FROM
  country c
JOIN
  city ci ON c.Code = ci.CountryCode
GROUP BY
  c.Code, c.Name
ORDER BY
  AverageCityPopulation DESC;
```

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

Country	NumberOfCities	AverageCityPopulation
Singapore	1	4017733.0000
Hong Kong	2	1650316.5000
Uruguay	1	1236000.0000
Guinea	1	1090610.0000
Uganda	1	890800.0000
Liberia	1	850000.0000
Sierra Leone	1	850000.0000
Mali	1	809552.0000
Australia	14	808119.0000
Mongolia	1	773700.0000
Congo	2	725000.0000
Libyan Arab...	4	674251.7500
Lebanon	2	670000.0000

Result 13

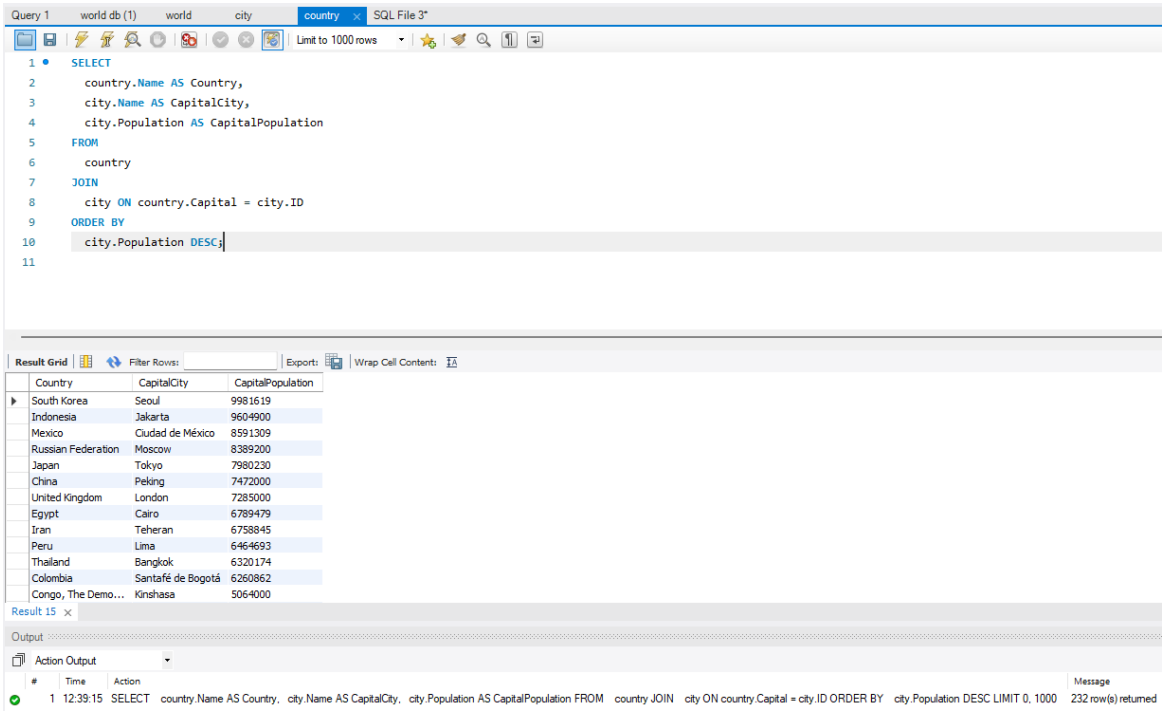
Output

Action Output

#	Time	Action	Message
1	12:34:44	SELECT Name, AVG(Population) AS AverageCountryPopulation FROM country Group By Name LIMIT 0, 1000	239 row(s) returned
2	12:35:32	SELECT c.Name AS Country, COUNT(ci.ID) AS NumberOfCities, AVG(ci.Population) AS AverageCityPopulation FROM country c JOIN city ci ON c.Code = ci.CountryCode GROUP BY c.Code, c.Name ...	232 row(s) returned

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.

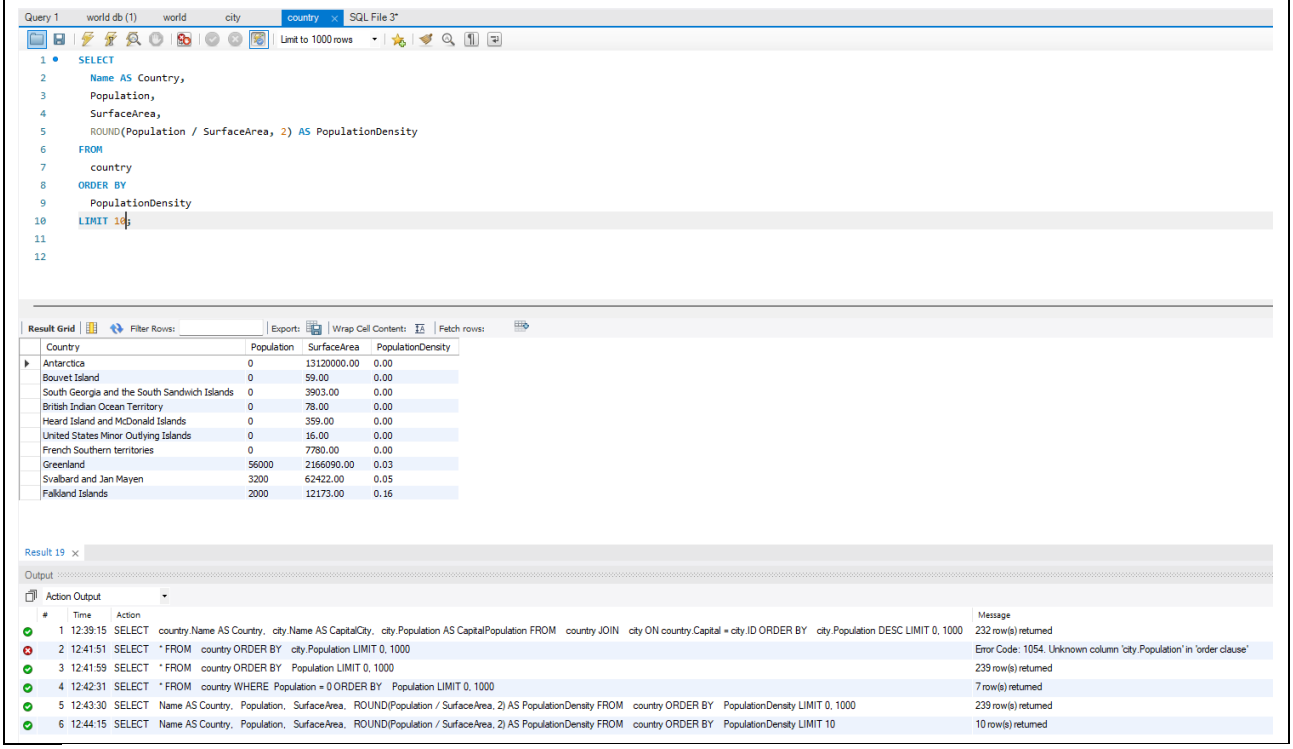
```
SELECT
  country.Name AS Country,
  city.Name AS CapitalCity,
  city.Population AS CapitalPopulation
FROM
  country
JOIN
  city ON country.Capital = city.ID
ORDER BY
  city.Population DESC;
```



Country	CapitalCity	CapitalPopulation
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
Peru	Lima	6464693
Thailand	Bangkok	6320174
Colombia	Santafé de Bogotá	6260862
Congo, The Demo...	Kinshasa	5064000

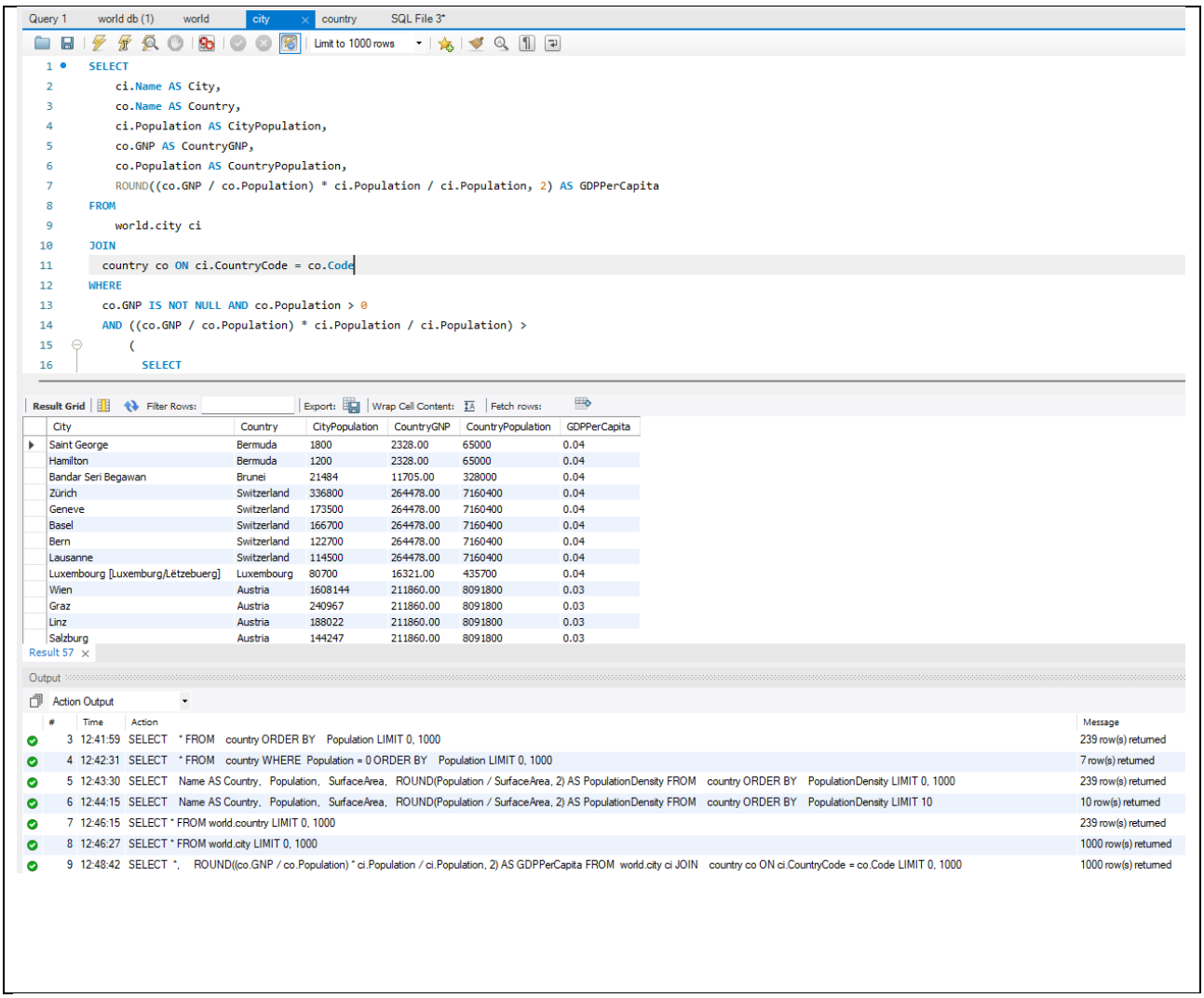
17. **Countries with Low Population Density:** *Scenario:* An agricultural research institute is studying countries with low population densities for potential agricultural development projects. You're tasked with identifying countries with sparse populations from the database to support the institute's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

```
SELECT
  Name AS Country,
  Population,
  SurfaceArea,
  ROUND(Population / SurfaceArea, 2) AS PopulationDensity
FROM
  country
ORDER BY
  PopulationDensity
LIMIT 10;
```



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.

```
SELECT
ci.Name AS City,
    co.Name AS Country,
    ci.Population AS CityPopulation,
    co.GNP AS CountryGNP,
    co.Population AS CountryPopulation,
    ROUND((co.GNP / co.Population) * ci.Population / ci.Population, 2) AS GDPPerCapita
FROM
world.city ci
JOIN
  country co ON ci.CountryCode = co.Code
WHERE
  co.GNP IS NOT NULL AND co.Population > 0
  AND ((co.GNP / co.Population) * ci.Population / ci.Population) >
      (
        SELECT
          AVG(co.GNP / co.Population)
        FROM
          country co
        WHERE
          co.GNP IS NOT NULL AND co.Population > 0
      )
ORDER BY
  GDPPerCapita DESC;
```



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.

```
SELECT
  ci.Name AS City,
  co.Name AS Country,
  ci.Population
FROM
world.city ci
JOIN
world.country co
ON
ci.CountryCode = co.Code
ORDER BY
ci.Population
LIMIT 10 OFFSET 30
```

Query 1 | world db (1) | world | city | country | SQL File 3*

Limit to 1000 rows

```sql
SELECT
  ci.Name AS City,
  co.Name AS Country,
  ci.Population
FROM
    world.city ci
JOIN
    world.country co
ON
    ci.CountryCode = co.Code
ORDER BY
    ci.Population
LIMIT 10 OFFSET 30
```

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

City	Country	Population
Vaduz	Liechtenstein	5043
Bikenibeu	Kiribati	5055
Tafuna	American Samoa	5200
Schaan	Liechtenstein	5346
Saint-Pierre	Saint Pierre and Miquelon	5808
Bridgetown	Barbados	6070
Valletta	Malta	7073
Belmopan	Belize	7105
Road Town	Virgin Islands, British	8000
Palikir	Micronesia, Federated States of	8600

Result 64

Output

Action Output

#	Time	Action	Message
14	12:54:29	SELECT Name AS City, CountryCode, Population FROM world.city ci ORDER BY Population LIMIT 10 OFFSET 30	10 row(s) returned
15	12:55:36	SELECT Name AS City, co.Name, Population FROM world.city ci JOIN word.country co ON ci.CountryCode = co.Code ORDER BY Population LIMIT 10 OFFSET 30	Error Code: 1049. Unknown datab
16	12:55:44	SELECT Name AS City, co.Name, Population FROM world.city ci JOIN world.country co ON ci.CountryCode = co.Code ORDER BY Population LIMIT 10 OFFSET 30	Error Code: 1052. Column 'Name'
17	12:56:59	SELECT ci.Name AS City, co.Name, ci.Population FROM world.city ci JOIN world.country co ON ci.CountryCode = co.Code ORDER BY ci.Population LIMIT 10 OFFSET 30	10 row(s) returned
18	12:57:17	SELECT ci.Name AS City, co.Name AS CountryName, ci.Population FROM world.city ci JOIN world.country co ON ci.CountryCode = co.Code ORDER BY ci.Population LIMIT 10 OFFSET 30	10 row(s) returned
19	12:57:30	SELECT ci.Name AS City, co.Name AS 'Country Name', ci.Population FROM world.city ci JOIN world.country co ON ci.CountryCode = co.Code ORDER BY ci.Population LIMIT 10 OFFSET 30	10 row(s) returned
20	12:57:40	SELECT ci.Name AS City, co.Name AS Country, ci.Population FROM world.city ci JOIN world.country co ON ci.CountryCode = co.Code ORDER BY ci.Population LIMIT 10 OFFSET 30	10 row(s) returned

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:

Database: Organised collection of data.

Relational Database: Data with table relationships.

Schema: Structure of the database.

Table: Structured set of rows and columns.

Column: Vertical set of fields.

Row: Horizontal record in a table.

Field: Single piece of data.

Record: Complete set of related fields.

Key: Identifier for data retrieval.

Primary Key: Unique row identifier.

Foreign Key: Links to another table's primary key.

SQL: Language to manage databases.

View: Virtual table from a query.

WHERE MAY WE USE DATABASES?

- **Websites** – to store user accounts, blog posts, comments, etc. (e.g., Instagram, Amazon)
- **Banking Systems** – to store customer details, transactions, account balances
- **Hospitals** – to manage patient records, appointments, prescriptions
- **Schools/Universities** – to track students, grades, schedules

- **Online Shopping** – to store product data, inventory, orders
- **Government Services** – for ID records, taxes, census data
- **Mobile Apps** – for things like saving messages, game progress, and user settings

WHAT TOOLS CAN WE USE TO WORK WITH DATABASES?

1. Database Management Systems (DBMS)

- **MySQL / MariaDB** – open-source and widely used for web development
- **PostgreSQL** – powerful, open-source, great for complex queries
- **SQLite** – lightweight, often used in mobile apps and small projects
- **Microsoft SQL Server** – used in enterprises, works well with Microsoft products
- **Oracle Database** – large-scale enterprise use

2. Database Tools / Interfaces

- **phpMyAdmin** – GUI tool for MySQL
- **pgAdmin** – GUI tool for PostgreSQL
- **SQL Server Management Studio (SSMS)** – for Microsoft SQL Server
- **DBeaver / HeidiSQL / DataGrip** – multi-database GUI tools

3. Programming Languages to Connect to DBs

- **Python** – with libraries like sqlite3, SQLAlchemy, psycopg2
- **JavaScript (Node.js)** – with mongoose, sequelize, mysql2
- **PHP** – with PDO or MySQLi

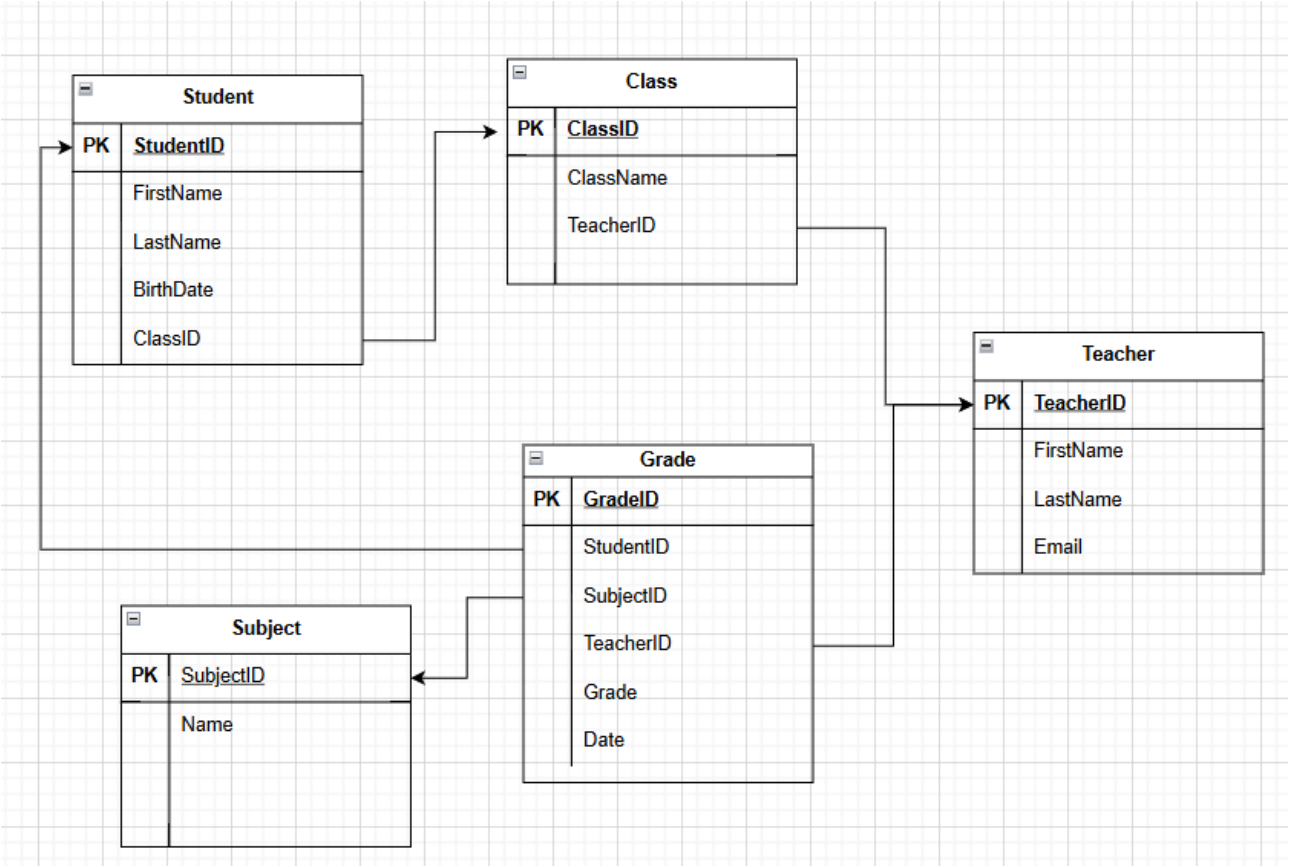
BENEFITS OF USING DATABASES

1. **Efficient Data Storage** – save large amounts of structured data.
2. **Fast Data Retrieval** – find and filter info quickly using queries.
3. **Data Integrity** – ensures data is accurate and consistent.
4. **Multi-user Access** – several people can work with the data at the same time.
5. **Security** – restrict access to sensitive data.
6. **Backup and Recovery** – protect data from loss.
7. **Scalability** – can handle growing data as your app or business grows.

DRAWBACKS OF RELATIONAL DATABASES

1. **Not Ideal for Unstructured Data**
    a. RDBs are designed for structured data (tables, rows, and columns).
    b. They struggle with unstructured formats like images, videos, or large documents.
    c. For example: Storing chat logs, social media posts, or multimedia can be harder.

2. **Scalability Limitations (especially horizontal scaling)**
    a. RDBs scale **vertically** (adding more power to one server), which has limits and can get expensive.
    b. Scaling **horizontally** (adding more servers) is more complex in RDBs compared to NoSQL systems.
3. **Complex Schema Management**
    a. The schema (structure) must be defined before storing data.
    b. Changing the schema (e.g., adding a new column) can be hard and time-consuming in large databases.
4. **Joins Can Be Expensive**
    a. Joins (combining data from multiple tables) are powerful but can slow down performance if overused or on large datasets.
5. **Rigid Data Structure**
    a. All data must fit the table structure. If data is flexible or varies a lot between entries, it's harder to manage.
6. **Overhead for Small or Simple Applications**
    a. For small apps or projects, RDBs might be too heavy or complex.
    b. Tools like SQLite may work better in simple use cases.
7. **Handling Big Data / Real-Time Data**
    a. RDBs aren't optimized for huge volumes of rapidly changing data (e.g., sensor data, IoT).
    b. NoSQL databases like MongoDB, Cassandra, or time-series DBs like InfluxDB handle this better.
8. **More Complex to Use in Distributed Systems**
    a. RDBs often assume one central location for data.
    b. Making them work across multiple locations or data centers can be difficult.


Student
PK
StudentID
FirstName
LastName
BirthDate
ClassID
Class
PK
ClassID
ClassName
TeacherID
Teacher
PK
TeacherID
FirstName
LastName
Email
Grade
PK
GradeID
StudentID
SubjectID
TeacherID
Grade
Date
Subject
PK
SubjectID
Name

DAY 2

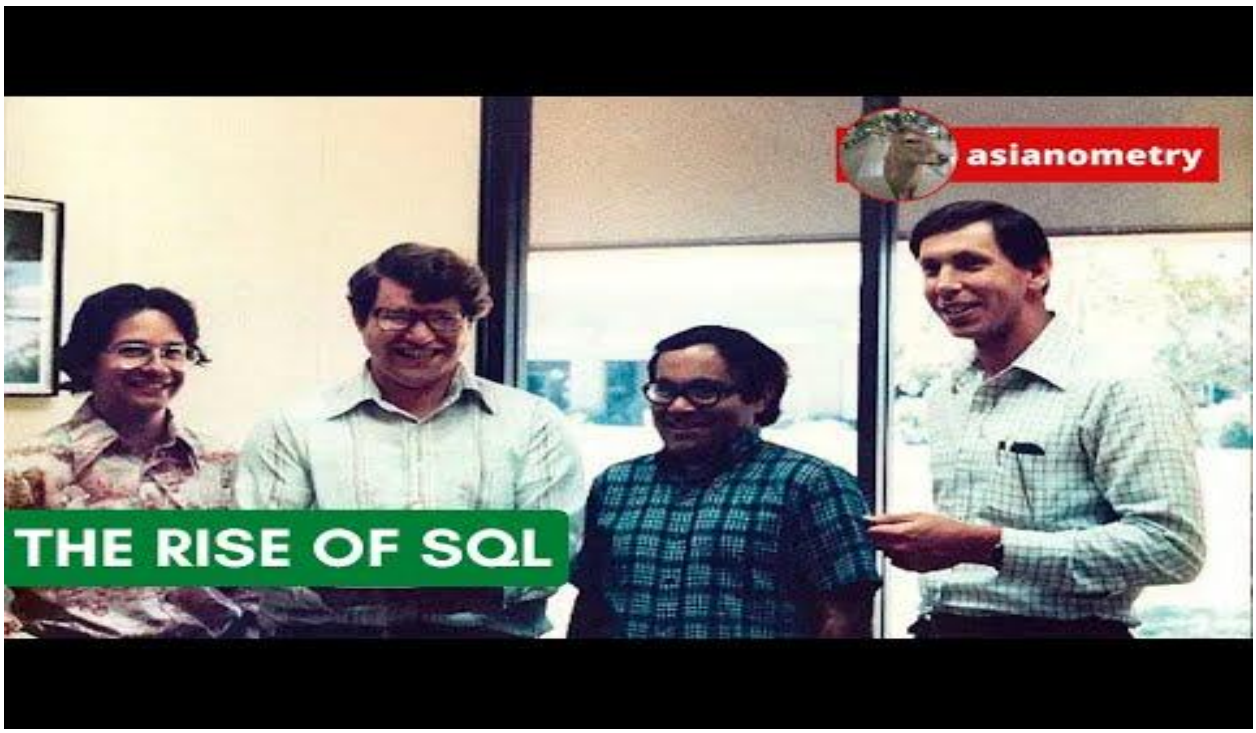
Yes, I use AI every day. This includes tools like ChatGPT and Copilot. At work, it helps me with coding, naming variables, quickly sorting data, writing emails correctly, and more. It has also become useful for creating content for social media — both text and images.

Without a doubt, ChatGPT is also helpful in everyday life — finding a recipe, helping with 5th-year homework, planning a trip, and so on.

Benefit	Description
▯ Saves Time	Completes tasks in seconds that take humans hours
▯ Increases Accuracy	Reduces human errors in repetitive tasks
▯ Improves Productivity	Supports faster output and better results
▯ Accessibility	Helps people with disabilities or language barriers
▯ Aids Learning	Offers instant help, tutoring, and explanations

Limitation	Description
▯ Not always correct	AI can **hallucinate** (give wrong info confidently)
▯ Lacks true understanding	Doesn't "think" like humans — it predicts patterns
▯ Privacy & Security	Risk of exposing sensitive data
▯ Job Displacement Risk	Some jobs may become automated
▯ Needs human judgment	Final decisions often require real-world experience

https://www.youtube.com/watch?v=zSn8iI5Mo5s&ab_channel=Asianometry



https://www.youtube.com/watch?v=btjcNSOUTOg&ab_channel=CalebCurry

https://www.programiz.com/sql/online-compiler

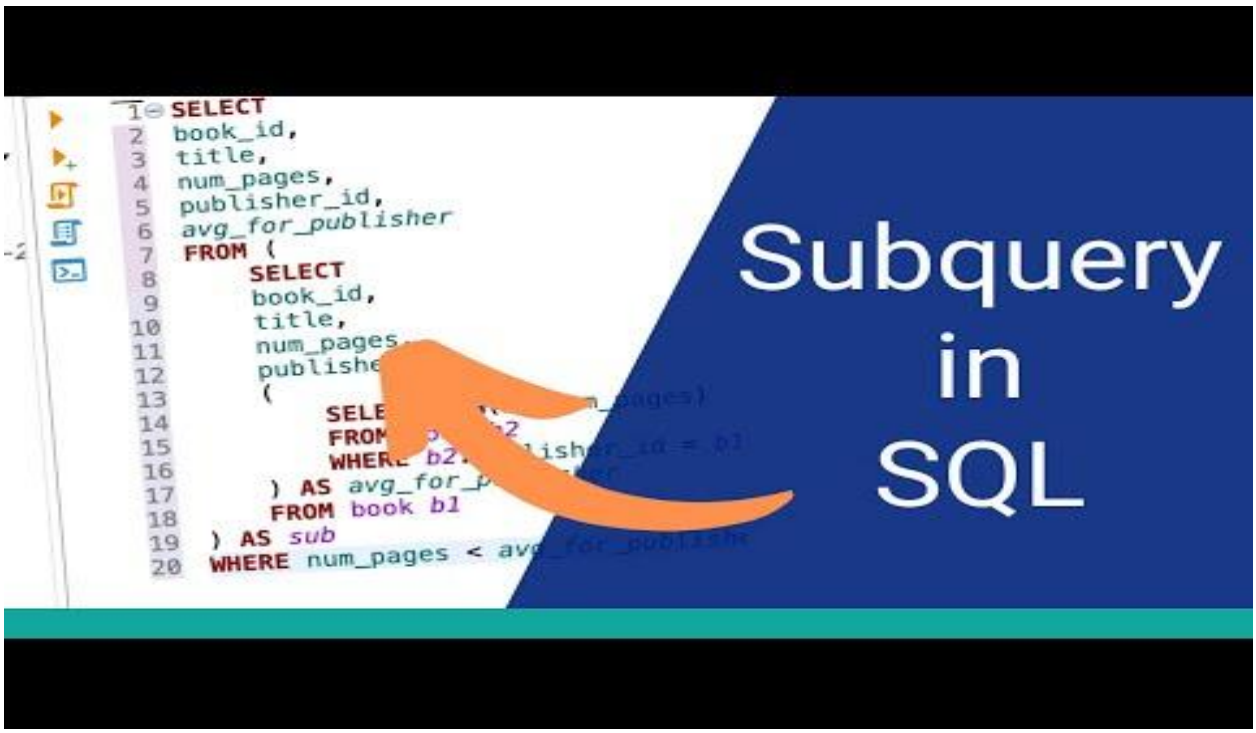
UPDATE customers SET country = 'Scotland' WHERE first_name = 'Betty' AND last_name = 'Doe';

UPDATE customers SET last_name = 'Johnson', country = 'France' WHERE customer_id = 2;

SELECT * FROM orders WHERE amount < 1000 ORDER BY amount DESC;

[MySQL Full Course for free](https://www.youtube.com/watch?v=tlvxb7UduJw&t=246s&ab_channel=DatabaseStar)

https://www.youtube.com/watch?v=tlvxb7UduJw&t=246s&ab_channel=DatabaseStar



1. Email CV tomorrow
2. Go over 5/6 jobs over weekend
3. Evaluate self (e.g. selling self, responding to interview, public speaking, self confidence). Any barriers, what is preventing from finding work basically

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.