Just IT

B2W group

Apprenticeships | Training | Recruitment

Data Technician

Name: Sergios Vasileiou

Course Date: 13/01/25

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 record in a database table.
How does this differ from a secondary key?	A primary key MUST be unique while a secondary key can be duplicated. A primary key cannot be null, whereas a secondary key can.
How are primary and foreign keys related?	A foreign key in one table points to the primary key in another table to establish a relationship. This relationship ensures that the foreign key is always associated with a valid record in the primary key table. Also, it enforces referential integrity. A foreign key must match an existing primary key value in the parent table. Deleting or updating a record in the one table may require action on the second table, such as updates or deletions.
Provide a real-world example of a one-to-one relationship	In a spotify dataset, a one-to-one example would be the trackID number and the track name. One trackID is referring to one specific track.
Provide a real-world example of a one-to-many relationship	In a hospital dataset, patients and doctors have their unique IDs. The doctor ID in relation to the patient ID is a one-to-many. From the Doctors perspective, one doctor treats many different patients.
Provide a real-world example of a many-to-many relationship	In the same hospital dataset, patients (patientID) can visit with many different doctors (doctorID). At the same time, many different doctors (doctorID) can treat many patients at the same time (patientID).

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 Databases organize data in tables (rows and columns) with a defined schema that specifies data types and relationships. They enforce ACID properties (Atomicity, Consistency, Isolation, Durability) to ensure reliable transactions and data integrity. Relationships between tables are established using foreign keys, making them ideal for applications requiring complex queries and data consistency. Examples include MySQL and Oracle Database. A common use case is customer relationship management (CRM) systems. In contrast, Non-Relational Databases (NoSQL) allow data to be stored in various formats, such as documents, wide-columns, or graphs, offering a more flexible schema for unstructured or semi-structured data. They typically focus on eventual consistency rather than strict ACID compliance. Relationships between data are usually not enforced, often resulting in denormalized data. Examples include MongoDB and Cassandra. They are commonly used in big data applications, real-time web applications, and scenarios requiring horizontal scaling and high availability.
What type of data would benefit off the non-relational model? **Why?**	A) Unstructured Data like text documents, images, videos, and social media posts. Non-relational databases can handle varying formats and do not require a fixed schema, making them ideal for unstructured data. B) Semi-Structured Data like JSON, XML, and other formats where data is organized but lacks a strict schema. They can easily accommodate changes in data structure without requiring extensive migrations. C) Big Data like Large datasets generated from sensors, logs, or user interactions. Non-relational databases are designed to scale horizontally, allowing them to handle vast amounts of data efficiently across distributed systems. D) Real-Time Data like social media feeds or online gaming data. Non-relational databases provide low-latency access and can quickly ingest high-velocity data, making them suitable for real-time applications. E) Dynamic Data like user profiles or product catalogues that change frequently. These databases allow for easy updates and modifications, accommodating the evolving nature of data without downtime.

Day 3: Task 1

Please research the below 'JOIN' types, explain what they are and provide an example (Syntax and or Scenario) of the types of data it would be used on (From Sakila dataset).

Self-join	A self-join is a regular join but the table is joined with itself. This is useful for comparing rows within the same table
Right join	A right join returns all records from the right table and the matched records from the left table. If there is no match, NULL values are returned for columns from the left table.
Full join	A full join returns all records when there is a match in either left or right table records. It combines the results of both left and right joins.
Inner join	An inner join returns only the rows that have matching values in both tables. It's the most common type of join.
Cross join	A cross join produces a Cartesian product of the two tables, meaning every row from the first table is combined with every row from the second table.
Left join	A left join returns all records from the left table and the matched records from the right table. If there is no match, NULL values are returned for columns from the right table.

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

1. Understanding the Business Requirements:

We need to consider what data we want in our table. An example of the data we would the following:

- Inventory – Product Name: Bread, Category: Bakery, Price: £0.90, Stock Level: 50.
- Sales – Customer: Jane Doe, Product: Bread, Date: 15/01/2025 Quantity: 2.
- Customer Information – Name: Jane Doe, Email: Jane.doe@gmail.com, Loyalty Points: 50.

Some examples of the type of people who would use this database:

- Store Managers – Allows them to see the sales, the number of customers and their loyalty points.
- Store workers – Allows them to check the inventory of the store.
- IT Staff – They will check that the database has no issues and fix any security problems with the database.

2. Designing the Database Schema:

a. Structuring the Database Tables:
To efficiently store inventory, sales, and customer information, the database should include three tables: Products Table, Customers Table, and Sales Table.

I. **Products Table**: Stores details of all products in inventory

- **Product ID (Primary Key)**: Unique identifier for each product.
- **Product Name**: Name of the product.
- **Category**: Category to which the product belongs.
- **Price**: Price of the product.
- **Stock Level**: Current quantity of the product in stock

Product ID	Product Name	Category	Price	Stock Level
1	Milk	Dairy	£1.50	100
2	Bread	Bakery	£0.90	50
3	Eggs	Dairy	£2.00	75
4	Apples	Fruits	£2.50	60
5	Juice	Beverages	£3.00	40

II. **Customers Table:** Stores information about customers.

- **Customer ID** (Primary Key): Unique identifier for each customer.
- **Name:** Name of the customer.
- **Email**: Contact email of the customer.
- **Loyalty Points**: Points earned by the customer as part of the loyalty program.

Customer ID	Name	Email	Loyalty Points
1	Jane Doe	jane.doe@example.com	50
2	John Smith	john.smith@example.com	30
3	Alice Brown	alice.brown@email.com	100
4	Bob White	bob.white@mail.com	70
5	Lucy Green	lucy.green@abc.com	20

III. **Sales Table:** Tracks sales transactions.

- **Sale ID** (Primary Key): Unique identifier for each sale.
- **Product ID** (Foreign Key): Links to **Product ID** in the **Products Table** to identify the product sold.
- **Customer ID** (Foreign Key): Links to **Customer ID** in the **Customers Table** to associate the sale with a customer.
- **Date**: Date of the sale.
- **Quantity**: Number of units sold in the transaction.

Sale ID	Product ID	Customer ID	Date	Quantity
1	1	1	15/01/2025	2
2	3	2	16/01/2025	2
3	2	3	16/01/2025	3
4	5	4	13/01/2025	1
5	4	5	13/01/2025	2

b. **Products and Sales**:

- **Relationship**: A sale must reference a product to track which item was sold. The **Product ID** column in the **Sales Table** serves as a foreign key that links to the **Product ID** in the **Products Table**. This ensures that the product sold exists in the inventory.

Customers and Sales:

- **Relationship**: Each sale should reference a customer, especially for loyalty programs or customer-specific analysis. The **Customer ID** column in the **Sales Table** serves as a foreign key that links to the **Customer ID** in the **Customers Table**. This ensures that each sale can be tied to a specific customer.

3. Creating the Database and Tables:

a) For us to **create a database** and the tables in SQL we need to use the create database command

"CREATE DATABASE RetailStoreDB;"

b) The we proceed to use the **create table** function, for each of our tables (, Products, Customers. and Sales).

I.E: For 'Products' Table

```
CREATE TABLE Products
(   ProductID INT PRIMARY KEY,
    ProductName VARCHAR(100),
    Category VARCHAR(50),
    Price DECIMAL(10, 2),
    StockLevel INT );
```

- For the Products table, the primary key is ProductID
- For the Customers table, the primary key is CustomerID
- For the Sales table, the primary key is SalesID

```
FOREIGN KEY (ProductID) REFERENCES Products (ProductID),
FOREIGN KEY (CustomerID) REFERENCES Customers (CustomerID)
```

c) Then, we proceed to insert all our data into the newly made tables using the **insert into** command.

I.E:

```
INSERT INTO Products (ProductID, ProductName, Category, Price, StockLevel)
VALUES
(1, 'Milk', 'Dairy', 1.50, 100),
(2, 'Bread', 'Bakery', 0.90, 50),
(3, 'Eggs', 'Dairy', 2.00, 75),
(4, 'Apples', 'Fruits', 2.50, 60),
(5, 'Juice', 'Beverages', 3.00, 40);
```

4. Populating the Database:

Once the schema is established, the next step is to populate the tables with initial data. For instance, to add a new product, we would use the following SQL statement:

- **INSERT INTO** Products (Product ID, Name, Description, Price, Stock Level) VALUES (1, 'Milk', '1L of Full Cream Milk', 1.50, 100);

Similarly, to add a new customer:

- **INSERT INTO** Customers (Customer ID, Name, Email, Phone, Loyalty Points) VALUES (1, 'John Doe', 'johndoe@example.com', '1234567890', 50);

Alternatively, most databases allow you to import data from existing files or from other spreadsheets.

5.Maintaining Database :

A) Regular Updates :

Ensure that all transactions are promptly recorded in the database to keep the data accurate and up to date.
Use the UPDATE statement to modify data based on transactions.
Example: After selling 2 bottles of milk, reduce the stock.

B) Query :

```
Update Products
Set StockLevel = StockLevel - 2
Where Productname  = 'Milk';
```

C) Monitor Data Quality :

Regularly audit the database to identify and correct any data quality issues. Implement validation rules to prevent incorrect data entry .

a. Check for duplicate entries:
Find duplicate customer records:
Query :-

```
SELECT CustomerID, COUNT(*)
FROM Customers
GROUP BY CustomerID
HAVING COUNT(*) > 1;
```

b. Check for missing data:
Find products with missing price values:
Query:-

```
SELECT *
FROM Products
WHERE Price IS NULL;
```

D) **Backups:**

Schedule regular backups to ensure that data can be restored in case of accidental loss or corruption. Store backups in a secure location.
Use SQL commands or database tools to export the database.
Example: Backup to a file in MySQL:

E) **Data Security:**

Implement Access control measures to ensure that only trusted members of staff have access to the database. Use encryption to protect sensitive data and parameterized queries to prevent SQL injection attacks.

By following these guidelines, you can ensure that your database remains accurate, up to date, and secure.

Day 4: Task 2: SQL Practical

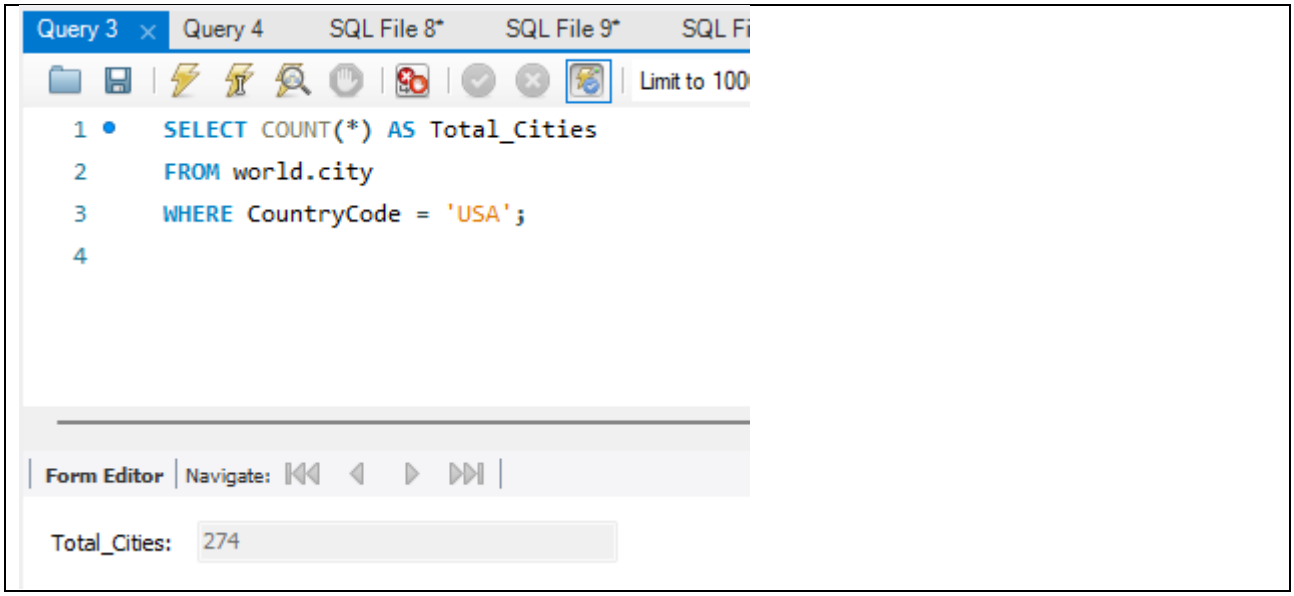
In your groups, work together to answer the below questions. It may be of benefit if one of you shares your screen with the group and as a team answer / take screen shots from there.

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.



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 with the highest life expectancy', LifeExpectancy AS 'Life Expectancy'
FROM world.country
WHERE LifeExpectancy = (SELECT MAX(LifeExpectancy) FROM world.country);
```

Form Editor | Navigate:

Country with the highest life expectancy:	Andorra
Life Expectancy:	83.5

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 Name AS 'City Names'
FROM world.city
WHERE Name LIKE 'New %';
```

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

City Names
New Bombay
New Delhi

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.

Query 3 | Query 4 | SQL File 8* | country | SQL File 9*

Limit to 1000 rows

```
SELECT Name AS 'Most populated Cities', Population AS 'Population'
From world.country
ORDER BY Population DESC
LIMIT 10 ;
```

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

Most populated Cities	Population
China	1277558000
India	1013662000
United States	278357000

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.

Query 3 | Query 4 | SQL File 8* | country | SQL File 9* | SQL File 10*

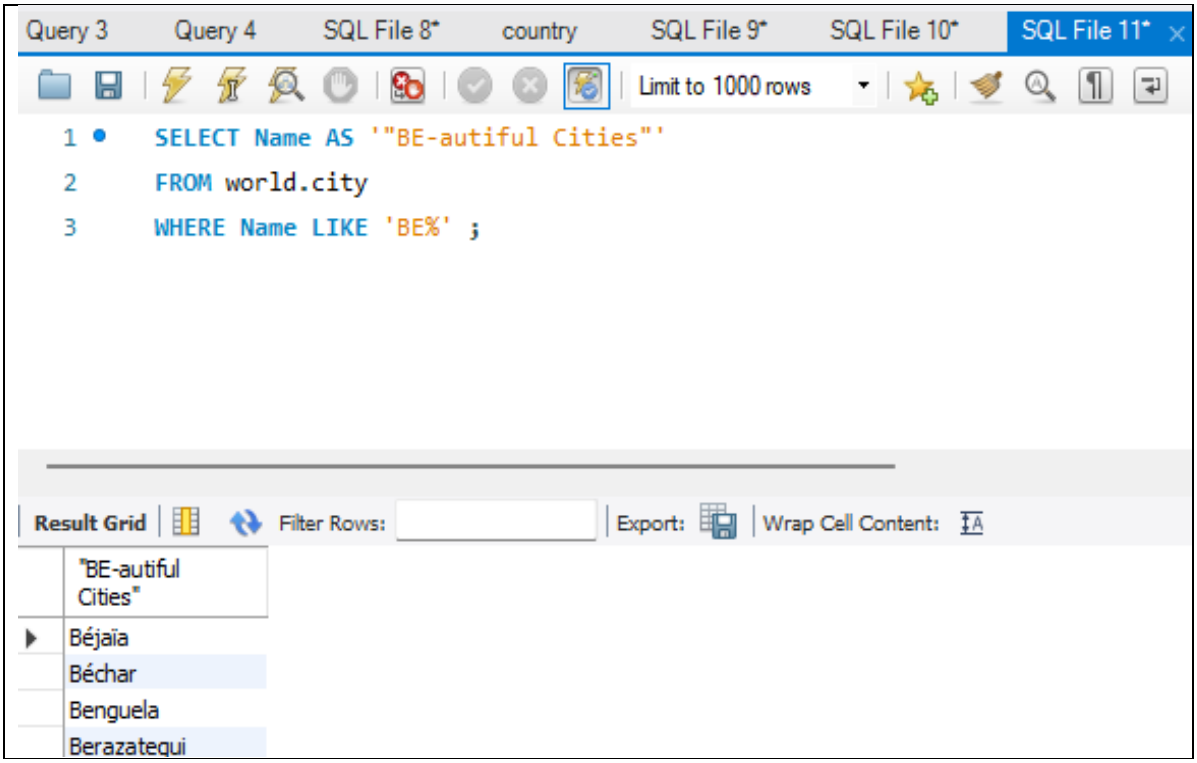
Limit to 1000 rows

```
SELECT Name AS 'Most populated Cities', Population AS 'Population'
From world.country
WHERE Population >2000000
ORDER BY Population ASC ;
```

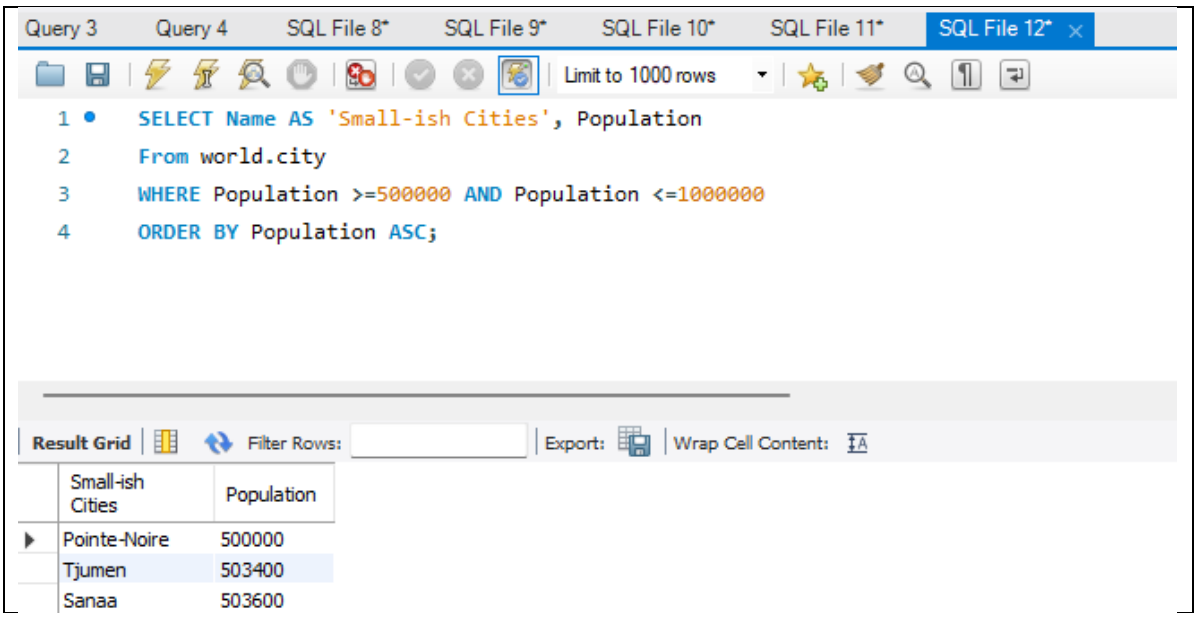
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Most populated Cities	Population
Macedonia	2024000
Bhutan	2124000
Lesotho	2153000
Latvia	2424200

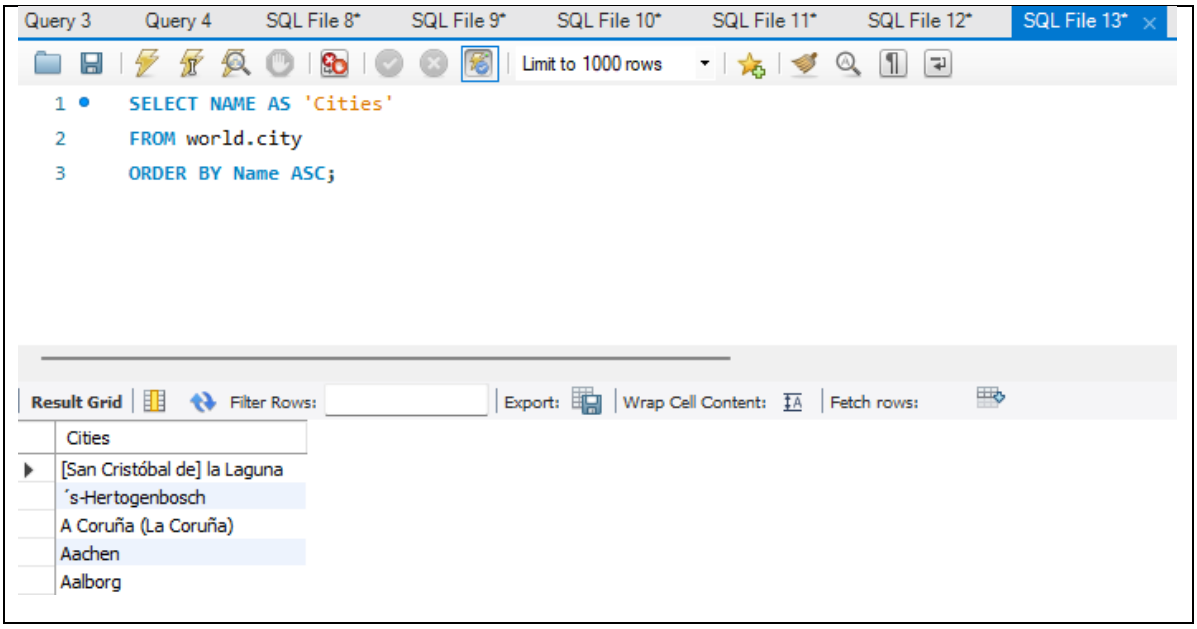
6. **Cities Beginning with 'Be' Prefix:** *Scenario:* A travel blogger is planning a series of articles featuring cities with unique names. You're tasked with compiling a list of cities from the database that start with the prefix 'Be' to assist in the blogger's content creation process.



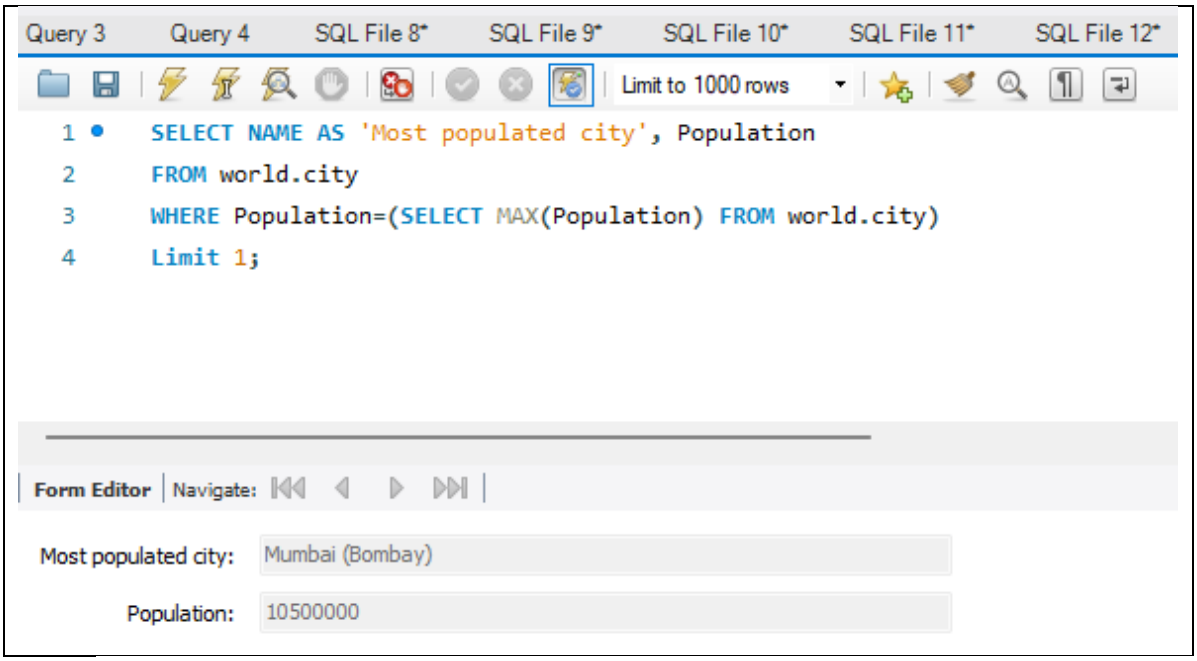
7. **Cities with Population Between 500,000-1,000,000:** *Scenario:* An urban planning committee needs to identify mid-sized cities suitable for infrastructure development projects. You're tasked with identifying cities with populations ranging between 500,000 and 1 million to inform their decision-making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

```
SELECT DISTINCT NAME AS 'Unique City Names', COUNT(*) AS 'Occurances'
FROM world.city
GROUP BY NAME
ORDER BY NAME ASC;
```

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

Unique City Names	Occurances
[San Cristóbal de] la Laguna	1
´s-Hertogenbosch	1
A Coruña (La Coruña)	1
Aachen	1

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.

Query 3 | Query 4 | SQL File 8* | SQL File 9* | SQL File 10* | SQL File 11* | SQL Fi

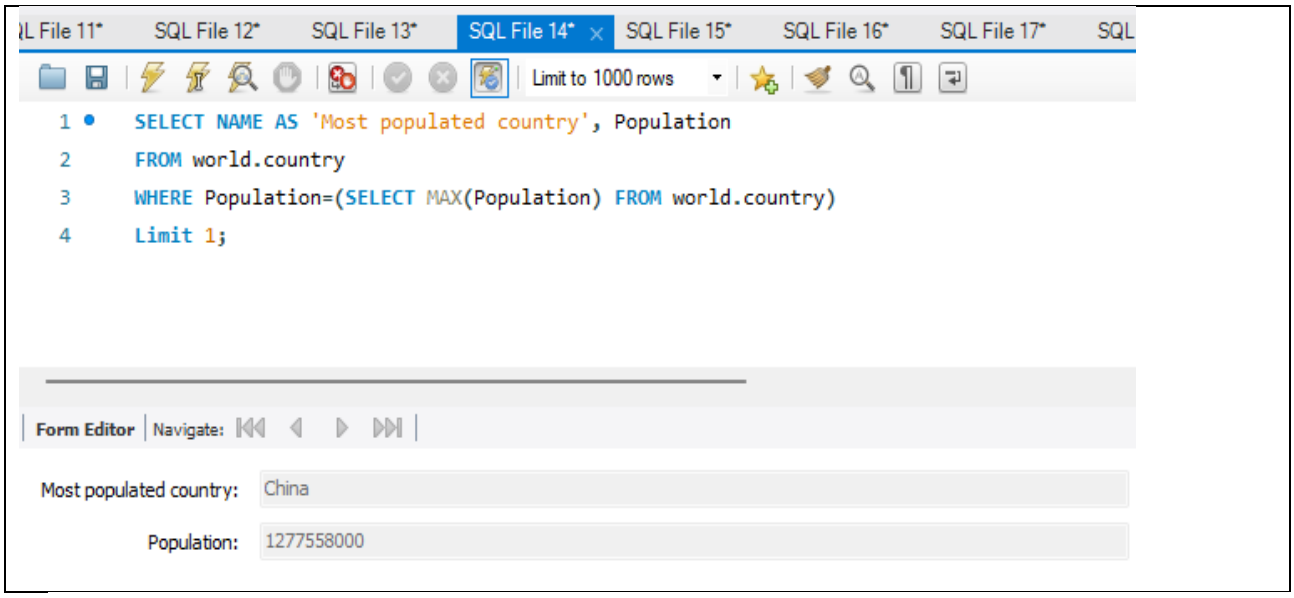
```
SELECT NAME AS 'Lowest populated city', Population
FROM world.city
WHERE Population = (SELECT MIN(Population) FROM world.city);
```

Form Editor | Navigate:

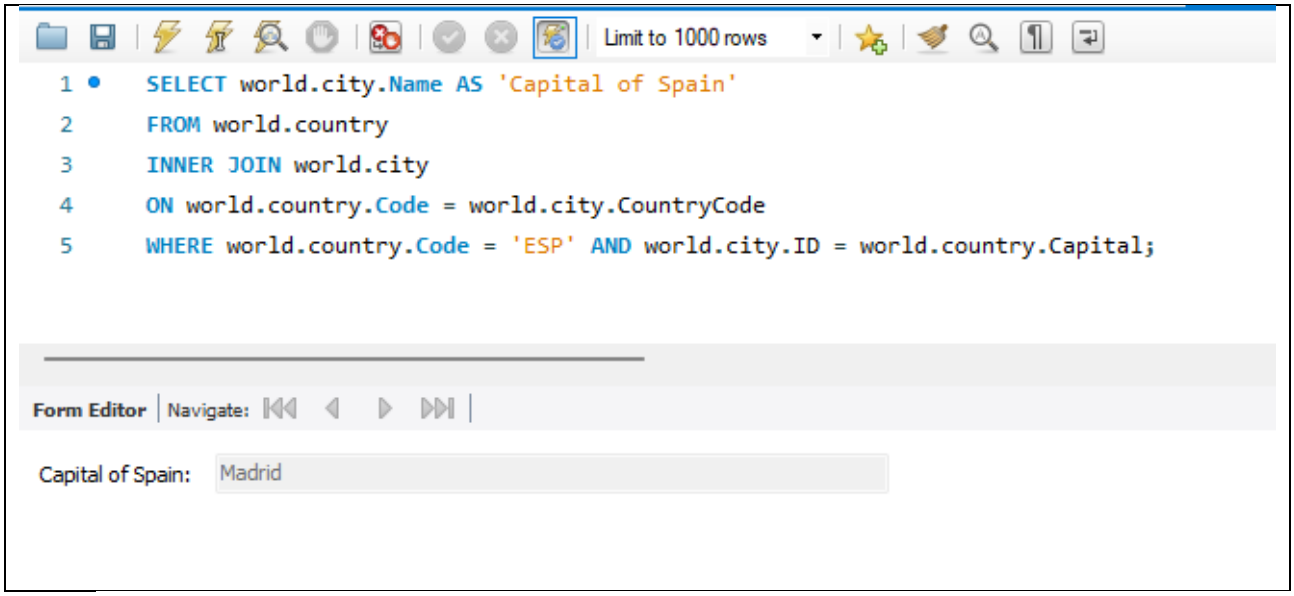
Lowest populated city: Adamstown

Population: 42

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 Highest 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.

```
SELECT NAME AS 'Country with the highest life expectancy', LifeExpectancy AS 'Life Expectancy'
FROM world.country
WHERE LifeExpectancy = (SELECT MAX(LifeExpectancy) FROM world.country);
```

Form Editor | Navigate:

Country with the highest life expectancy:	Andorra
Life Expectancy:	83.5

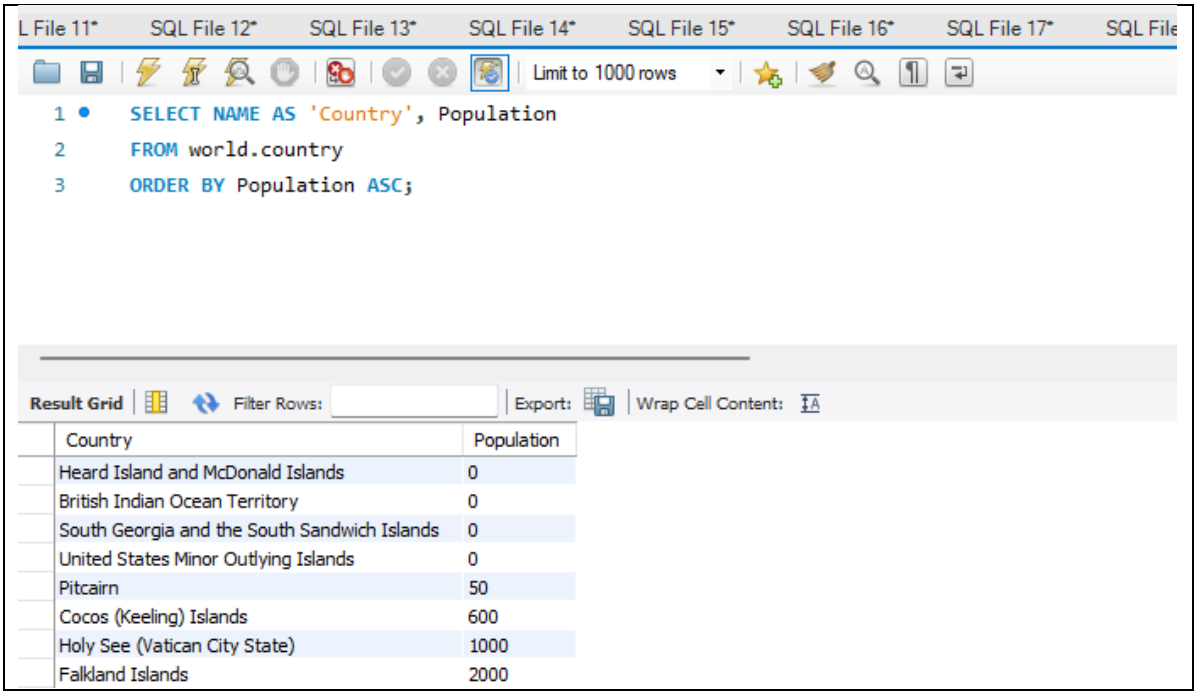
15. **Cities in Europe:** *Scenario:* A European cultural exchange program is seeking to connect students with cities across the continent. You're tasked with compiling a list of cities located in Europe from the database to facilitate program planning and student engagement.

```
SELECT world.city.NAME AS'Cities in Europe', world.country.NAME AS 'Country', world.country.NAME AS 'Continent'
FROM world.country
INNER JOIN world.city
ON world.city.CountryCode=world.country.Code
WHERE Continent='EUROPE';
```

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

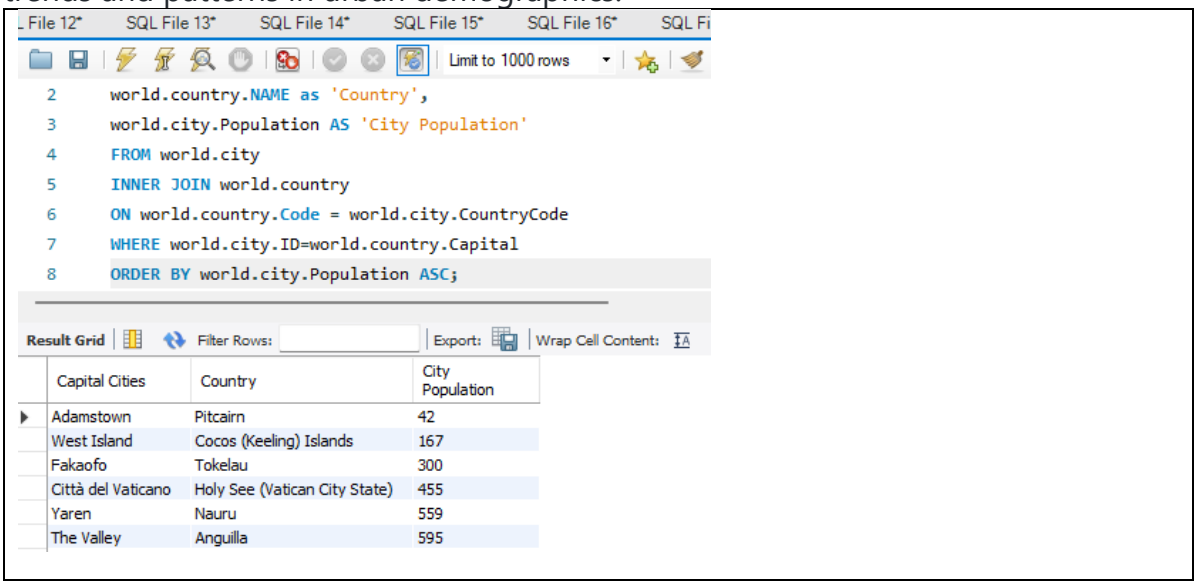
Cities in Europe	Country	Continent
Tirana	Albania	Albania
Andorra la Vella	Andorra	Andorra
Wien	Austria	Austria
Graz	Austria	Austria

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



Country	Population
Heard Island and McDonald Islands	0
British Indian Ocean Territory	0
South Georgia and the South Sandwich Islands	0
United States Minor Outlying Islands	0
Pitcairn	50
Cocos (Keeling) Islands	600
Holy See (Vatican City State)	1000
Falkland Islands	2000

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.



Capital Cities	Country	City Population
Adamstown	Pitcairn	42
West Island	Cocos (Keeling) Islands	167
Fakaofo	Tokelau	300
Città del Vaticano	Holy See (Vatican City State)	455
Yaren	Nauru	559
The Valley	Anguilla	595

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.

```
SELECT NAME AS 'Country',
Population, SurfaceArea,
(Population/SurfaceArea) AS 'Population Density'
FROM world.country;
```

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

Country	Population	SurfaceArea	Population Density
Australia	18886000	7741220.00	2.4397
Suriname	417000	163265.00	2.5541
Mauritania	2670000	1025520.00	2.6036
Iceland	279000	103000.00	2.7087
Botswana	1622000	581730.00	2.7882
Canada	31147000	9970610.00	3.1239

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.

```
FROM world.city
JOIN world.country ON world.city.CountryCode=world.country.Code
WHERE (world.country.GNP/world.city.Population) >
(SELECT AVG (world.country.GNP/world.city.Population)
FROM world.city
JOIN world.country ON world.city.CountryCode=world.country.Code
WHERE world.city.Population>0);
```

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

City Name	GDP per Capita
Gelsenkirchen	7.565695
Zalantun	7.554106
Leiyang	7.549230
Luton	7.531858

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.

SQL File 15* SQL File 16* SQL File 17* SQL File 18* SQL File 19* SQL File 20*

Limit to 1000 rows

```
SELECT world.city.Name AS 'City',
world.country.Name AS 'Country',
world.city.Population
FROM world.city
JOIN world.country ON world.city.CountryCode = world.country.Code
ORDER BY world.city.Population ASC
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

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.