

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..... 9

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?	<p>A primary key is a field (or combination of fields) that uniquely identifies each record in a table.</p> <ul style="list-style-type: none">• It must be unique and cannot be missing (i.e. NULL).• Every table should have one. <p>E.g. In a Students table, Student_ID can be the primary key – no two students can share the same ID.</p>
How does this differ from a secondary key?	<p>A secondary key (also called an alternate key) is any field (or set of fields) that can also uniquely identify a record but is not the primary key.</p> <ul style="list-style-type: none">• It is used for indexing or searching but not as the main identifier. <p>E.g. In the same Students table, Email might be a secondary key – it should also be unique, but it's not the official primary key.</p>
How are primary and foreign keys related?	<ul style="list-style-type: none">• A foreign key is a field in one table that references the primary key in another table.• It creates a link between two tables and enforces referential integrity. <p>Example:</p> <ul style="list-style-type: none">• Students.Guardian_ID is a foreign key referencing Guardians.Guardian_ID (the primary key).• This ensures every student is linked to a valid guardian.
Provide a real-world example of a one-to-one relationship	Each passport is assigned to one person, and each person can hold only one valid passport.
Provide a real-world example of a	One teacher can teach many students, but each student has only one main teacher.



one-to-many relationship	
Provide a real-world example of a many-to-many relationship	A student can enroll in many subjects, and each subject 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?	Feature	Relational DB	Non-Relational DB
	Structure	Tables (rows & columns)	Documents, key-value, graphs, wide-columns
	Schema	Fixed schema (strict)	Dynamic schema (flexible)
	Relationships	Strong use of joins and foreign keys	Usually denormalized, embeds related data
	Scalability	Vertical scaling (bigger server)	Horizontal scaling (more servers)
	Examples	MySQL, PostgreSQL, SQL Server	MongoDB, Redis, Cassandra, Neo4j
	Best For	Structured, consistent data	Semi-structured/unstructured or fast-evolving data
What type of data would benefit off the non-relational model? Why?	<p>Non-relational databases are ideal when:</p> <ol style="list-style-type: none">The data is unstructured or semi-structured Example: JSON, multimedia, logs. Relational databases require structured data with predefined columns and types. Unstructured data like images, videos, or JSON objects doesn't fit cleanly into rows and columns. Non-relational databases (like MongoDB or Couchbase) let you store documents or binary data without forcing a rigid schema.Rapid development with frequent schema changes You don't need to constantly migrate tables.		



In fast-paced projects (startups, MVPs), the data model evolves constantly — new features might require adding or removing fields.

Relational databases require schema migrations for every structural change, which can be time-consuming and error-prone.

Document databases allow you to add new fields to documents without changing a global table definition.

3. **High-speed read/write is more important than strict consistency**

Use cases like caching don't need guaranteed consistency — it's okay if some nodes are slightly behind.

Relational databases enforce strict ACID compliance (especially consistency), which can slow down performance.

Non-relational databases (like Redis or DynamoDB) offer eventual consistency, prioritizing speed and availability.

4. **Massive scale-out applications**

Like IoT data ingestion, social media feeds, or event logs. These systems generate huge volumes of data from many sources in real time.

Relational databases struggle to scale horizontally — they scale better vertically (which hits hardware limits quickly).

Non-relational systems are designed for horizontal scaling, distributing data across multiple nodes.

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>Joins a table to itself to find relationships within the same table.</p> <p>E.g. Finding employees who report to the same manager in an employee table.</p> <pre>SELECT A.name AS employee, B.name AS manager FROM Employees A JOIN Employees B ON A.manager_id = B.employee_id;</pre> $R \bowtie R = \{ (r1, r2) \mid r1, r2 \in R \text{ and } r1.key = r2.foreign_key \}$
Right outer join	<p>Returns all records from the right table, and matched records from the left table.</p> <p>E.g. Getting all customers when the customer table is on the right.</p> <pre>SELECT Orders.order_id, Customers.name FROM Orders RIGHT JOIN Customers ON Orders.customer_id = Customers.customer_id;</pre> $A \bowtie B = \{ (a, b) \mid a \in A, b \in B, a.key = b.key \} \cup \{ (NULL, b) \mid b \in B \text{ and no } a \in A \text{ such that } a.key = b.key \}$
Full outer join	<p>Returns all records from both tables, matching where possible. Fills in NULL where there is no match.</p> <p>E.g. Listing all products and orders, including products never ordered and orders with missing product info.</p> <pre>SELECT Products.name, Orders.order_id FROM Products FULL JOIN Orders ON Products.product_id = Orders.product_id;</pre> $A \bowtie B = (A \bowtie B) \cup (A \bowtie B)$
Inner join	Returns only the rows with matching values in both tables.

	<p>I.e. Intersection of 2 sets</p> <p>E.g. Finding customers who have placed orders.</p> <pre>SELECT Customers.name, Orders.order_id FROM Customers INNER JOIN Orders ON Customers.customer_id = Orders.customer_id;</pre> $A \cap B = \{ x \in A \text{ and } y \in B \mid x.\text{key} = y.\text{key} \}$
Cross join	<p>Returns the Cartesian product of both tables. Every row of A combined with every row of B.</p> <p>E.g. Generating every possible combination of colours and sizes for a product.</p> <pre>SELECT Colours.name, Sizes.size FROM Colours CROSS JOIN Sizes;</pre> $A \times B = \{ (a, b) \mid a \in A, b \in B \}$
Left outer join	<p>Returns all records from the left table and matched records from the right.</p> <p>E.g. Listing all customers and any orders they may have, including customers without orders.</p> <pre>SELECT Customers.name, Orders.order_id FROM Customers LEFT JOIN Orders ON Customers.customer_id = Orders.customer_id;</pre> $A \bowtie B = \{ (a, b) \mid a \in A, b \in B, a.\text{key} = b.\text{key} \} \cup \{ (a, \text{NULL}) \mid a \in A \text{ and no } b \in B \text{ such that } a.\text{key} = b.\text{key} \}$

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

The data stored will include Inventory data (product names, categories, quantities, suppliers, and prices), Sales data (transaction dates, items sold, quantities, customer and totals), Customer info (names, contact details, and loyalty program participation) & Loyalty program info (points earned, redeemed, and membership status)

The primary users will be store's staff and manager. Staff need to add and retrieve sales and inventory data quickly, while the manager needs reports on sales trends, inventory levels, and customer engagement.

To structure the database tables efficiently, I would create the tables:

- Products – Stores all product info
- Customers – Stores customer profiles and loyalty data
- Sales – Records each sale transaction
- Sale_Items – Detailed item descriptions in each sale
- Loyalty – Tracks the points, status and redemption history of customers

One Sale is linked to One Customer

One Sale can include Many Sale_Items, which are tied to One Product

One Customer is linked to One Loyalty profile

E.g. Creating the tables:

```
CREATE TABLE Products (  
  product_id SERIAL PRIMARY KEY,  
  name VARCHAR(100),  
  category VARCHAR(50),  
  price DECIMAL(10,2),  
  stock_quantity INT  
);
```

```
CREATE TABLE Customers (  
  customer_id SERIAL PRIMARY KEY,  
  name VARCHAR(100),
```



```
email VARCHAR(100),
phone VARCHAR(20)
);

CREATE TABLE Loyalty (
  customer_id INT PRIMARY KEY REFERENCES
Customers(customer_id),
  points INT DEFAULT 0
);

CREATE TABLE Sales (
  sale_id SERIAL PRIMARY KEY,
  customer_id INT REFERENCES Customers(customer_id),
  sale_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  total_amount DECIMAL(10,2)
);

CREATE TABLE SaleItems (
  sale_item_id SERIAL PRIMARY KEY,
  sale_id INT REFERENCES Sales(sale_id),
  product_id INT REFERENCES Products(product_id),
  quantity INT,
  item_price DECIMAL(10,2)
);

E.g. Inputting initial records:

INSERT INTO Products (name, category, price, stock_quantity)
VALUES ('Milk', 'Dairy', 1.99, 50);

INSERT INTO Customers (name, email, phone)
VALUES ('John Smith', 'john98@example.com', '0121 555 5555');

INSERT INTO Loyalty (customer_id, points)
VALUES (1, 20);

INSERT INTO Sales (customer_id, total_amount)
VALUES (1, 5.97);
```

```
INSERT INTO SaleItems (sale_id, product_id, quantity, item_price)
VALUES (1, 1, 3, 1.99);
```

To ensure the database remains accurate and up to date, I would add constraints and validation, i.e. using NOT NULL, UNIQUE, CHECK, etc. to prevent the wrong data inputted

E.g. To prevent using negative values for prices:

```
ALTER TABLE Products
ADD CONSTRAINT check_stock_positive CHECK (stock_quantity >= 0);
```

I would also add automated triggers – which can update stock levels or loyalty points when a sale is recorded – and regularly running reports to detect anomalies. E.g. products with stock but no sales, or duplicate customer emails.

To handle backups and data security, I would set up daily backups of the entire database – either hourly (incremental), daily (full) or weekly (off-site) – and test the restore process to ensure the backups work as intended. Also, keeping track with version control and changelogs (e.g. of schema changes and SQL scripts) can help look back at previous queries if something goes wrong.

For data security, user roles and permissions can be created for cashiers and administrators:

```
CREATE ROLE cashier NOINHERIT;
GRANT SELECT, INSERT ON Sales, SaleItems TO cashier;
```

Aswell as encrypting/masking sensitive data (E.g. customer data) to limit access, and regularly updating the database system with security patches.

Day 4: Task 2: SQL Practical

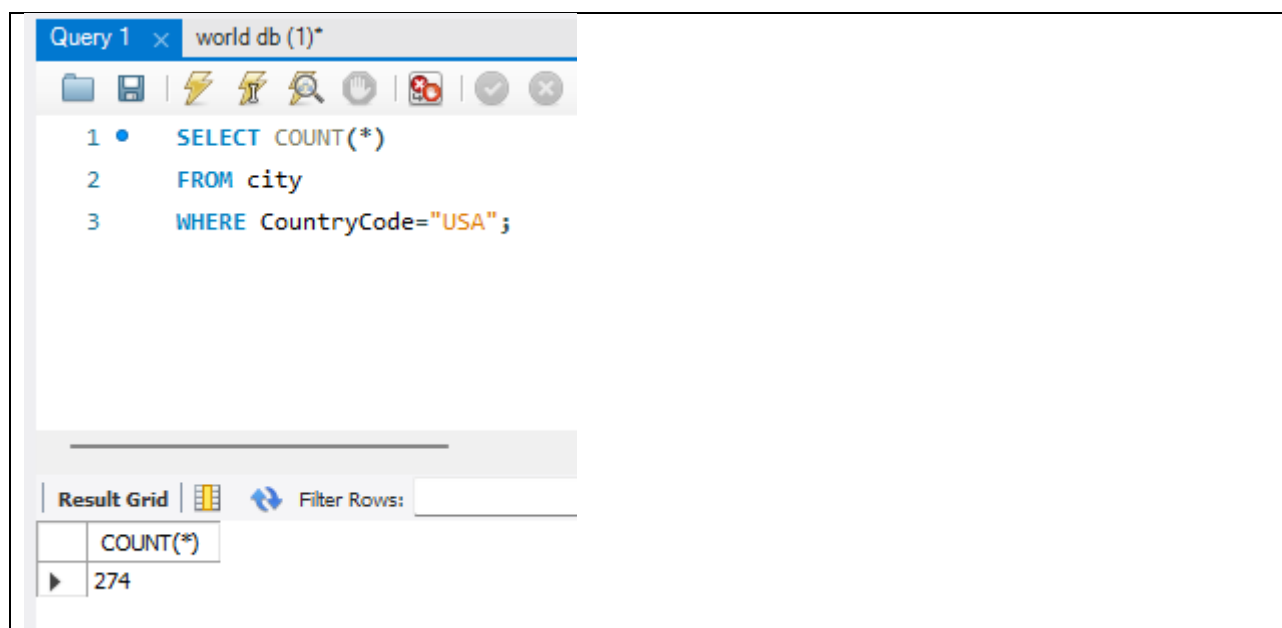
In your groups, work together to answer the below questions. It may be of benefit if one of you shares your screen with the group and as a team answer / take screen shots from there.

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.



The screenshot shows a SQL query editor window titled "Query 1" and "world db (1)*". The query is as follows:

```
1 • SELECT COUNT(*)
2 FROM city
3 WHERE CountryCode="USA";
```

Below the query editor, the "Result Grid" is displayed, showing the output of the query:

COUNT(*)
274

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.

Query 1 x world db (1)*

Limit to 1000 rows

```

1 • SELECT *
2 FROM city
3 WHERE Name LIKE '%New%';

```

Result Grid Filter Rows: Edit: Export/Import:

	ID	Name	CountryCode	District	Population
▶	137	Newcastle	AUS	New South Wales	270324
	482	Newcastle upon Tyne	GBR	England	189150
	502	Newport	GBR	Wales	139000
	734	Newcastle	ZAF	KwaZulu-Natal	222993
	936	Kowloon and New Kowloon	HKG	Kowloon and New Kowl	1987996
	1106	New Bombay	IND	Maharashtra	307297
	1109	New Delhi	IND	Delhi	301297
	2857	Khanewal	PAK	Punjab	133000
	3793	New York	USA	New York	8008278
	3823	New Orleans	USA	Louisiana	484674
	3855	Newark	USA	New Jersey	273546
	3905	Newport News	USA	Virginia	180150
	3971	New Haven	USA	Connecticut	123626
	4044	New Bedford	USA	Massachusetts	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.

Query 1 x world db (1)*

Limit to 1000 rows

```

1 • SELECT *
2 FROM city
3 ORDER BY Population DESC
4 LIMIT 10;

```

Result Grid Filter Rows: Edit:

	ID	Name	CountryCode	District	Population
▶	1024	Mumbai (Bombay)	IND	Maharashtra	10500000
	2331	Seoul	KOR	Seoul	9981619
	206	São Paulo	BRA	São Paulo	9968485
	1890	Shanghai	CHN	Shanghai	9696300
	939	Jakarta	IDN	Jakarta Raya	9604900
	2822	Karachi	PAK	Sindh	9269265
	3357	Istanbul	TUR	Istanbul	8787958
	2515	Ciudad de México	MEX	Distrito Federal	8591309
	3580	Moscow	RUS	Moscow (City)	8389200
	3793	New York	USA	New York	8008278
	NULL	NULL	NULL	NULL	NULL

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 1 × world db (1)*

Limit to 1000 rows

1 • SELECT *

2 FROM city

3 WHERE Population > 2000000;

4

Result Grid

Filter Rows:

Edit:

	ID	Name	CountryCode	District	Population
▶	35	Alger	DZA	Alger	2168000
	56	Luanda	AGO	Luanda	2022000
	69	Buenos Aires	ARG	Distrito Federal	2982146
	130	Sydney	AUS	New South Wales	3276207
	131	Melbourne	AUS	Victoria	2865329
	150	Dhaka	BGD	Dhaka	3612850
	206	São Paulo	BRA	São Paulo	9968485
	207	Rio de Janeiro	BRA	Rio de Janeiro	5598953
	208	Salvador	BRA	Bahia	2302832
	209	Belo Horizonte	BRA	Minas Gerais	2139125
	210	Fortaleza	BRA	Ceará	2097757
	456	London	GBR	England	7285000
	554	Santiago de ...	CHL	Santiago	4703954
	502	Quito	ECU	Quito	2070040

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.

Query 1 x world db (1)*

Limit to 1000 rows

```

1 • SELECT *
2 FROM city
3 WHERE Name LIKE 'Be%';

```

Result Grid Filter Rows: Edit:

	ID	Name	CountryCode	District	Population
▶	45	Béjaïa	DZA	Béjaïa	117162
	49	Béchar	DZA	Béchar	107311
	59	Benguela	AGO	Benguela	128300
	93	Berazategui	ARG	Buenos Aires	276916
	184	Belize City	BLZ	Belize City	55810
	185	Belmopan	BLZ	Cayo	7105
	209	Belo Horizonte	BRA	Minas Gerais	2139125
	216	Belém	BRA	Pará	1186926
	246	Belford Roxo	BRA	Rio de Janeiro	425194
	266	Betim	BRA	Minas Gerais	302108
	453	Bento Gonçalves	BRA	Rio Grande d...	89254
	469	Belfast	GBR	North Ireland	287500
	724	Benoni	ZAF	Gauteng	365467
	840	Belozi	TUN	West Tunis	644300

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.

Query 1 x world db (1)*

Limit to 1000 rows

```

1 • SELECT *
2 FROM city
3 WHERE Population BETWEEN 500000 AND 1000000;

```

Result Grid Filter Rows: Edit:

	ID	Name	CountryCode	District	Population
▶	5	Amsterdam	NLD	Noord-Holland	731200
	6	Rotterdam	NLD	Zuid-Holland	593321
	36	Oran	DZA	Oran	609823
	64	Dubai	ARE	Dubai	669181
	72	Rosario	ARG	Santa Fé	907718
	73	Lomas de Zamora	ARG	Buenos Aires	622013
	74	Quilmes	ARG	Buenos Aires	559249
	75	Almirante Brown	ARG	Buenos Aires	538918
	76	La Plata	ARG	Buenos Aires	521936
	77	Mar del Plata	ARG	Buenos Aires	512880
	134	Adelaide	AUS	South Australia	978100
	152	Khulna	BGD	Khulna	663340
	186	Cotonou	BEN	Atlantique	536827
	102	Santa Cruz de la	BOL	Santa Cruz	925351

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.

Query 1 x world db (1)*

Limit to 1000 rows

```

1 • SELECT *
2 FROM city
3 ORDER BY Name ASC;











```

Result Grid Filter Rows: Edit: Export/Import:



	ID	Name	CountryCode	District	Population
▶	698	[San Cristóbal de] la Laguna	ESP	Canary Islands	127945
	20	's-Hertogenbosch	NLD	Noord-Brabant	129170
	670	A Coruña (La Coruña)	ESP	Galicia	243402
	3097	Aachen	DEU	Nordrhein-Westfalen	243825
	3318	Aalborg	DNK	Nordjylland	161161
	2760	Aba	NGA	Imo & Abia	298900
	1404	Abadan	IRN	Khuzestan	206073
	395	Abaetetuba	BRA	Pará	111258
	3683	Abakan	RUS	Hakassia	169200
	1849	Abbotsford	CAN	British Colombia	105403
	2747	Abeokuta	NGA	Ogun	427400
	478	Aberdeen	GBR	Scotland	213070
	3191	Abha	SAU	Asir	112300
	2812	Abidjan	CIV	Abidjan	2500000

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.

Query 1 x world db (1)*



```
1 • SELECT Name, COUNT(*) AS Count
2 FROM city
3 GROUP BY Name
4 ORDER BY Name ASC;
```

Result Grid   Filter Rows:

	Name	Count
▶	[San Cristóbal de] la Laguna	1
	's-Hertogenbosch	1
	A Coruña (La Coruña)	1
	Aachen	1
	Aalborg	1
	Aba	1
	Abadan	1
	Abaetetuba	1
	Abakan	1
	Abbotsford	1
	Abeokuta	1
	Aberdeen	1
	Abha	1
	Abidjan	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 1 x world db (1)*

1 • SELECT Name, Population
2 FROM city
3 ORDER BY Population ASC
4 LIMIT 1;
5

Result Grid Filter Rows:

	Name	Population
▶	Adamstown	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.

Query 1 x world db (1)*

1 • SELECT Name, Population
2 FROM country
3 ORDER BY Population DESC
4 LIMIT 1;

Result Grid Filter Rows:

	Name	Population
▶	China	1277558000

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.

Query 1 x world db (1)*

```

1 • SELECT *
2 FROM city
3 WHERE CountryCode="ESP"
4 ORDER BY Population DESC
5 LIMIT 1;

```

Result Grid | Filter Rows: | Edit:

	ID	Name	CountryCode	District	Population
▶	653	Madrid	ESP	Madrid	2879052
*	NULL	NULL	NULL	NULL	NULL

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.

Query 1 x world db (1)*

```

1 • SELECT *
2 FROM city
3 JOIN country ON city.CountryCode = country.Code
4 WHERE country.Continent="Europe";

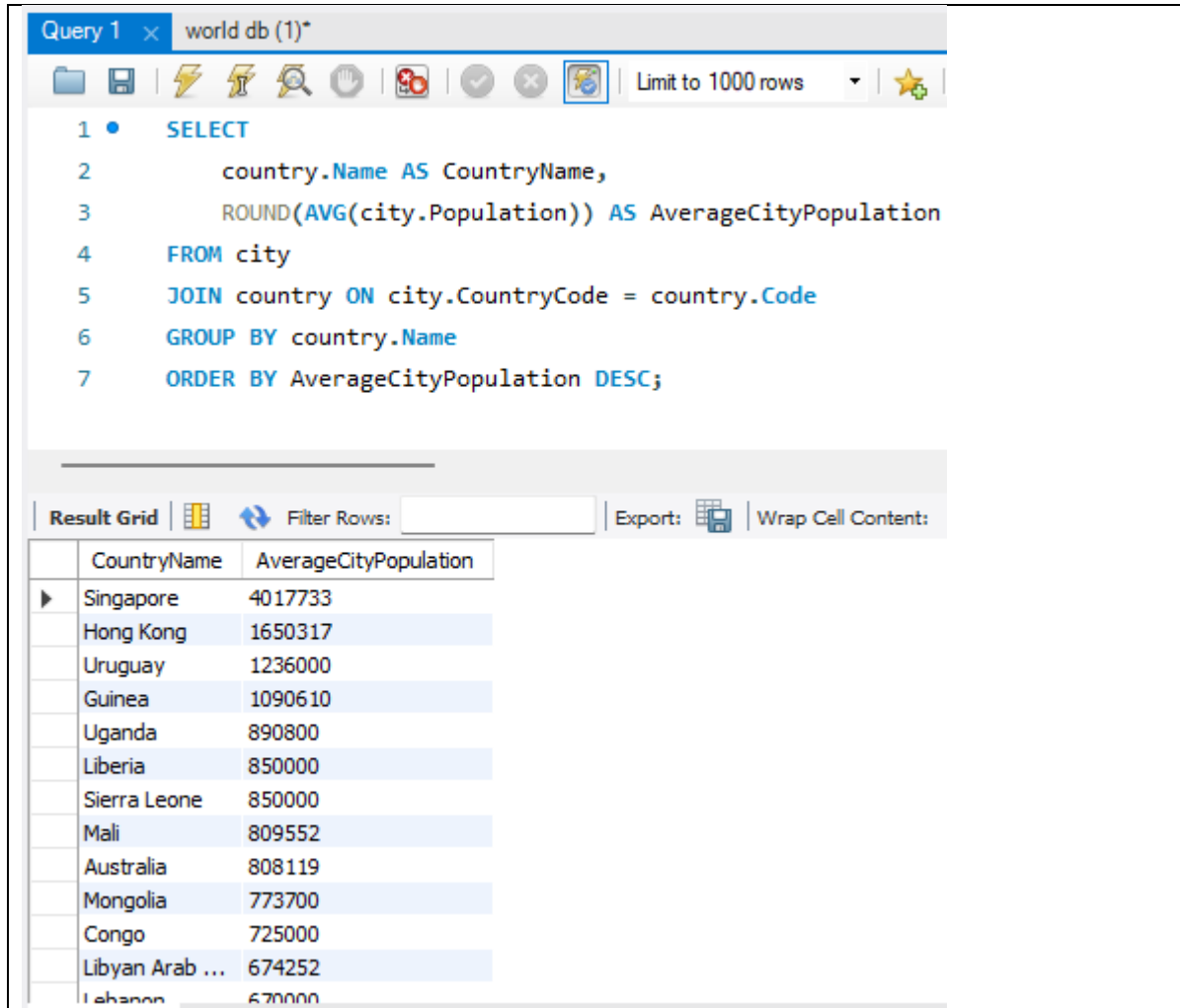
```

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

	ID	Name	CountryCode	District	Population	Code	Name	Continent	Region
▶	34	Tirana	ALB	Tirana	270000	ALB	Albania	Europe	South
	55	Andorra la Vella	AND	Andorra la Vella	21189	AND	Andorra	Europe	South
	1523	Wien	AUT	Wien	1608144	AUT	Austria	Europe	West
	1524	Graz	AUT	Steiermark	240967	AUT	Austria	Europe	West
	1525	Linz	AUT	North Austria	188022	AUT	Austria	Europe	West
	1526	Salzburg	AUT	Salzburg	144247	AUT	Austria	Europe	West
	1527	Innsbruck	AUT	Tirol	111752	AUT	Austria	Europe	West
	1528	Klagenfurt	AUT	Kärnten	91141	AUT	Austria	Europe	West
	175	Antwerpen	BEL	Antwerpen	446525	BEL	Belgium	Europe	West
	176	Gent	BEL	East Flandre	224180	BEL	Belgium	Europe	West
	177	Charleroi	BEL	Hainaut	200827	BEL	Belgium	Europe	West
	178	Lüttich	BEL	Lüttich	195630	BEL	Belgium	Europe	West



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.



The screenshot displays a database query editor window titled "Query 1" and "world db (1)". The SQL query is as follows:

```
1 • SELECT
2     country.Name AS CountryName,
3     ROUND(AVG(city.Population)) AS AverageCityPopulation
4 FROM city
5 JOIN country ON city.CountryCode = country.Code
6 GROUP BY country.Name
7 ORDER BY AverageCityPopulation DESC;
```

Below the query editor, the "Result Grid" shows the output of the query, sorted by average city population in descending order. The grid includes columns for "CountryName" and "AverageCityPopulation".

CountryName	AverageCityPopulation
Singapore	4017733
Hong Kong	1650317
Uruguay	1236000
Guinea	1090610
Uganda	890800
Liberia	850000
Sierra Leone	850000
Mali	809552
Australia	808119
Mongolia	773700
Congo	725000
Libyan Arab ...	674252
Lebanon	670000

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.

Query 1 x world db (1)*

Limit to 1000 rows

```

1 • SELECT country.Name AS Country, city.Name AS City, city.Population
2   FROM country
3  LEFT JOIN city
4   ON country.Capital = city.ID
5  ORDER BY city.Population DESC;

```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: [IA](#)

	Country	City	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

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.

Query 1
world db (1)*

Limit to 1000 rows

```

1  SELECT
2      Name,
3      Population,
4      SurfaceArea,
5      ROUND(Population / SurfaceArea, 2) AS PopulationDensity
6  FROM country
7  WHERE (Population / SurfaceArea) < 50
8  ORDER BY PopulationDensity ASC;
9

```

Result Grid

Filter Rows:
Export:

Wrap Cell Content:

	Name	Population	SurfaceArea	PopulationDensity
▶	French Southern territories	0	7780.00	0.00
	Antarctica	0	13120000.00	0.00
	Heard Island and McDonald Islands	0	359.00	0.00
	Bouvet Island	0	59.00	0.00
	British Indian Ocean Territory	0	78.00	0.00
	South Georgia and the South Sandwich Islands	0	3903.00	0.00
	United States Minor Outlying Islands	0	16.00	0.00
	Greenland	56000	2166090.00	0.03
	Svalbard and Jan Mayen	3200	62422.00	0.05

	Name	Population	SurfaceArea	PopulationDensity
►	French Southern territories	0	7780.00	0.00
	Antarctica	0	13120000.00	0.00
	Heard Island and McDonald Islands	0	359.00	0.00
	Bouvet Island	0	59.00	0.00
	British Indian Ocean Territory	0	78.00	0.00
	South Georgia and the South Sandwich Islands	0	3903.00	0.00
	United States Minor Outlying Islands	0	16.00	0.00
	Greenland	56000	2166090.00	0.03
	Svalbard and Jan Mayen	3200	62422.00	0.05

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.

Query 1 x world db (1)*

Limit to 1000 rows

```

1 WITH AvgGDP AS (
2     SELECT AVG(GNP / Population) AS AvgGDPPerCapita
3     FROM country
4     WHERE GNP IS NOT NULL AND Population > 0
5 )
6 SELECT
7     city.Name AS City,
8     country.Name AS Country,
9     city.Population,
10    ROUND(country.GNP / country.Population, 2) AS GDPPerCapita
11 FROM city
12 JOIN country ON city.CountryCode = country.Code
13 JOIN AvgGDP ON 1=1
14 WHERE country.GNP IS NOT NULL
15     AND country.Population > 0
16     AND (country.GNP / country.Population) > AvgGDP.AvgGDPPerCapita
17 ORDER BY GDPPerCapita DESC, city.Population DESC;

```

Result Grid

Filter Rows:

Export:

Wrap Cell Content:

Fetch rows:

	City	Country	Population	GDPPerCapita
▶	Zürich	Switzerland	336800	0.04
	Geneve	Switzerland	173500	0.04
	Basel	Switzerland	166700	0.04
	Bern	Switzerland	122700	0.04
	Lausanne	Switzerland	114500	0.04
	Luxembourg [Luxemburg/Lëtzebuerg]	Luxembourg	80700	0.04
	Bandar Seri Begawan	Brunei	21484	0.04
	Saint George	Bermuda	1800	0.04
	Hamilton	Bermuda	1200	0.04

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.

The screenshot shows a database query editor interface. At the top, the title bar reads "Query 1" and "world db (1)". Below the title bar is a toolbar with various icons for file operations, execution, and editing. The main area displays a SQL query:

```
1 SELECT *
2 FROM city
3 ORDER BY Population DESC
4 LIMIT 10 OFFSET 30;
```

Below the query editor, the "Result Grid" tab is active, showing a table of results. The table has six columns: ID, Name, CountryCode, District, and Population. The results are sorted by Population in descending order, starting from the 31st row (offset 30).

ID	Name	CountryCode	District	Population
1896	Shenyang	CHN	Liaoning	4265200
1897	Kanton [Guangzhou]	CHN	Guangdong	4256300
3208	Singapore	SGP	-	4017733
3769	Ho Chi Minh City	VNM	Ho Chi Minh City	3980000
1027	Chennai (Madras)	IND	Tamil Nadu	3841396

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.

