Just IT

B2W group

Apprenticeships | Training | Recruitment

Data Technician

Name: Thandeka Mukazi

Course Date: 27th May – 30th May

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?	Primary Key = can be described as a column or combination of columns within a database table that uniquely identifies each row in that table.
How does this differ from a secondary key?	A secondary key can also uniquely identify rows but it is an alternate identifier. It is not chosen as the primary key. It can be used for look ups or queries like primary keys but do have the same strict constraints (e.g. you can have multiple secondary keys in a table while there is usually only one primary key per table).
How are primary and foreign keys related?	A primary key uniquely identifies records within one table. The foreign key essentially creates a connection to that primary key from a different table.
Provide a real-world example of a one-to-one relationship	A real world example of a one to one relationship could be people and passports. Each person has one unique passport – each passport belongs to only one person.
Provide a real-world example of a one-to-many relationship	A real world example of a one to many relationship could be form mentors and secondary school students. One form mentor can have many students in their group – each student only has one form mentor.
Provide a real-world example of a many-to-many relationship	A real world example of a many to many relationship could be students and subject courses. One student can choose to enrol in many courses – each course can have many students enrolled onto it.

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?	Key difference between a relational and non-relational database is how they store and manage data. Relational databases are best for structured data (e.g. banking systems(. Stores data in tables which includes rows and columns. On the other hand, non relational databases stores data in flexible formats so is ideal for unstructured or semi-structured data (e.g. documents)
What type of data would benefit off the non-relational model? **Why?**	Generally, non-relational databases are best suited for data that is unstructured, semi structured, rapidly changing and doesn't fit into a fixed table schema very well. For example, e-commerce product catalogues. Products tend to have very different attributes (e.g. a dress vs a phone). Non-relational databases are able to store variable fields feasibly without having to redesign the entire database schema.

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 a join where each row of a table is joined to other rows in the same table. It is useful for comparing rows within the same table. For example, to find relationships or hierarchies within one data set.

```
SELECT
    e1.EmployeeID AS Employee,
    e1.Name AS EmployeeName,
    e2.Name AS ManagerName
FROM Employees e1
JOIN Employees e2 ON e1.ManagerID = e2.EmployeeID;
```

Example of a Self-Join:

	- The Employees table has a ManagerID column - Each employee may report to another employee (i.e. the manager) - The table is joined to itself to find the name of the manager for each employee

Right join	A right join (aka right outer join) returns all records from the right table and the matched records from the left table. If no match is found, the result is null on the left side.
	
	`SELECT`
	`  Orders.OrderID,`
	`  Customers.CustomerName`
	`FROM Orders`
	`RIGHT JOIN Customers ON Orders.CustomerID = Customers.CustomerID;`
	
	Example of a Right Join:
	- Returns all rows from the right table (Customers)
	- And matched rows from the left table (Orders)
	- Showing NULL for orders if a customer has made no orders
	- Useful to see all customers, including those who haven't placed an order

Full join	A full join (aka a full outer join) combines the results of both a left join and a right join. It returns all rows from both tables with matched rows where available and null where there is no match.
	
	`SELECT`
	`  Customers.CustomerName,`
	`  Orders.OrderID`
	`FROM Customers`
	`FULL JOIN Orders ON Customers.CustomerID = Orders.CustomerID;`
	
	Example of a Full Join:
	- Returns all records from both Customers and Orders tables
	- Matching rows where possible
	- And null in places where there is no match
	- You would see customers with and without orders
	- Orders with and without a matching customer

Inner join	An inner join returns only the rows that have matching values in both tables, hinged upon a specified condition. Deemed to be the most common type of join and is utilised when combining related data that exists in both tables.
	`SELECT`
	`  Customers.CustomerName,`
	`  Orders.OrderID`
	`FROM Customers`
	`INNER JOIN Orders ON Customers.CustomerID = Orders.CustomerID;`
	
	Example of an Inner Join:
	- Returns only the records where there is a match in both tables
	- If a customer has placed an order, you'll see the customer and order data
	- Customers with no orders or orders with no customer match are excluded

Cross join	A cross join returns the cartesian products of two tables. This means every row from the first table is combined with every row from the second table.
	`SELECT`
	`  Employees.EmployeeName,`
	`  Products.ProductName`
	`FROM Employees`
	`CROSS JOIN Products;`
	
	Example of a Cross Join:
	- Every row from the first table is combined with every row from the second table
	- If there are 5 employees and 10 products, the result will be 50 rows

Left join	A left join (aka a left outer join) returns all rows from the left table and the matching from the right table. If there is no match, the result is Null for the right table's columns. Example of a Left Join:

```sql
SELECT
    Customers.CustomerName,
    Orders.OrderID
FROM Customers
LEFT JOIN Orders ON Customers.CustomerID = Orders.CustomerID;
```

Example of a Left Join:

- Returns all records from the left table (Customers)
- And matched records from the right table (Orders)
- With Null if no match exists on the right side
- You would see all customers, even those who have not made any orders

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

Understanding the business requirements

The following 5 types of data in the database would be stored: Products, Sales, Sales Details, Customer Information, and Loyalty Accounts . The users of this database could include shop managers and shop assistants. Shop managers may need to monitor sales performance; sales assistants may need to check product prices and availability. Overall, these steps are key in creating a functional and efficient database by providing focus to the metrics that are most important to the retail business.

Designing the database schema

The following table structures would be used: Products Table (Product ID, Product Name, Category, Stock Availability, Price), Customers Table (Customer ID, First Name, Last Name, Email, Phone Number), Loyalty Accounts Table (LoyaltyID CustomerID, Points, Tier), Sales Table (SalesID, CustomerID, Date of Transaction, Total) and Sales Detail Table (SaleDetailID, SalesID, ProductID, Quantity, Price). Three key relationships can be established between Customer and Sales (1 to many - one customer can make multiple purchases but each sale is linked to one customer), Sales and Sales Detail (1 to many - one sale could involve multiple individual items but each item links to one sale) and Customer and Loyalty Account 1 to 1 - each customer has one loyalty account and each account belongs to one customer). Overall, these steps are key in creating a functional and efficient database for the retail business by establishing a blueprint of the relevant tables and relationships that underpin the key metrics.

Implementing the database

To create the retail database and the associated tables, the 5 following SQL commands would be used: CREATE DATABASE, USE, CREATE TABLE, PRIMARY KEY and FOREIGN KEY. One example of an SQL statement could look like this (/ = new line): SALES - CREATE TABLE Sales/ SalesID INT PRIMARY KEY,/CustomerID INT,/Dateoftransaction DATE,/Total DECIMAL (10,2),/FOREIGN KEY(CustomerID) REFERENCES Customers(CustomerID);. Overall, these steps are key in creating a functional and efficient database

for the retail business by providing a feasible format to explore the key metrics and their relationships more seamlessly.

Populating the database

This is an SQL INSERT statement to demonstrate how initial data would be put: CUSTOMERS - INSERT INTO Customers(CustomerID, FirstName, LastName, Email, Phone Number)/VALUES(1, 'Toby', 'Elis', 'toby.elis@email.co.uk', '07549124598');. Overall, these steps are key in creating a functional and efficient database for the retail business as it enables the business to update and modify as is needed.

Maintaining the database

To ensure that the retail database remains accurate and up to date, enabling triggers for automatic updates is one measure. By utilising the CREATE TRIGGER function, specific fields such as stock levels can be automatically updated when a sale is recorded. When handling backups, regular scheduled backups is one measure. The retail database could be automatically updated on a daily or weekly basis, which could depend on the transaction volume. With data security, restricted access could be considered so that the database has different levels of access depending on user role. Overall, these steps are key in creating a functional and efficient database for the retail business ensuring that the database remains consistent and protected overtime.

Word count: 499/500 (not including subheadings)

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.

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

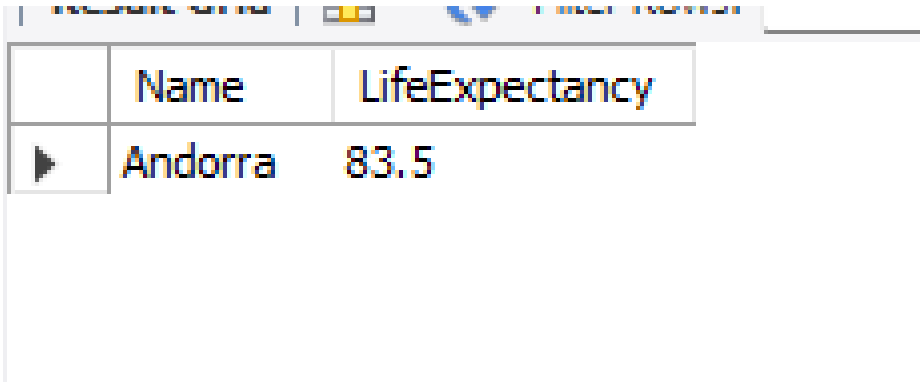
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

COUNT(CountryCode)
274

The total number of cities within the United States is 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.

```sql
SELECT Name, LifeExpectancy
FROM country
ORDER BY LifeExpectancy DESC
LIMIT 1;
```



Name	LifeExpectancy
Andorra	83.5

Andorra is the country with the highest life expectancy.

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.

```sql
SELECT(Name)
FROM city
WHERE Name LIKE 'New%';
```

Name
Newcastle
Newcastle upon Tyne
Newport
Newcastle
New Bombay
New Delhi
New York
New Orleans
Newark
Newport News
New Haven
New Bedford

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.

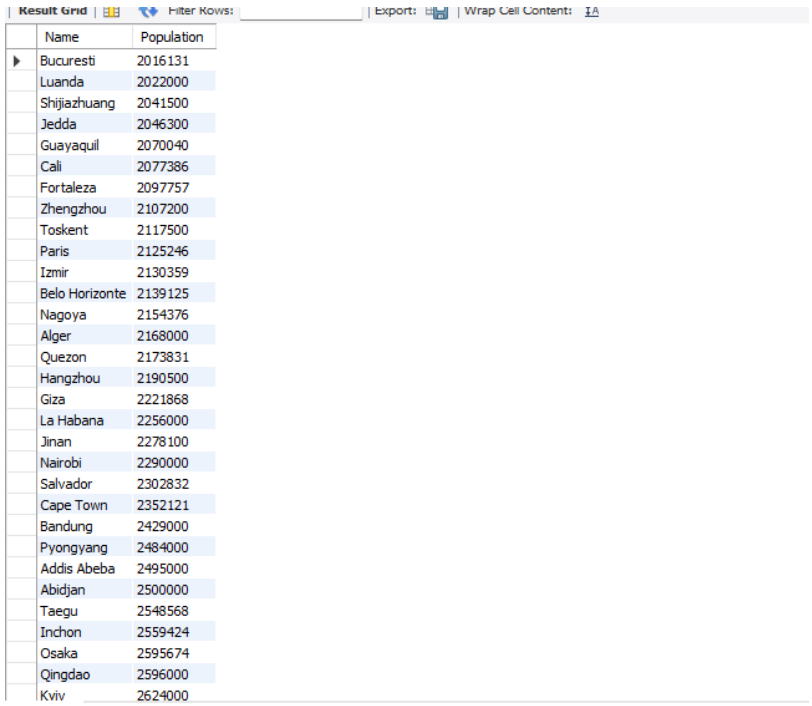
```
1    SELECT Name,Population
2    FROM city
3    ORDER BY Population DESC
4    LIMIT 10;
```

Result Grid | Filter Rows: | Export:

Name	Population
Mumbai (Bombay)	10500000
Seoul	9981619
São Paulo	9968485
Shanghai	9696300
Jakarta	9604900
Karachi	9269265
Istanbul	8787958
Ciudad de México	8591309
Moscow	8389200
New York	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.

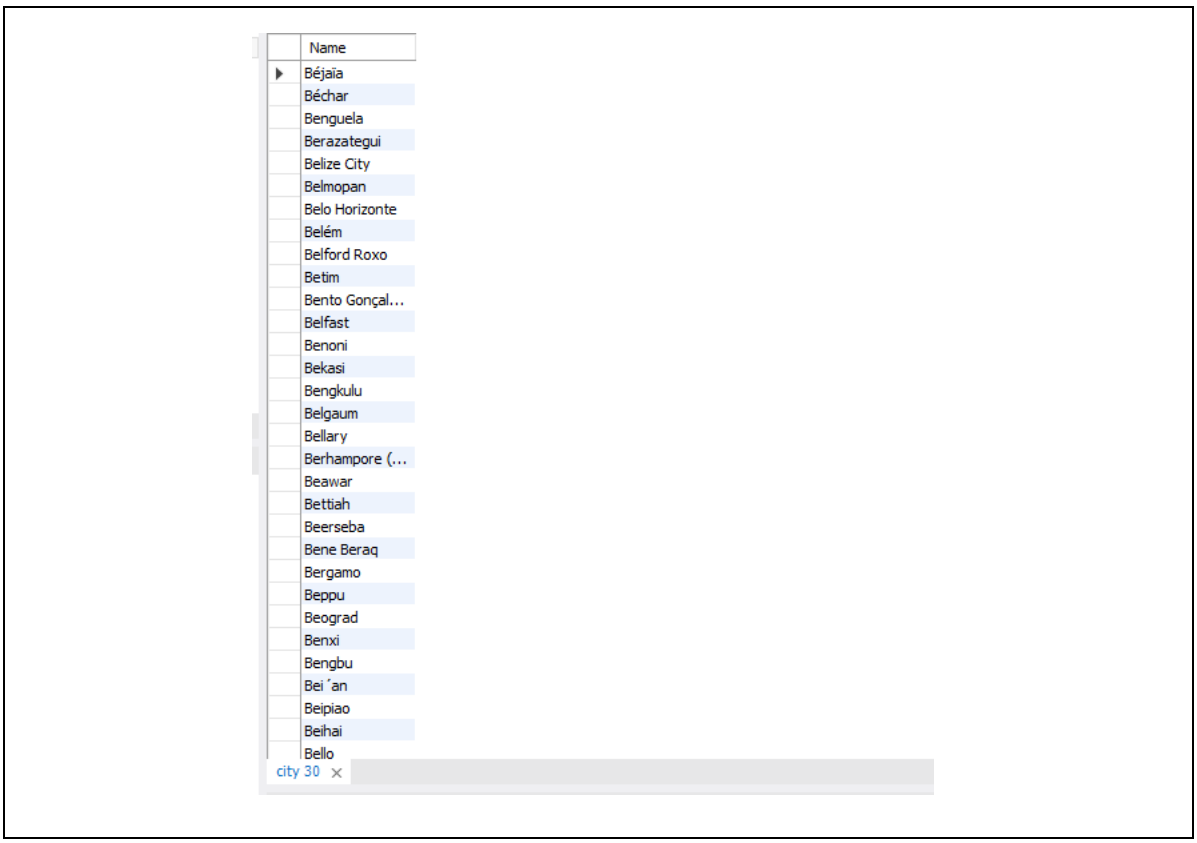
```sql
SELECT Name,Population
FROM city
WHERE Population > 2000000
ORDER BY Population ASC;
```



Name	Population
Bucuresti	2016131
Luanda	2022000
Shijiazhuang	2041500
Jedda	2046300
Guayaquil	2070040
Cali	2077386
Fortaleza	2097757
Zhengzhou	2107200
Toskent	2117500
Paris	2125246
Izmir	2130359
Belo Horizonte	2139125
Nagoya	2154376
Alger	2168000
Quezon	2173831
Hangzhou	2190500
Giza	2221868
La Habana	2256000
Jinan	2278100
Nairobi	2290000
Salvador	2302832
Cape Town	2352121
Bandung	2429000
Pyongyang	2484000
Addis Abeba	2495000
Abidjan	2500000
Taegu	2548568
Inchon	2559424
Osaka	2595674
Qingdao	2596000
Kyiv	2624000

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.

```sql
SELECT Name
FROM city
WHERE Name LIKE 'Be%';
```



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.

```
1    SELECT Name,Population
2    FROM city
3    WHERE Population BETWEEN 500000 AND 1000000
4    ORDER BY Population ASC;
```

Name	Population
Pointe-Noire	500000
Tjumen	503400
Sanaa	503600
Chandigarh	504094
Salé	504420
Pasig	505058
Gorakhpur	505566
Tula	506100
Oklahoma City	506132
Hims	507404
Mykolajiv	508000
Oslo	508726
Pohang	508899
Jalandhar (Ju...	509510
Kaifeng	510000
Ansan	510314
Mar del Plata	512880
Hachioji	513451
Vancouver	514008
Nabereznyje ...	514700
Hannover	514718
São José dos ...	515553
Bucaramanga	515555
Chiclayo	517000
Pimpri-Chinch...	517083
Higashiosaka	517785
Koyang	518282
Surakarta	518600
Duisburg	519793
Centro (Villah...	519873
Dandong	520000

city 32 ×

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.

```
1    SELECT Name
2    From city
3    ORDER BY Name ASC;
```

Name
[San Cristóbal de] la Laguna
´s-Hertogenbosch
A Coruña (La Coruña)
Aachen
Aalborg
Aba
Abadan
Abaetetuba
Abakan
Abbotsford
Abeokuta
Aberdeen
Abha
Abidjan
Abiko
Abilene
Abohar
Abottabad
Abu Dhabi
Abuja

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.

```
1    SELECT Name, Population
2    From city
3    ORDER BY Population DESC
4    LIMIT 1;
```

	Name	Population
▶	Mumbai (Bombay)	10500000

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.

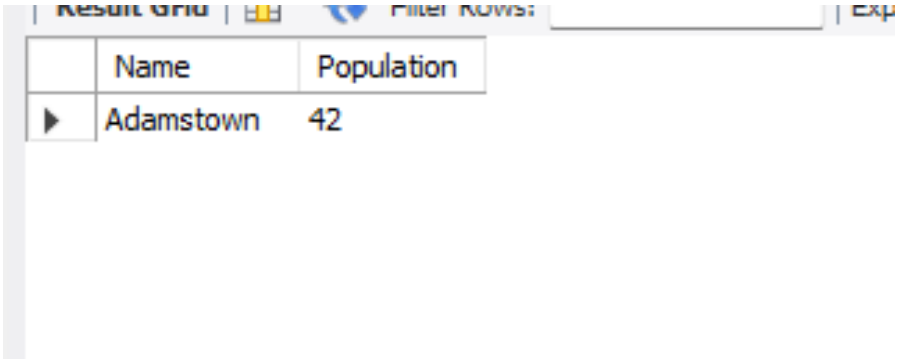
```
1 •   SELECT Name, Count(*) AS Frequencies
2     FROM city
3     GROUP BY Name
4     ORDER BY Frequencies DESC;
```

	Name	Frequencies
▶	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
	Jining	2
	Kansas City	2
	Ede	2
	Mérida	2
	Santa Clara	2
	Saint John´s	2
	Glendale	2
	Anyang	2
	San Juan	2
	Matamoros	2

Result 13 ×

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.

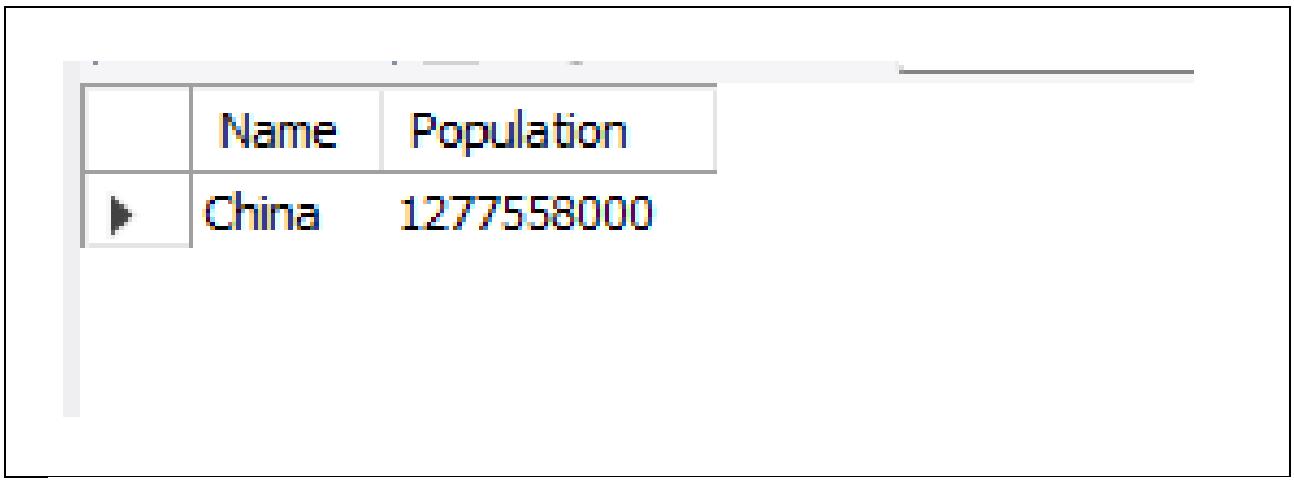
```
1 •   SELECT Name, Population
2     FROM city
3     ORDER BY Population ASC
4     LIMIT 1;
```



Name	Population
Adamstown	42

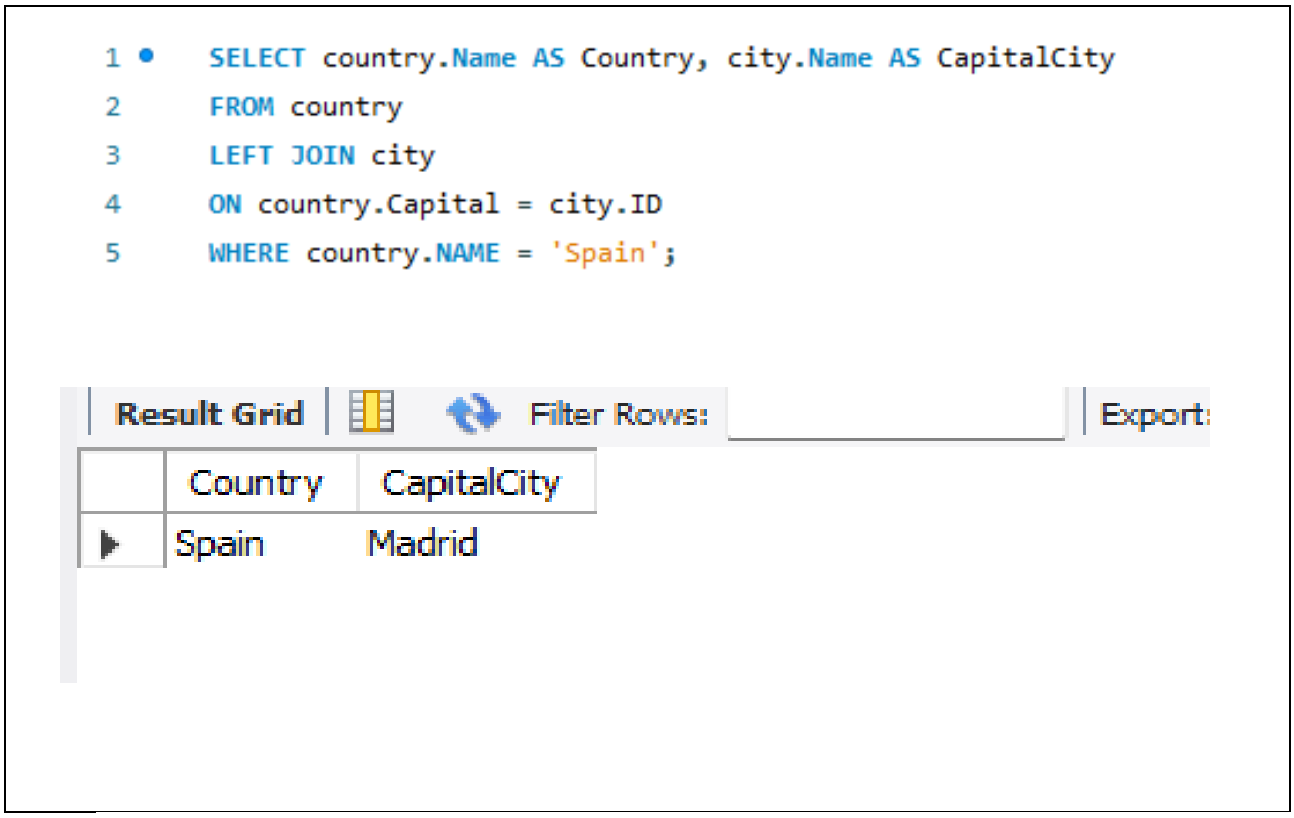
12. **Country with Largest Population:** *Scenario:* A global economic research institute requires data on countries with the largest populations for a comprehensive analysis. You're tasked with identifying the country with the highest population from the database to provide valuable insights into demographic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

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



	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.



```
1 •  SELECT country.Name AS Country, city.Name AS CapitalCity
2    FROM country
3    LEFT JOIN city
4    ON country.Capital = city.ID
5    WHERE country.NAME = 'Spain';
```

Result Grid | Filter Rows: | Export:

	Country	CapitalCity
▶	Spain	Madrid

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.

```sql
SELECT city.Name AS City, country.NAME AS Country
FROM city
JOIN Country ON City.CountryCode = Country.Code
WHERE Country.Continent = 'Europe';
```

City	Country
Tirana	Albania
Andorra la Vella	Andorra
Wien	Austria
Graz	Austria
Linz	Austria
Salzburg	Austria
Innsbruck	Austria
Klagenfurt	Austria
Antwerpen	Belgium
Gent	Belgium
Charleroi	Belgium
Liège	Belgium
Bruxelles [Brus...	Belgium
Brugge	Belgium
Schaerbeek	Belgium
Namur	Belgium
Mons	Belgium
Sofija	Bulgaria
Plovdiv	Bulgaria
Varna	Bulgaria
Burgas	Bulgaria
Ruse	Bulgaria
Stara Zagora	Bulgaria
Pleven	Bulgaria
Sliven	Bulgaria

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.

```sql
SELECT country.Name AS Country_Name, AVG(city.Population) AS Average_City_Population
FROM city
JOIN country ON city.CountryCode = country.Code
GROUP BY Country_Name
ORDER BY Average_City_Population DESC;
```

Country_Name	Average_City_Population
Singapore	4017733.0000
Hong Kong	1650316.5000
Uruguay	1236000.0000
Guinea	1090610.0000
Uganda	890800.0000
Liberia	850000.0000
Sierra Leone	850000.0000
Mali	809552.0000
Australia	808119.0000
Mongolia	773700.0000
Congo	725000.0000
Libyan Arab J...	674251.7500
Lebanon	670000.0000
Thailand	662763.4167
Côte d'Ivoire	638227.4000
Azerbaijan	616000.0000
Iraq	595069.4000
Afghanistan	583025.0000
South Korea	557141.3286
Peru	552147.3636
Congo, The D...	548034.1667
Armenia	544366.6667
Egypt	542785.9189
Pakistan	534690.5932
Colombia	532920.7895

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.

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

Country	Capital_City	Capital_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
Peru	Lima	6464693
Thailand	Bangkok	6320174
Colombia	Santafé de Bogotá	6260862
Congo, The Demo...	Kinshasa	5064000
Chile	Santiago de Chile	4703954
Iraq	Baghdad	4336000
Singapore	Singapore	4017733
Bangladesh	Dhaka	3612850
Germany	Berlin	3386667
Myanmar	Rangoon (Yangon)	3361700
Saudi Arabia	Riyadh	3324000
Turkey	Ankara	3038159
Argentina	Buenos Aires	2982146
Spain	Madrid	2879052
Italy	Roma	2643581
Taiwan	Taipei	2641312

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.

```
1  SELECT Name, Population, SurfaceArea,
2  Population/SurfaceArea AS Population_Density
3  FROM country
4  ORDER BY Population_Density ASC;
```

Name	Population	SurfaceArea	Population_Density
Antarctica	0	13120000.00	0.0000
French Southern territories	0	7780.00	0.0000
Bouvet Island	0	59.00	0.0000
Heard Island and McDonald Islands	0	359.00	0.0000
British Indian Ocean Territory	0	78.00	0.0000
South Georgia and the South Sandwich Islands	0	3903.00	0.0000
United States Minor Outlying Islands	0	16.00	0.0000
Greenland	56000	2166090.00	0.0259
Svalbard and Jan Mayen	3200	62422.00	0.0513
Falkland Islands	2000	12173.00	0.1643
Pitcairn	50	49.00	1.0204
Western Sahara	293000	266000.00	1.1015
Mongolia	2662000	1566500.00	1.6993
French Guiana	181000	90000.00	2.0111
Namibia	1726000	824292.00	2.0939
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
Libyan Arab Jamahiriya	5605000	1759540.00	3.1855
Guyana	861000	214969.00	4.0052
Gabon	1226000	267668.00	4.5803
Central African Republic	3615000	622984.00	5.8027

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.

```
1  SELECT city.name AS City_Name, country.Name AS Country_Name, country.GNP/country.Population AS GDP_PER_CAPITA
2  FROM city
3  JOIN country ON city.CountryCode = Country.Code
4  WHERE country.GNP IS NOT NULL AND
5  country.Population > 0 AND(Country.GNP/Country.Population)> (
6  SELECT AVG(GNP/Population)
7  FROM country
8  WHERE GNP IS NOT NULL AND Population > 0)
9  ORDER BY GDP_PER_CAPITA DESC;
```

City_Name	Country_Name	GDP_PER_CAPITA
Luxembourg [Luxemburg/Lëtzebuerg]	Luxembourg	0.037459
Zürich	Switzerland	0.036936
Geneve	Switzerland	0.036936
Basel	Switzerland	0.036936
Bern	Switzerland	0.036936
Lausanne	Switzerland	0.036936
Saint George	Bermuda	0.035815
Hamilton	Bermuda	0.035815
Bandar Seri Begawan	Brunei	0.035686
Schaan	Liechtenstein	0.034644
Vaduz	Liechtenstein	0.034644
George Town	Cayman Islands	0.033237
København	Denmark	0.032664
Århus	Denmark	0.032664
Odense	Denmark	0.032664
Aalborg	Denmark	0.032664
Frederiksberg	Denmark	0.032664
Oslo	Norway	0.032577
Bergen	Norway	0.032577
Trondheim	Norway	0.032577
Stavanger	Norway	0.032577

Result 18

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.

```
1  SELECT city.Name AS City_Name,
2         city.Population,
3         country.Name AS Country_Name
4  FROM city
5  JOIN country ON city.CountryCode = country.Code
6  ORDER BY city.Population DESC
7  LIMIT 10 OFFSET 30;
```

Result Grid | Filter Rows: | Export: | Wrap

City_Name	Population	Country_Name
Shenyang	4265200	China
Kanton [Guangzhou]	4256300	China
Singapore	4017733	Singapore
Ho Chi Minh City	3980000	Vietnam
Chennai (Madras)	3841396	India
Pusan	3804522	South Korea
Los Angeles	3694820	United States
Dhaka	3612850	Bangladesh
Berlin	3386667	Germany
Rangoon (Yangon)	3361700	Myanmar

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:

Introduction to Relational Databases - YouTube

Relational Database Relationships (Updated)

The Birth of SQL & the Relational Database

MySQL 24 - Important Data Types

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

SQL ORDER BY Keyword

(14) SQL JOINS Tutorial for beginners | Practice SQL Queries using JOINS - Part 1 - YouTube

```
INSERT INTO employees (name, role)
VALUES ('John Doe', 'Manager');

UPDATE employees
SET role = 'Senior Manager'
WHERE name = 'John Doe';

DELETE FROM employees
WHERE name = 'John Doe';

ALTER TABLE employees
ADD COLUMN department VARCHAR(50);

CREATE TABLE departments (id INT, name VARCHAR(50));

DROP TABLE departments

ALTER TABLE employees
ADD CONSTRAINT pk_employee_id PRIMARY KEY (id);

SELECT * FROM Studentscoring ORDER BY student DESC

SELECT * FROM Studentscoring ORDER BY student ASC
```

```
SELECT MIN (sCORE)

FROM Studentscoring


SELECT MIN (Score) AS Lowest

FROM student scoring


SELECT SUM

SELECT AVG

SELECT COUNT


SELECT CustomerID, CustomerName

FROM Customers

WHERE CustomerID IN (SELECT CustomerID FROM Orders WHERE OrderDate >
'2023-01-01');


SELECT Customers.customer_id

FROM Customers

INNER JOIN Orders ON Customers.customer_id = Orders.customer_id;

This query will return the customer_idvalues of customers who have placed
orders.


SELECT customer_id, SUM(amount) AS TotalSales

FROM Orders

GROUP BY customer_id;
```

```
SELECT status AS shipping_id, COUNT(*) AS order_count
FROM Shippings
GROUP BY status
ORDER BY order_count DESC
;
```

Understanding the business requirements

1a) What kind of data will the database need to store?

- Products = stores information about the products that are for sale
- Sales = Record of each transaction
- Sales Details = Record of items in each sale
- Customer Information = Holds customer contact information and links to loyalty program
- Loyalty Accounts = tracks points and tier status for each customer

1b) Who will be the users of the database, and what will they need to accomplish?

Shop managers = would use it to oversee day to day business operations; making decisions based on sales and inventory data. To accomplish:

- Monitor overall sales performance
- Tracking which products sell well and those that don't
- Identifying when to restock items

Shop/Stock assistants = would use to input sales at the point of sale and recording customer purchases that could be linked to the loyalty programme. To accomplish:

- Quickly processing transactions
- To check product prices and availability
- Applying loyalty discounts or adding points

Designing the database schema

2a) How would you structure the database tables to efficiently store inventory, sales, and customer information?

Key Table Structures:

Products Table:

- Product ID
- Product Name
- Category
- Stock Availability
- Price

Customers Table:
- Customer ID
- First Name
- Last Name
- Email
- Phone Number

Loyalty Accounts Table:
- Loyalty ID
- CustomerID
- Points
- Tier

Sales Table:
- SalesID
- CustomerID
- Date of transaction
- Total

Sales Detail Table:
- SaleDetailID
- SalesID
- ProductID
- Quantity
- Price

2b) What relationships between tables are necessary (e.g., how sales relate to inventory and customers)?
- Customer to Loyalty Account (1 to 1) = One customer has one loyalty account. Each loyalty account belongs to only one customer.
- Customer to Sales (1 to many) = One customer can make many purchases but each sale is linked to only one customer.

- Sales to Sales Detail (1 to many) = One sale could include many individual items (line entries) but each line item refers to one sale.
- Sales Detail to Products (many to 1) = Many line items across multiple sales can reference the same product . One product can appear in many sale details.
- Product to Sales detail (many to many with sales detail as junction table) = A product can appear in multiple sales and one sale can include many products.

Implementing the database

3a) What SQL commands would you use to create the database and its tables?

- CREATE DATABASE = To create the database
- USE = To select the database
- CREATE TABLE = To create the structure of each table
- PRIMARY KEY = To uniquely identify each row
- FOREIGN KEY = To establish relationships between tables

3b) Provide examples of SQL statements for creating tables and defining relationships between them.

```
CREATE TABLE Sales
SalesID INT Primary Key,
CustomerID INT,
Dateoftransaction DATE,
Total DECIMAL (10,2),
FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID);
```

Explanation = Create a table called Sales. Each sale has an ID, belongs to a customer, happened on a specific date and has a total amount. Ensure that customer ID matches one in the customers table.

```
CREATE TABLE LoyaltyAccounts
Loyalty ID INT PRIMARY KEY,
CustomerID INT,
Points INT,
Tier VARCHAR(20)
FOREIGN KEY (CustomerID) REFERENCES Customers (CustomerID);
```

Explanation = Create a table to hold loyalty accounts, Each account has its own ID, links to a customer, tracks the customer's points and records what tier they're in. Ensure the customer ID matches one in the customers table.

Populating the database

4a) How would you input initial data into the database? Give examples of SQL INSERT statements.

1) Insert into Customers Table
   INSERT INTO Customers (CustomerID, FirstName, LastName, Email, Phone Number)
   VALUES (1, 'Toby', 'Elis', 'toby.elis@email.co.uk', '07549124598')

2) Insert into Products Table
   INSERT INTO Products (ProductID, ProductName, Category, Stock Availability, Price)
   VALUES (89, 'Vaseline Body Lotion – 500ml,' 'Health and Beauty',100, 3.00);

Maintaining the database

5a) What measures would you take to ensure the database remains accurate and up to date?

Using triggers for automatic updates = using the CREATE TRIGGER function to automatically update specific fields such as stock levels or loyalty points when a sale is recorded

Regular data validation and cleaning = running queries to find any anomalies such as duplicate emails or sales with no details. Removing/fixing corrupt or invalid records.

Using constraints to ensure data integrity = using primary keys (unique identification), foreign keys (maintaining valid relationships), NOT NULL (preventing missing data in essential columns).

4b) How would you handle backups and data security?

Handling Backups:

- Regular scheduled backups = automating backups of the database daily or weekly, depending on transaction volume for example.

- Test backups regularly = test restoring the backups every few weeks to make sure they are valid.

Data Security

- Restricted access = implement the idea of granting permissions based on user roles.
- Avoid storing personally identifiable information as plain text = consider masking or encrypting data such as customer emails and phone numbers.

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.