

**Data Technician**

|  |
| --- |
|  |

|  |
| --- |
| Name: |
| Course Date: |
|  |

**Table of contents**

[Day 1: Task 1 3](#_Toc106466110)

[Day 1: Task 2 3](#_Toc1673105856)

[Day 3: Task 1 4](#_Toc1682646377)

[Day 4: Task 1: Written 6](#_Toc135095808)

[Day 4: Task 2: SQL Practical 9](#_Toc1228298484)

[Course Notes 19](#_Toc631049525)

[Additional Information 19](#_Toc1612679949)

# 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 column that can identify and distinguish between each record in a database.  Source - [Primary key - Wikipedia](https://en.wikipedia.org/wiki/Primary_key) |
| How does this differ from a secondary key? | A secondary key is a column that can also identify and distinguish between each record in a database but is not selected as the primary key. The differences between them is that the secondary key can have null values but the primary key cannot and there can be zero or multiple secondary keys but every must have only one primary key.  Source - [What is a Secondary Key in DBMS? - Scaler Topics](https://www.scaler.com/topics/secondary-key-dbms/) |
| How are primary and foreign keys related? | A foreign key is a column or columns in a database that links to a primary key of another table.  Source - [What is a foreign key? (with SQL examples)](https://www.cockroachlabs.com/blog/what-is-a-foreign-key/) |
| Provide a real-world example of a one-to-one relationship | In most websites that allow users to create accounts, there is a one-to-one relationship between the user accounts and email addresses. Most websites will only allow each account to have one email address associated to it (they may allow a backup email but there will still be a primary email address) and will only allow one account to be associated with each email address. |
| Provide a real-world example of a one-to-many relationship | There is a one-to-many relationship between customers of a shop and products bought by the customer. This is because a customer can buy multiple items, but each item can only be bought by a single customer. |
| Provide a real-world example of a many-to-many relationship | There is a many-to-many relationship between clothing item and clothes brands. For example, Adidas, Levi’s and Next all make shirts and shirts are made by multiple brands. |

# 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 are always stored in columns and rows whilst non-relational databases can use many different data models.  The schemas of non-relational databases are flexible and more easily changed whilst the schemas for relational databases are more fixed.  Non-relational databases can have higher performance and availability than relational databases whilst relational databases are more consistent. |
| What type of data would benefit off the non-relational model?  Why? | Non-relational databases are more flexible and so can more easily accommodate data that does not easily fit into a relational database such as data no clear structure e.g. a JSON file. Since it also can have high performance and availability, this also makes it suitable for large data and data that requires rapid processing. |

# 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 | Returns the table joined with itself.  Example: In the sakila database we can join the film\_actor table to match the actors who appeared in the same film.  SELECT A.actor\_id AS actor1, B.actor\_id AS actor2, A.film\_id  FROM film\_actor A, film\_actor B  WHERE A.film\_id = B.film\_id; |
| Right join | Returns all the records from the right table and the records from the left table that match with the right table.  Example: In the sakila database, the address and customer tables share the address\_id column. If we right join the address and customer tables, it will show all the customers and join the customers that have an address in the address table.  SELECT \*  FROM address  RIGHT JOIN customer ON address.address\_id = customer.address\_id; |
| Full join | Returns all the records which have a match in either the left or right table. This not available in SQL but not in MySQL.  Example: In the W3Schools SQL database, the customers and the orders table both contain the OrderID column. If we full join the tables, it will return all the customers and the orders regardless of if they match or not.  SELECT Customers.CustomerName, Orders.OrderID FROM Customers FULL OUTER JOIN Orders ON Customers.CustomerID=Orders.CustomerID ORDER BY Customers.CustomerName; |
| Inner join | Returns the records that have matching values in both tables.  Example: In the Sakila database there is a film\_actor and a film table both containing the film\_id column. We can do an inner join with the film\_id column to which films are in both the film\_actor table and film table.  SELECT \*  FROM film  INNER JOIN film\_actor ON film.film\_id = film\_actor.film\_id  ORDER BY film\_actor.actor\_id; |
| Cross join | Returns all records from both tables.  Example – In the Sakila database, if we cross join the address and the city tables we get every record from both tables, including the ones that do not match.  SELECT \*  FROM city  CROSS JOIN address; |
| Left join | Returns all the records from the left table and the records from the right table that match with the left table.  Example: In the Sakila database there is a customer and a rental table both containing the customer\_id column. If we left join the customer table to the rental table it will give us all the customers and join the customers that have a rental in the rental table.  SELECT \*  FROM customer  LEFT JOIN rental ON customer.customer\_id = rental.customer\_id  ORDER BY rental\_id; |

# 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****:*
   1. *What kind of data will the database need to store?*
   2. *Who will be the users of the database, and what will they need to accomplish?*
2. ***Designing the Database Schema****:*
   1. *How would you structure the database tables to efficiently store inventory, sales, and customer information?*
   2. *What relationships between tables are necessary (e.g., how sales relate to inventory and customers)?*
3. ***Implementing the Database****:*
   1. *What SQL commands would you use to create the database and its tables?*
   2. *Provide examples of SQL statements for creating tables and defining relationships between them.*
4. ***Populating the Database****:*
   1. *How would you input initial data into the database? Give examples of SQL INSERT statements.*
5. ***Maintaining the Database****:*
   1. *What measures would you take to ensure the database remains accurate and up to date?*
   2. *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.  1. **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.  1. **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 |   **2. 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** |   **3. 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.    Implementing the Database:**  a)      For us to create a database and the tables in SQL we need to use the create database command  “CREATE DATABASE RetailStoreDB;”  b)     Then we proceed to use the **create table** function, for each of our tables ( Products, Customers, and Sales)  I.E:     CREATE TABLE Products   (         ProductID INT PRIMARY KEY,  ProductName VARCHAR(100),  Category VARCHAR(50),  Price DECIMAL(10, 2),  StockLevel INT );    c)      Then we insert all of 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:**  **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.  Query :-  Update Products  Set StockLevel = StockLevel - 2  Where Productname  = ‘Milk’;  **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;  **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:  **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.

|  |
| --- |
| SELECT COUNT(Name) AS "Number of Cities in USA"  FROM city  WHERE CountryCode = "USA"; |

1. **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, LifeExpectancy  FROM country  WHERE LifeExpectancy = (SELECT MAX(LifeExpectancy) FROM country); |

1. **"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  FROM city  WHERE Name LIKE '%New %'; |

1. **Display Columns with Limit (First 10 Rows):** *Scenario:* You're tasked with providing a brief overview of the most populous cities in the world. To keep the report concise, you're instructed to list only the first 10 cities by population from the database.

|  |
| --- |
| SELECT Name, Population  FROM city  ORDER BY Population DESC LIMIT 10; |

1. **Cities with Population Larger than 2,000,000:** *Scenario:* A real estate developer is interested in cities with substantial population sizes for potential investment opportunities. You're tasked with identifying cities from the database with populations exceeding 2 million to focus their research efforts.

|  |
| --- |
| SELECT Name, Population  FROM city  WHERE Population > 2000000  ORDER BY Population; |

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

|  |
| --- |
| SELECT Name  FROM city  WHERE Name LIKE 'Be%'; |

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

|  |
| --- |
| SELECT Name, Population  FROM city  WHERE Population > 500000 AND Population < 1000000  ORDER BY Population; |

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

|  |
| --- |
| SELECT Name  FROM city  ORDER BY Name; |

1. **Most Populated City:** *Scenario:* A real estate investment firm is interested in cities with significant population densities for potential development projects. You're tasked with identifying the most populated city from the database to guide their investment decisions and strategic planning.

|  |
| --- |
| SELECT Name, Population  FROM city  WHERE Population = (SELECT MAX(Population) FROM city); |

1. **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, COUNT(Name) AS "Number of Occurances"  FROM city  GROUP BY Name  ORDER BY Name; |

1. **City with the Lowest Population:** *Scenario:* A census bureau is conducting an analysis of urban population distribution. You're tasked with identifying the city with the lowest population from the database to provide a comprehensive overview of demographic trends.

|  |
| --- |
| SELECT Name, Population  FROM city  WHERE Population = (SELECT MIN(Population) FROM city); |

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

|  |
| --- |
| SELECT Name, Population  FROM country  WHERE Population = (SELECT MAX(Population) FROM country); |

1. **Capital of Spain:** *Scenario:* A travel agency is organising tours across Europe and needs accurate information on capital cities. You're tasked with identifying the capital of Spain from the database to ensure itinerary accuracy and provide travellers with essential destination information.

|  |
| --- |
| Select country.Name, city.Name AS "Capital City"  FROM country  INNER JOIN city ON country.Capital = city.ID  WHERE country.Name = "Spain"; |

1. **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, LifeExpectancy  FROM country  WHERE LifeExpectancy = (SELECT MAX(LifeExpectancy) FROM country); |

1. **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 city.Name  FROM country  INNER JOIN city ON country.Code = city.CountryCode  WHERE country.Continent = "Europe"  ORDER BY city.Name; |

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

|  |
| --- |
| SELECT Name, Population  FROM country; |

1. **Capital Cities Population Comparison:** *Scenario:* A statistical analysis firm is examining population distributions between capital cities worldwide. You're tasked with comparing the populations of capital cities from different countries to identify trends and patterns in urban demographics.

|  |
| --- |
| Select country.Name AS "Country", city.Name AS "Capital City", city.Population AS "Population"  FROM country  INNER JOIN city ON country.Capital = city.ID; |

1. **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, Population/SurfaceArea AS "Population Density"  FROM country  ORDER BY Population/SurfaceArea; |

1. **Cities with High GDP per Capita:** *Scenario:* An economic consulting firm is analysing cities with high GDP per capita for investment opportunities. You're tasked with identifying cities with above-average GDP per capita from the database to assist the firm in identifying potential investment destinations.

|  |
| --- |
| SELECT city.Name AS CityName,         (country.GNP / city.Population) AS GDP\_Per\_Capita FROM city JOIN country ON city.CountryCode = country.Code WHERE (country.GNP / city.Population) > (     SELECT AVG(country.GNP / city.Population)     FROM city     JOIN country ON city.CountryCode = country.Code     WHERE city.Population > 0 ) AND city.Population > 0  ORDER BY GDP\_Per\_Capita; |

1. **Display Columns with Limit (Rows 31-40):** *Scenario:* A market research firm requires detailed information on cities beyond the top rankings for a comprehensive analysis. You're tasked with providing data on cities ranked between 31st and 40th by population to ensure a thorough understanding of urban demographics.

|  |
| --- |
| SELECT city.Name AS CityName,  city.Population,  country.Name AS CountryName  FROM city  JOIN country ON city.CountryCode = country.Code  ORDER BY city.Population DESC  LIMIT 10 OFFSET 30; |

|  |
| --- |
| **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:

|  |
| --- |
| **Additional Information** |

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.**