

**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** in SQL is a unique identifier for a table's records. It ensures that each record has a unique value and cannot contain NULLs. A table can only have one primary key, which can consist of one or multiple columns (called a composite key). It’s crucial for maintaining data integrity. |
| How does this differ from a secondary key? | A **secondary key**, on the other hand, is an attribute or set of attributes used for indexing or searching, but it is not necessarily unique and does not have to follow the same strict rules as the primary key. It’s mainly used to improve query performance. |
| How are primary and foreign keys related? | Primary and foreign keys are related because they establish relationships between tables in a database:   * A **primary key** uniquely identifies each record in its own table. * A **foreign key** is a field (or set of fields) in one table that refers to the primary key in another table, creating a link between the two. |
| Provide a real-world example of a one-to-one relationship | A real-world example of a one-to-one relationship in a database could be **passports and individuals**:   * Each person can have only one passport. * Each passport is assigned to only one individual.   In a database, there would be a table for individuals and another table for passports. The primary key in the "Individuals" table (e.g., "Person ID") would relate to a unique foreign key in the "Passports" table, ensuring that the relationship is strictly one-to-one. This setup maintains data integrity and avoids duplication! |
| Provide a real-world example of a one-to-many relationship | A real-world example of a one-to-many relationship is **teachers and students**:   * A single teacher can have many students they teach (one-to-many). * Each student, however, is taught by only one specific teacher for a particular class.   In a database, the "Teachers" table would have a primary key (e.g., "Teacher ID"), and the "Students" table would include a foreign key (e.g., "Teacher ID") to link each student to their teacher. This ensures that one teacher is associated with multiple students, but each student relates back to only one teacher in this context. |
| Provide a real-world example of a many-to-many relationship | A real-world example of a many-to-many relationship is **students and courses**:   * A single student can enrol in many courses. * Each course can have many students enrolled.   In a database, this relationship would require a junction table, such as "Enrolments," to link the "Students" table and the "Courses" table. The "Enrolments" table would include foreign keys like "Student ID" and "Coursed ID" to establish the connection between students and courses. This setup ensures flexibility while maintaining data integrity. |

# 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? | The key difference lies in how data is organized and stored:   * **Relational Databases**: Use tables with rows and columns to store data in a structured format. They rely on predefined schemas and relationships (e.g., primary and foreign keys) to connect data. Examples: MySQL, PostgreSQL. Best for structured data and complex relationships. * **Non-Relational Databases**: Store data in various formats like documents, key-value pairs, or graphs, without strict schemas. They are more flexible and handle unstructured or semi-structured data well. Examples: MongoDB, Cassandra. Best for dynamic or large-scale data.   Both have their strengths depending on your needs! |
| What type of data would benefit off the non-relational model?  Why? | Non-relational databases are ideal for **unstructured, semi-structured, or rapidly changing data**. Here's why:   1. **Dynamic Data**: Data without a fixed schema, such as social media posts or user-generated content, benefits because non-relational models adapt easily to changes. 2. **Large-Scale Data**: Big data applications like sensor logs or real-time analytics require scalability, which non-relational databases handle well by distributing data across servers. 3. **Hierarchical Data**: Data like JSON or XML, commonly used in APIs, fits naturally into document-based non-relational databases. 4. **Complex Relationships**: Graph databases (a type of non-relational model) excel at managing networks, such as social connections or recommendation systems.   Their flexibility and performance make them perfect for modern, high-volume, or evolving datasets. |

# 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 | * **What it is Self-join**: A table joins with itself to compare rows within the same table. * **Example Usage**: Employee hierarchy in an organization.   + Data: An "Employees" table with columns Employee ID, Name, and Manager ID (which references Employee ID).   + Use Case: Finding employees who report to the same manager. |
| Right join | **Right Join (Right Outer Join):**   * **What it is**: Retrieves all records from the **right table** and only matching records from the left table. * **Example Usage**: Comparing customer orders and product lists.   + Data: "Orders" table (left) and "Products" table (right).   + Use Case: Finding all products, including those that haven’t been ordered. |
| Full join | **Full Join (Full Outer Join):**   * **What it is**: Retrieves all records when there’s a match in **either** the left or right table (includes unmatched rows from both). * **Example Usage**: Merging customer and survey data.   + Data: "Customers" table and "Surveys" table.   + Use Case: Identifying customers who completed the survey, those who didn’t, and surveys without assigned customers. |
| Inner join | **. Inner Join:**   * **What it is**: Retrieves only records with matching values in both tables. * **Example Usage**: Linking orders to customers.   + Data: "Orders" table and "Customers" table.   + Use Case: Generating a list of completed orders with customer information. |
| Cross join | **. Cross Join:**   * **What it is**: Creates a Cartesian product of two tables by combining each row from the first table with every row from the second table. * **Example Usage**: Generating all combinations of products and promotions.   + Data: "Products" table and "Promotions" table.   + Use Case: Listing all possible product-promotion pairings. |
| Left join | **. Left Join (Left Outer Join):**   * **What it is**: Retrieves all records from the **left table** and only matching records from the right table. * **Example Usage**: Tracking employees and assigned projects.   + Data: "Employees" table (left) and "Projects" table (right).   + Use Case: Listing all employees, including those not assigned to any project. |

# 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 | A small corner shop selling groceries and domestic products requires an efficient database system to manage inventory, sales, and customer information, including a loyalty program. This essay outlines the steps to set up and create the database, emphasizing how each stage contributes to streamlined operations.  **1. Understanding the Business Requirements** The database must store key data such as product details (name, price, stock level, supplier), sales records (transaction date, products sold, quantity, total price), and customer information (name, contact details, loyalty points). The users of the database will be shop staff and managers, who need to update inventory, record sales, and retrieve customer loyalty data efficiently. Real-time updates and reports are crucial for decision-making, stock management, and enhancing customer engagement.  **2. Designing the Database Schema** The schema will consist of several tables, including:   * **Products**: Contains product ID (primary key), name, price, and stock level. * **Customers**: Stores customer ID (primary key), name, contact details, and loyalty points. * **Sales**: Includes sale ID (primary key), transaction date, customer ID (foreign key), and product details.   The relationships between tables link sales to both inventory and customers. For example, the **Sales** table references **Products** via product IDs and **Customers** via customer IDs, ensuring that transactions can be traced back to inventory adjustments and customer interactions.  **3. Implementing the Database** To create the database, SQL commands such as the following will be used:  SQL  CREATE TABLE Products (  ProductID INT PRIMARY KEY,  Name VARCHAR(50),  Price DECIMAL(10, 2),  StockLevel INT  );    CREATE TABLE Customers (  CustomerID INT PRIMARY KEY,  Name VARCHAR(50),  ContactDetails VARCHAR(100),  LoyaltyPoints INT  );  CREATE TABLE Sales (  SaleID INT PRIMARY KEY,  TransactionDate DATE ,  CustomerID INT,  FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)  );  These commands define the tables, fields, and relationships necessary for efficient data storage and retrieval.  **4. Populating the Database** Initial data for the tables can be input using SQL INSERT statements, such as:  sql  INSERT INTO Products ( Product ID, Name, Price, Stock Level) VALUES  (1, 'Milk', 1.50, 50),  (2, 'Bread', 0.80, 30);  INSERT INTO Customers (Customer ID, Name, Contact Details, Loyalty Points) VALUES  (1, 'John Doe', 'john.doe@example.com', 100);  These ensure the database has sufficient information to start operations effectively.  **5. Maintaining the Database** Regular updates to inventory and loyalty points ensure accuracy. For example, stock levels are adjusted after each sale using triggers or update commands. Backups are vital to prevent data loss, and security measures like access controls protect sensitive customer information. Indexing can optimize query performance, ensuring quick data retrieval as the database grows.  This plan ensures the corner shop operates efficiently, leveraging its database system for better inventory management, improved customer engagement, and streamlined sales tracking. |

# 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(\*) FROM city WHERE CountryCode = 'USA';    A screenshot of a computer  AI-generated content may be incorrect. |

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 FROM country WHERE LifeExpectancy IS NOT NULL ORDER BY LifeExpectancy DESC LIMIT 1;  A screenshot of a computer  AI-generated content may be incorrect. |

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%';  A screenshot of a computer  AI-generated content may be incorrect. |

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 \* FROM city ORDER BY Population DESC LIMIT 10;  A screenshot of a computer  AI-generated content may be incorrect. |

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 \* FROM city WHERE Population > 2000000;  A computer screen shot of a computer screen  AI-generated content may be incorrect. |

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%';  A screenshot of a computer  AI-generated content may be incorrect. |

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 \* FROM city WHERE Population >= 500000 AND Population <= 1000000;  A screenshot of a computer  AI-generated content may be incorrect. |

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 ASC;  A screenshot of a computer  AI-generated content may be incorrect. |

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 ORDER BY Population DESC LIMIT 1;  A screenshot of a computer  AI-generated content may be incorrect. |

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 Name, COUNT(\*) AS Frequency FROM city GROUP BY Name ORDER BY Name ASC;  A screenshot of a computer  AI-generated content may be incorrect. |

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 FROM city WHERE population = (SELECT MIN(population) FROM city WHERE population > 0);  A screenshot of a computer  AI-generated content may be incorrect. |

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 order by population desc limit 1;  A screenshot of a computer  AI-generated content may be incorrect. |

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 c.name from city c  join country co on c.id = co.capital  where co.name = 'spain';  A screenshot of a computer  AI-generated content may be incorrect. |

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 from country  where lifeexpectancy is not null  order by lifeexpectancy desc limit 1;  A screenshot of a computer  AI-generated content may be incorrect. |

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 name from city where countrycode in (select code from country where continent = 'europe');  A screenshot of a computer  AI-generated content may be incorrect. |

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 countrycode, avg(population) from city group by countrycode;  A screenshot of a computer  AI-generated content may be incorrect. |

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 c.name, c.population  from city c join country co on c.id = co.capital;  A screenshot of a computer  AI-generated content may be incorrect. |

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 avg(population / surfacearea) from country; select name from country  where population / surfacearea < (select avg(population / surfacearea) from country);  A screenshot of a computer  AI-generated content may be incorrect. |

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.Name AS CountryName,        city.Population AS CityPopulation,        country.GNP AS CountryGNP,        (country.GNP \* 1000000000 / country.Population) AS GDPPerCapita FROM city JOIN country ON city.CountryCode = country.Code WHERE (country.GNP \* 1000000000 / country.Population) >        (SELECT AVG(GNP \* 1000000000 / Population)         FROM country         WHERE Population > 0 AND GNP > 0) ORDER BY GDPPerCapita DESC;  A screenshot of a computer  AI-generated content may be incorrect. |

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.ID, city.Name, city.CountryCode, city.District, city.Population,        country.Name AS CountryName, country.Continent, country.Region FROM city JOIN country ON city.CountryCode = country.Code ORDER BY city.Population DESC LIMIT 30, 10;  A screenshot of a computer  AI-generated content may be incorrect. |