

**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 unique identifier for each record in a database table. It ensures that each record can be uniquely identified and does not have duplicates.  **Properties of a Primary Key:**   * **Unique:** No two records can have the same primary key value. * **Not Null:** Every record must have a primary key value (it cannot be left empty). * **Immutable:** It should not change once it’s set. |
| **How does this differ from a secondary key?** | A secondary key is any field (or set of fields) used to search or retrieve data but does not have to be unique.  **Key Characteristics:**   * **Can be duplicate:** Multiple records can share the same value for a secondary key. * **Used for querying:** Helps in searching for records based on criteria other than the primary key. |
| **How are primary and foreign keys related?** | * **Primary Key:** Identifies a unique record in a table. * **Foreign Key:** A foreign key is a field (or combination of fields) in one table that links to the primary key of another table. It establishes a relationship between the two tables.   **Relationship:**   * A primary key ensures the uniqueness of data within its own table. * A foreign key establishes a relationship between two tables by referring to the primary key in another table. |
| **Provide a real-world example of a one-to-one relationship** | Each person has one passport, and each passport belongs to only one person. |
| **Provide a real-world example of a one-to-many relationship** | A single author can write many books, but each book is written by only one author. |
| **Provide a real-world example of a many-to-many relationship** | A student can enroll in many courses, and each course 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?** | **Relational Database:**   * Stores data in tables with rows and columns. * Uses SQL for querying. * Has a predefined schema and relationships (via foreign keys). * Examples: MySQL, PostgreSQL.   **Non-relational Database:**   * Stores data in formats like JSON, key-value, documents, or graphs. * Uses NoSQL queries. * Schema-less and doesn’t require predefined relationships. * Examples: MongoDB, Cassandra, Redis. |
| **What type of data would benefit off the non-relational model?**  **Why?** | * **Unstructured/Semi-structured Data** * **Example:** Social media posts, comments. * **Why:** Flexible schema (JSON format) to handle varied data types. * **High-Volume, High-Speed Data** * **Example:** IoT data, logs. * **Why:** Optimized for high-speed reads/writes and horizontal scalability. * **Flexible Schema Needs** * **Example:** Product catalogs, user profiles. * **Why:** No need for a fixed schema, allowing easy data structure changes. * **Graph-Based Data** * **Example:** Social networks, recommendations. * **Why:** Graph databases like Neo4j handle complex relationships efficiently. * **Scalable Applications** * **Example:** Global platforms (e-commerce). * **Why:** Horizontal scalability across multiple servers for large datasets. * **Summary**   All of these data types benefit from NoSQL databases due to their flexibility, scalability, and ability to handle large, unstructured datasets efficiently. |

# 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:** A self-join is a regular join, but the table is joined with itself.  · **When to use:** When you want to compare rows within the same table — often used for hierarchical relationships like employees and their managers.  **Extra Code Example (Use case: Find each employee’s manager using the same employees table):**  SELECT e1.name AS Employee, e2.name AS Manager  FROM employees e1  JOIN employees e2 ON e1.manager\_id = e2.id; |
| **Right join** | · **What it is:** Returns all rows from the **right** table and the matched rows from the **left** table. If there's no match, NULL is returned for left table columns.  · **When to use:** When you want to include **all rows from the right table**, even if there’s no matching data on the left.  **Extra Code Example (Use case: Get a list of all customers, including those who haven’t made any orders):**  SELECT orders.order\_id, customers.name  FROM orders  RIGHT JOIN customers ON orders.customer\_id = customers.id; |
| **Full join** | · **What it is:** Combines the results of both left and right joins. Returns all rows from both tables, matching them where possible.  · **When to use:** When you want to see **all data from both tables**, including unmatched records on either side.  **Extra Code Example (Use case: Show all employees and all projects, whether or not they’re assigned to each other):**  SELECT employees.name, projects.project\_name  FROM employees  FULL JOIN projects ON employees.project\_id = projects.id; |
| **Inner join** | · **What it is:** Returns only rows where there is a match in **both** tables.  · **When to use:** When you want only the data that exists in both tables — the most common type of join.  **Extra Code Example (Use case: Show only customers who have made orders):**  SELECT orders.order\_id, customers.name  FROM orders  INNER JOIN customers ON orders.customer\_id = customers.id; |
| **Cross join** | · **What it is:** Returns the **Cartesian product** of both tables — every row from the first table combined with every row from the second.  · **When to use:** When you want all possible combinations — usually for generating test data or pairing items from different categories.  **Extra Code Example (Use case: Create a list of all possible color-size combinations for a product):**  SELECT color.name, size.label  FROM colors  CROSS JOIN sizes; |
| **Left join** | · **What it is:** Returns all rows from the **left** table, and matched rows from the **right** table. If there is no match, returns NULL on the right side.  · **When to use:** When you want all records from the left table, even if there’s no match in the right table.  **Extra Code Example (Use case: Get a list of all customers, including those with no orders):**  SELECT customers.name, orders.order\_id  FROM customers  LEFT JOIN orders ON customers.id = orders.customer\_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**  **a. Data Storage Needs:**  The business needs to store information on inventory (names, categories, prices, quantities, and suppliers), sales (items sold, sale date, and amounts), customers (names, contact details, and loyalty points), suppliers (names, phones and accounts) and loyalty transactions (points earned or redeemed). This data will support daily operations and strategic decisions.  **b. Database Users:**  Typical users include cashiers, store managers, and database administrators. Cashiers will record sales and update inventory, managers will monitor stock and analyze sales, and admins will maintain data integrity and security.  **2. Designing the Database Schema**  **a. Table Structure:**  The database will include these main tables:   * **Customers:** to store personal and loyalty information. * **Inventory:** to manage inventory. * **Suppliers:** to record every supplier * **Sales:** to record each transaction. * **Sale\_Items:** to track each product within a sale. * **Loyalty\_Transactions:** to track loyalty point usage.     **b. Relationships:**   * Each Sale is linked to one Customer. * Each Sale contains multiple Products (via Sale\_Items). * Each Product can appear in many sales. * Each Supplier supply multiple products. * Loyalty transactions are tied to customers.   **3. Implementing the Database**  a**. SQL Commands to Create Tables:**  **CREATE DATABASE** retail\_store;  **USE** retail\_store;  **CREATE TABLE** Customers (  customer\_id INT PRIMARY KEY AUTO\_INCREMENT,  first\_name VARCHAR(50),  last\_name VARCHAR(50),  email VARCHAR(100),  phone VARCHAR(20),  loyalty\_points INT DEFAULT 0  );  **CREATE TABLE** Inventory (  product\_id INT PRIMARY KEY AUTO\_INCREMENT,  name VARCHAR(100),  category VARCHAR(50),  price DECIMAL(10,2),  stock\_quantity INT,  supplier VARCHAR(100)  );  **CREATE TABLE** Suppliers (  Supplier\_id INT PRIMARY KEY AUTO\_INCREMENT,  name VARCHAR(100),  phone VARCHAR(50),  account VARCHAR(100)  );  **CREATE TABLE** Sales (  sale\_id INT PRIMARY KEY AUTO\_INCREMENT,  customer\_id INT,  sale\_date DATE,  total\_amount DECIMAL(10,2),  FOREIGN KEY (customer\_id) REFERENCES Customers(customer\_id)  );  **CREATE TABLE** Sale\_Items (  sale\_item\_id INT PRIMARY KEY AUTO\_INCREMENT,  sale\_id INT,  product\_id INT,  quantity INT,  subtotal DECIMAL(10,2),  FOREIGN KEY (sale\_id) REFERENCES Sales(sale\_id),  FOREIGN KEY (product\_id) REFERENCES Products(product\_id)  );  **CREATE TABLE** Loyalty\_Transactions (  loyalty\_id INT PRIMARY KEY AUTO\_INCREMENT,  customer\_id INT,  points\_earned INT,  points\_redeemed INT,  transaction\_date DATE,  FOREIGN KEY (customer\_id) REFERENCES Customers(customer\_id)  );  These SQL statements establish the schema and enforce relationships to maintain referential integrity.  **4. Populating the Database**  **a. Inserting Initial Data:**  **INSERT INTO** Inventory (name, category, price, stock\_quantity, supplier)  **VALUES** ('Chocolate Stick', 'Bakery', 3.20, 50, 'Fresh Bakes Ltd.');  **INSERT INTO** Customers (name, email, phone)  **VALUES** ('Sean Chen', 'Sean@example.com', '444-1234');  The statements above show sample sql codes to test and run the system.  **5. Maintaining the Database**  **a. Ensuring Accuracy:**  Implement input validation, use foreign keys, and regularly audit data for errors. Automated scripts can alert for low inventory or inactive customers.  **b. Backups and Security:**  Schedule daily backups using SQL tools. Limit access with user roles (e.g., READ ONLY, ADMIN) and encrypt sensitive data like emails. Use SSL for remote connections and enable logging for traceability. |

# 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)  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 DESC; |

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 Distinct 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 Between 500000 AND 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 Distinct Name  FROM city  ORDER BY Name ASC; |

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; |

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 Name\_frequency  FROM city  GROUP BY Name  ORDER BY Name ASC; |

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  ORDER BY Population ASC  Limit 1; |

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; |

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 AS Country, city.Name AS city  FROM country  LEFT JOIN city  ON country.Capital = city.ID  WHERE country.Name = "Spain"; |

1. **Country with Shortest Life Expectancy:** *Scenario:* A healthcare foundation is conducting research on global health indicators. You're tasked with identifying the country with the shortest life expectancy from the database to inform their efforts in improving healthcare systems and policies.

|  |
| --- |
| SELECT Name, LifeExpectancy  FROM country  WHERE LifeExpectancy  ORDER BY LifeExpectancy ASC  LIMIT 1; |

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 country.Continent AS Continent, city.Name AS city  FROM country  LEFT JOIN city  ON country.Code = city.CountryCode  WHERE country.Continent = "Europe"; |

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 country.Name, A.CountryCode, A.AVG\_population  FROM country  LEFT JOIN (SELECT CountryCode, AVG(Population) AS AVG\_population  FROM city  GROUP BY CountryCode) AS A  ON country.code = A.CountryCode  ORDER BY A.AVG\_population DESC; |

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\_Name, city.Name AS Capital, city.Population  FROM country  LEFT JOIN city  ON country.Capital = city.ID  ORDER BY city.Population DESC; |

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\_Density ASC; |

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, A.GDP, A.AVG\_GDP  FROM (SELECT Name, capital, (GNP/Population) AS GDP,  (SELECT AVG(GNP/Population) FROM country) AS AVG\_GDP  FROM country  WHERE (GNP/Population) > (SELECT AVG(GNP/Population) FROM country)) AS A  LEFT JOIN city  ON A.Capital = city.ID  ORDER BY A.GDP DESC; |

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 Name, Population  FROM city  ORDER BY 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.**