

**Data Technician**

|  |
| --- |
|  |

|  |
| --- |
| Name: Anu Chandrasekar |
| Course Date: Data Technician  Week 3 Work Book |
|  |

**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 distinctly identified, and no two records in the table can have the same value for the primary key fields. This is essential for maintaining data integrity and enabling efficient data retrieval. |
| How does this differ from a secondary key? | A secondary key, often referred to as a non – Primary key, is any field or combination of fields, in a database table that is used to identify records in a way that is not the primary key. While a Primary key uniquely identifies each record, a Secondary key provides an alternate way to look up or organize data. |
| How are primary and foreign keys related? | Primary key and foreign key are closely related in relational databases because they help establish and enforce relationships between tables. |
| Provide a real-world example of a one-to-one relationship | A one-to-one relationship in a database means that each record in one table is linked to exactly one record in another table, and vice versa.  For Example: -  Employees table linked to Employee details table where each entry in this table corresponds to one employee in the Employees table defining a 1-1 relationship |
| Provide a real-world example of a one-to-many relationship | A one- to – many relationships in a database means that a single record in one table is linked to multiple records in another table. This is one of the most common types of relationships in relational databases.  For Example: -  In e-commerce, Customer table linked to order table, where every customer can have multiple order entry details |
| Provide a real-world example of a many-to-many relationship | A many-to-many relationship occurs when multiple records in one table are related to multiple records in another table. This type of relationship is often implemented using a Junction table, also called an associative or bridge table, to link the two tables together.  For Example: -  Students can enrol into multiple courses and each course can have multiple students and courses |

# 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 main difference between a relational database (RDBMS) and a non-relational database (NoSQL) lies in how data is structured, stored, and managed.  **Key difference mentioned below:** -  **Data Structure**  **Relational Database (RDBMS)**:   * + Data is stored in **tables** with rows and columns.   + Each table has a **schema** that defines the structure of the data (i.e., the types of data each column can hold).   + Data is organized into **relations**, and each row in a table represents a **record**.   + **Examples**: MySQL, PostgreSQL, Oracle, SQL Serve   **Non-relational Database (NoSQL)**:   * Data can be stored in various formats like **key-value pairs**, **documents**, **graphs**, or **wide-column stores**. * Data does not require a fixed schema (schema-less), and different records can have different structures. * Examples of NoSQL types:   + **Document-based**: Stores data as documents (e.g., JSON or BSON). Example: MongoDB.   + **Key-value stores**: Data is stored as a collection of key-value pairs. Example: Redis.   + **Column-family stores**: Data is stored in column families. Example: Apache Cassandra.   + **Graph databases**: Data is represented as nodes and edges (relationships). Example: Neo4j. |
| What type of data would benefit off the non-relational model?  Why? | The non-relational database (NoSQL) model is particularly well-suited for handling unstructured, semi-structured, or large-scale data that doesn't fit neatly into traditional tabular schemas. Here are some types of data that would benefit from the non-relational model:  Time Series Data:  Measurement: "temperature"  Tags: {"sensor\_id": "A1", "location": "room1"}  Fields: {"value": 72.5}  Time: 2025-03-14T12:00:00Z  Time-series data is typically large and grows over time. For example, sensor readings may occur every second, resulting in millions or billions of records over time. Traditional SQL databases may not handle such large datasets efficiently in terms of indexing, partitioning, and querying. They are typically optimized for smaller transactional data rather than massive, append-only 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 | **Self-join –**  Is a type of join in SQL where a table is joined with itself. In other words, you use the same table twice in a query to compare rows within that table. This is useful when you need to find relationships within a single table.  **When to use a Self-Join:**  A self-join is used when you have data in a table that has a hierarchical or relational relationship with itself. For example, when you want to find the relationship between employees and managers in an organization (where both employees and managers are listed in the same table), a self-join is needed.  **Syntax:**  The syntax of a self-join involves giving two aliases to the same table so that you can refer to them as if they were two different tables.  SELECT a.column\_name, b.column\_name  FROM table\_name a  JOIN table\_name b  ON a.some\_column = b.some\_column; |
| Right join | **RIGHT JOIN**  (also known as **RIGHT OUTER JOIN**) is a type of join in SQL that returns all the rows from the right table and the matching rows from the left table. If there is no match, the result is NULL for columns from the left table. In simpler terms, it keeps all the rows from the right table and only the rows from the left table that have matching keys. |
| Full join | **FULL JOIN**  (also known as FULL OUTER JOIN) is a type of SQL join that combines the results of both LEFT JOIN and RIGHT JOIN. It returns all the rows from both tables being joined, and when there is no match, it will show NULL for the columns of the table that does not have the matching row. |
| Inner join | **INNER JOIN**  Is one of the most commonly used types of joins in SQL. It returns only the rows where there is a **match** in both the **left** and **right** tables. In other words, it combines rows from both tables based on the specified condition (usually matching column values) but only includes rows where there is a **corresponding match** in both tables. If there’s no match between the two tables, those rows are **excluded** from the result.  **Key Characteristics of INNER JOIN:**   * **Only matched rows**: An INNER JOIN only returns rows that have matching values in both tables. * If no match is found for a row in one table, that row will not appear in the result. * It is used when you only want to retrieve the data that exists in both tables, based on the join condition. |
| Cross join | **CROSS JOIN**  Is a type of join in SQL that returns the **Cartesian product** of two tables. This means it returns **every combination of rows** from the two tables, i.e., it pairs each row of the first table with each row of the second table. Unlike other joins (e.g., INNER JOIN, LEFT JOIN), a CROSS JOIN does not require a condition to match rows.  **Key Characteristics of a CROSS JOIN:**   * **No condition**: A CROSS JOIN does not require any condition to combine rows. * **Cartesian product**: It produces all possible combinations of rows from the two tables. * **Potentially large result**: The number of rows in the result set will be the product of the number of rows in each table (i.e., if Table A has m rows and Table B has n rows, the result will have m \* n rows). |
| Left join | **LEFT JOIN**  (also known as **LEFT OUTER JOIN**) is a type of SQL join that returns all the rows from the **left table** (the table listed first in the query) and the matching rows from the **right table** (the table listed second). If there is no match in the right table, the result will still include all rows from the left table, but the columns from the right table will contain NULL values.  **Characteristics of LEFT JOIN:**   * **All rows from the left table**: It returns every row from the left table, even if there is no corresponding match in the right table. * **Matching rows from the right table**: If a match is found between the tables based on the join condition, those rows are combined. * **NULL for non-matching rows**: If there is no matching row in the right table, the columns from the right table will contain NULL. |

# 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 | Day 4 Task 1:  1.**Understanding the Business Requirements:**  **Types of Data:**   * Customer Information: Names, addresses, phone numbers, email addresses, purchase history, preferences. * Product/Service Details: Product names, descriptions, prices, stock quantities, and categories (if applicable). * Transactions/Orders: Transaction IDs, order details, payment information, shipping status, dates.   **Who will be the user?**   * Manager- has restricted access to updates database, accounting revenue, * Store Staff- has restricted access to update inventory, stock availability, access loyalty points, discounted items, record sales * IT staff- has full access to modify and delete   **2.Designing the Database Schema**:   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | **Product** | | | | | | | ProductID | Product Name | Category | Price | Quantity stock | CustomerID | | 001 | Crisps | Snacks | £ 0.50 | 10 | 100 | | 002 | Popcorn | Snacks | £ 1.00 | 15 | 101 | | 003 | Nuts | Snacks | £1.20 | 8 | 100 | | 004 | Water | Beverages | £1.00 | 10 | 103 | | 005 | Soft drinks | Beverages | £ 1.50 | 20 | 103 | | 006 | **Sandwiches** | Fresh and Packaged Foods | £ 2.50 | 5 | 100 | | 007 | **Bread** | Bakery | £ 2.80 | 10 | 104 | | 008 | Milk | Dairy and refrigerated goods | £ 1.45 | 10 | 101 | | 009 | Yogurt | Dairy and refrigerated goods | £ 1.20 | 5 | 102 |   **Customers**   |  |  |  |  |  | | --- | --- | --- | --- | --- | | Customer ID | First Name | Last Name | Email address | Loyalty Points | | 100 | Jane | Doe | JaneDoe@gmail.com | 50 | | 101 | Jonh | Smith | JohnSmith@gmail.com | 10 | | 102 | Ray | Charles | RayCharles@gmail.com | 5 | | 103 | Jennifer | Vincent | JenVincent@gmail.com | 25 | | 104 | Richard | Brown | RichBrown@gmail.com | 30 |   **Sales**   |  |  |  |  |  | | --- | --- | --- | --- | --- | | Sales ID | ProductID | CustomerID | Date | Quantity | | 1001 | 001 | 100 | 2025/03/13 | 2 | | 1002 | 003 | 100 | 2025/03/10 | 1 | | 1003 | 006 | 100 | 2025/03/13 | 5 | | 1004 | 002 | 101 | 2025/03/13 | 2 | | 1005 | 008 | 101 | 2025/03/01 | 3 | | 1006 | 009 | 102 | 2025/03/13 | 4 | | 1007 | 004 | 103 | 2025/03/13 | 1 | | 1008 | 005 | 103 | 2025/03/03 | 2 | | 1009 | 007 | 104 | 2025/03/13 | 5 |   **Relationship:**  Product’s table will link to Customer’s table through the CustomerID  Product’s table will link to Sales’ table through the ProductID  **3. Implementing the database:**  To create a database, use a tool like MYSQL:   * Open a new table * Create 3 tables for Customers, product and sales. * Create the column headers and populate the tables   Example: CREATE TABLE Products (ProductID INT PRIMARY KEY, ProductName VARCHAR(100) NOT NULL, Category VARCHAR(50), Price DECIMAL(10, 2) NOT NULL, Quantity INT NOT NULL, Stock Int, Customerid Int);  4. **Populating the Database**:  Insert the data into product, customers and sales tables.  Product Table:  INSERT INTO PRODUCT (ProductID ,PRODUCTNAME,CATEGORY,PRICE,QUANTITY)  VALUES (001,"MILK","DAIRY AND REFIRIGRATED",1,10,100);  5**. Maintaining the Database:**   * **Loyalty Points Management**: Update points based on customer purchases * Regular Updates, regular audits: Monitoring stock to check for duplications. * Database Security: IT monitoring access control and actives * Backups, data encryption for sensitive data: to comply GDPR, Data protection Act and Computer Misuse Act. And regular back-up should be taken. * Staff training: training on how to use the system and comply with the laws pertaining the database. * Data Cleaning: to send inactive customers promotions, offers to up-sale * Monitoring: set-up, performance, alerts for issues.   System/ database should be stored in the cloud and external hard drive. Back up needs to be done daily. |

# 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 shot 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.

|  |
| --- |
|  |

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.

|  |
| --- |
|  |

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.

|  |
| --- |
|  |

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.

|  |
| --- |
|  |

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.

|  |
| --- |
|  |

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.

|  |
| --- |
|  |

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.

|  |
| --- |
|  |

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.

|  |
| --- |
|  |

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.

|  |
| --- |
|  |

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.

|  |
| --- |
|  |

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.

|  |
| --- |
|  |

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.

|  |
| --- |
|  |

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.

|  |
| --- |
|  |

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.

|  |
| --- |
|  |

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.

|  |
| --- |
|  |

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.

|  |
| --- |
|  |

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.

|  |
| --- |
|  |

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.

|  |
| --- |
|  |

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.

|  |
| --- |
|  |

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.

|  |
| --- |
|  |

|  |
| --- |
| **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.**