

**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? | Uniquely identifies each row in that table. It's like a unique ID number for each record, ensuring that no two rows have the same primary key value. |
| How does this differ from a secondary key? | Secondary key :  Uniqueness: Values in the secondary key column(s) can be duplicated (though this is often not ideal).  Nullability: It CAN contain NULL values.  Multiple per table: There can be MULTIPLE secondary keys per table.  Purpose: Used primarily for indexing and searching data, potentially improving query performance. |
| How are primary and foreign keys related? | They work together to establish and enforce relationships between tables. |
| Provide a real-world example of a one-to-one relationship | Each person is issued one, and only one, primary passport from a particular country |
| Provide a real-world example of a one-to-many relationship | One Customer, Many Orders |
| Provide a real-world example of a many-to-many relationship | Many Students, Many 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? | **Relational:** Ideal for applications requiring strict data consistency, ACID properties (atomicity, consistency, isolation, durability), and complex relationships between data.  **Non-Relational:** Suitable for applications requiring high scalability, flexibility, and handling large volumes of unstructured or semi-structured data.   |  |  |  | | --- | --- | --- | | Feature | Relational Database | Non-Relational Database | | Data Model | Relational (tables) | Key-value, document, graph, etc. | | Structure | Highly structured | Flexible, schema-less | | Relationships | Explicit, using keys | Implicit or explicit | | Scalability | Vertical | Horizontal | | Query Language | SQL | Varies | | Examples | MySQL, PostgreSQL | MongoDB, Cassandra | |
|  |  |
| What type of data would benefit off the non-relational model?  Why? | **1. Unstructured and Semi-structured Data:**   * Social Media Data: Posts, comments, likes, shares - this data often has varying formats and doesn't fit neatly into tables. * Sensor Data: IoT devices generate massive streams of data with changing structures. * **Log Files:** Application logs, server logs - these can have unpredictable formats and high volume. * Text Documents: Articles, emails, social media posts - analyzing these for sentiment or topic modeling.   **2. Large-Scale Data:**   * Big Data Applications: When dealing with terabytes or petabytes of data, NoSQL's horizontal scalability is often essential. * Real-time Analytics: Applications that need to process and analyze data in real-time, like online gaming or financial trading.   **3. Rapidly Changing Data:**   * Evolving Schemas: When your data structure changes frequently, NoSQL's schema-less nature allows for easier adaptation. * Agile Development: NoSQL can be a good fit for projects with evolving requirements and rapid iterations.   **The non-relational model (NoSQL ) is a Good Fit because :**   * **Flexibility:** NoSQL databases can handle diverse data formats and evolving schemas, making them well-suited for unstructured and semi-structured data. * **Scalability:** They excel at horizontal scaling, distributing data across multiple servers to handle massive datasets and high traffic loads. * **Performance:** NoSQL databases can offer high performance for specific types of queries and data retrieval patterns. * **Cost-Effectiveness:** NoSQL solutions can be more cost-effective for large-scale data storage and processing compared to traditional relational databases |

# Day 3: Task 1

Please research the below ‘JOIN’ types, explain what they are and provide an example of the types of data it would be used on.

|  |  |
| --- | --- |
| Self-join | A table is joined with itself. This is useful when you need to relate compare rows within the same table and finding related records.  Self-joins are useful for querying hierarchical data or comparing rows within the same table. |
| Right join | All rows from the right table are included: Regardless of whether they have a match in the left table,  matching rows from the left table are included: Only the rows from the left table that have a matching value in the join column with the right table,  If there's no match: If a row in the right table doesn't have a matching row in the left table, the columns from the left table will contain NULL values for that row. |
| Full join | To combine the results of two tables, is particularly useful when you need to see all records from both tables, regardless of whether there's a match. |
| Inner join | Is used to combine rows from two or more tables based on a related column between them. It returns only the rows where the 1 join condition is met.  It’s especially useful when data is split across multiple related tables. |
| Cross join | This every row from the first table is paired with every row from the second table, resulting in a combination of all possible pairs.  they’re practical only with small tables or when paired with additional filtering. |
| Left join | All rows from the left table are included: Regardless of whether they have a match in the right table,  matching rows from the left table are included: Only the rows from the right table that have a matching value in the join column with the left table,  If there's no match: If a row in the left table doesn't have a matching row in the right table, the columns from the right table will contain NULL values for that row. |

# 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 | **Understanding the Business Requirements**  The first step is to identify the data the database must store. For inventory, it needs product details like ID, name, category (e.g., groceries, cleaning supplies), price, stock quantity, and supplier. Sales data should include transaction ID, date, products sold, quantities, total amount, and payment method. Customer information requires customer ID, name, contact details (phone, email), and loyalty points earned or redeemed.  Users of the database will primarily be the shop staff—managers and cashiers—who need to track stock, process sales, and manage loyalty rewards, and possibly the owner for reporting purposes. Understanding these needs ensures the database aligns with daily operations.  **Designing the Database Schema**  The database will consist of four main tables: Products, Sales, Customers, and Sale\_Items (to link sales to products).  1.The Products table stores inventory data (e.g., product\_id, product\_name, price, stock\_quantity).   |  |  |  |  | | --- | --- | --- | --- | | Product\_id | Product\_name | Price | Stock\_quantity |   2.The Customers table holds customer data (e.g., Customer\_id, Customer\_name, Email, Loyalty\_points).   |  |  |  |  | | --- | --- | --- | --- | | Customer\_id | Customer\_name | Email | Loyalty\_points |   3.The Sales table records transactions (e.g., sale\_id, customer\_id, sale\_date, total\_amount), linking to Customers via customer\_id for loyalty tracking.   |  |  |  |  | | --- | --- | --- | --- | | sale\_id | customer\_id | sale\_date | total\_amount |   4.The Sale\_Items table connects sales to products (e.g., sale\_id, product\_id, quantity\_sold), enabling detailed sales analysis.   |  |  |  | | --- | --- | --- | | sale\_id | product\_id | quantity\_sold |   A foreign key in Sales ties it to Customers, and Sale\_Items links Sales and Products via sale\_id and product\_id, forming a many-to-many relationship between sales and inventory.  **Implementing the Database**  Creating the database: CREATE DATABASE:  CREATE TABLE Products (  product\_id INT PRIMARY KEY,  product\_name VARCHAR(255),  category VARCHAR(255),  price DECIMAL(10, 2),  stock\_quantity INT  );  CREATE TABLE Customers (  customer\_id INT PRIMARY KEY,  customer\_name VARCHAR(255),  email VARCHAR(255),  loyalty\_points INT  );  CREATE TABLE Sales (  sale\_id INT PRIMARY KEY,  customer\_id INT,  sale\_date DATE,  total\_amount DECIMAL(10, 2),  FOREIGN KEY (customer\_id) REFERENCES Customers(customer\_id)  );  CREATE TABLE Sale\_Items (  sale\_id INT,  product\_id INT,  quantity\_sold INT,  PRIMARY KEY (sale\_id, product\_id),  FOREIGN KEY (sale\_id) REFERENCES Sales(sale\_id),  FOREIGN KEY (product\_id) REFERENCES Products(product\_id)  );          **Visual Representation:**  Customers (customer\_id (PK), customer\_name, email, loyalty\_points)  | 1:M (One-to-Many)  Sales (sale\_id (PK), customer\_id (FK), sale\_date, total\_amount)  | 1:M (One-to-Many)  Sale\_Items (sale\_id (FK,), product\_id (FK), quantity\_sold)  | M:1 (Many-to-One)  Products (product\_id (PK), product\_name, category, price, stock\_quantity)    **Database Maintenance Strategy**  To maintain database accuracy and reliability, the following practices will be implemented:  1. Data Integrity and Updates:   * Real-time Transactional Updates: Implement immediate updates following sales transactions to ensure accurate inventory levels. * Input Validation: Implement robust input validation at the application level to prevent data entry errors and maintain data consistency.   2. Data Backup and Recovery:   * Scheduled Weekly Backups: Automate weekly database backups to mitigate data loss risk.   + Storage: Securely store backups offsite to ensure data availability in case of local system failures.   3. Data Security and Access Control:   * User Authentication and Authorization: Implement strong user authentication mechanisms and role-based access control to restrict database modifications to authorized personnel. * Data Encryption: Encrypt sensitive customer data at rest and in transit to protect against unauthorized access.   4. Data Auditing and Consistency:   * Regular Data Audits: Conduct periodic audits to identify and rectify data inconsistencies, ensuring the database remains a reliable source of information for shop operations |



# 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 CountryCode, count(countryCode) as cites  FROM world.city  where CountryCode = "USA"  ; |

2.**Country with Highest Life Expectancy:** *Scenario:* As part of a global health initiative, you've been assigned to identify the country with the highest life expectancy. This information will be crucial for prioritising healthcare resources and interventions.

|  |  |  |
| --- | --- | --- |
| SELECT Name, LifeExpectancy  FROM world.country  order by LifeExpectancy desc  LIMIT 1 ;     |  |  | | --- | --- | |  |  | |

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 world.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 world.city  order by Population desc ; |

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  FROM world.city  WHERE Population > 2000000  ; |

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 world.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  FROM world.city  WHERE Population Between 500000 and 1000000; |

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 world.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  FROM world.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 the a list of unique city names sorted alphabetically, along with their respective counts of occurrences in database. You're tasked with this sorted list to support the geography teacher.

|  |
| --- |
| SELECT Name, COUNT(\*) AS OccurrenceCount  FROM world.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  FROM world.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  FROM world.country  order by Population desc LIMIT 1; |

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

|  |
| --- |
| SELECT Name  FROM world.city  where ID IN(  SELECT Capital  FROM world.country  WHERE 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 \* FROM world.country;  SELECT Name  FROM world.country  order by LifeExpectancy desc 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.

|  |
| --- |
|  |

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 Region, AVG(Population) AS AveragePopulation  FROM country  GROUP BY Region  ORDER BY AveragePopulation DESC;  SELECT Name, AVG(Population) AS AveragePopulation  FROM country  GROUP BY Name; |

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 ID, Name, Population from city  where ID IN ( select capital from country)  order by Name  ; |

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 PopulationDensity  FROM country  WHERE SurfaceArea IS NOT NULL AND Population IS NOT NULL  order by PopulationDensity ; |

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.

|  |
| --- |
| with high GDP per capita), try filtering countries with a GDP per capita above the average using GNP / Population. |

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.**