

**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 field used to uniquely identify every record in the database. Sometimes primary keys are obvious, for example a car license plate could be used to uniquely identify cars. Often, primary keys are less obvious and it is necessary to create a unique value – often an ID number.  A primary key is a guaranteed way of uniquely identifying each record. In the school example, the primary key used to identify each member of staff in the teacher table is Staff ID. |
| How does this differ from a secondary key? | A "secondary" or "alternate" key is any key that is not selected as the primary. |
| How are primary and foreign keys related? | The primary key is a unique identifier within its table, whereas a foreign key is a reference in one table to a primary key in another. Primary keys enforce uniqueness within their table, ensuring each record is identifiable. Foreign keys, however, are used to establish and navigate relationships between tables. |
| Provide a real-world example of a one-to-one relationship | A real-world example of a one-to-one relationship is a passport being assigned to one individual, and that individual having only one passport. Each person can have only one passport, and each passport belongs to only one person. |
| Provide a real-world example of a one-to-many relationship | A real-world example of a one-to-many relationship is the relationship between a customer and their orders in an e-commerce system. A single customer can place many orders, but each order is linked to only one specific customer. This relationship can be visualized in a database with a "Customers" table and an "Orders" table, where the "Orders" table would contain a foreign key referencing the "Customers" table to indicate which customer placed each order. |
| Provide a real-world example of a many-to-many relationship | A many-to-many relationship occurs when multiple records in a table are associated with multiple records in another table. For example, a many-to-many relationship exists between employees and projects: employees can work on various projects, and a project can have many employees working on it. Another example is the relationship between orders and product: you can order various products when placing an order, and each product is contained in zero or more orders. |

# 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 primary difference between relational and non-relational databases lies in their data structure and storage models. Relational databases use a tabular format (rows and columns) with predefined schemas and SQL for querying, while non-relational (NoSQL) databases offer flexible storage models like documents, key-value pairs, graphs, and column families, often without a strict schema.  Relational Databases:   * **Structure:** Data is organized into tables with rows and columns, like a spreadsheet. * **Schema:** Have a predefined schema, meaning the structure of the data is set in advance. * **Query Language:** Primarily use SQL (Structured Query Language) to manage and query data. * **Use Cases:** Ideal for applications requiring structured data with strong consistency, like financial records and inventory systems.   In databases, **strong consistency** means that **after an update is made to the database, *any* subsequent read will return the most recent value**—no matter which replica or server you read from.  Put simply:  **Once you write data, everyone sees the updated data immediately.**   * **Examples:** MySQL, PostgreSQL, Oracle.   Non-Relational Databases (NoSQL)  Str**ucture:**  Data can be stored in various formats like documents, key-value pairs, graphs, or column families.  **Schema:**  Often schema-less or schema-flexible, meaning the structure can be more dynamic and less rigid.  **Query Language:**  May not use SQL, or might use it as a secondary language. Many NoSQL databases have their own query languages.  **Use Cases:**  Well-suited for handling unstructured or semi-structured data like social media content, multimedia files, and large, diverse datasets.  **EEExamples:**  MongoDB, Cassandra, Redis.  Key Differences Summarized:   |  |  |  | | --- | --- | --- | | **Feature** | **Relational Databases** | **Non-Relational Databases (NoSQL)** | | Data Structure | Tables (rows and columns) | Documents, Key-Value, Graphs, etc. | | Schema | Predefined | Schema-less or flexible | | Query Language | Primarily SQL | Various, may not use SQL | | Use Cases | Structured data, strong consistency | Unstructured data, scalability | | Scalability | Typically vertical | Often horizontal |   In essence, the choice between relational and non-relational databases depends on the nature of the data and the application's requirements. If you need highly structured data with strong consistency and ACID properties, relational databases are a good choice. If you need flexibility, scalability, and can handle unstructured data, non-relational databases might be more suitable. |
| What type of data would benefit off the non-relational model?  Why? | Non-relational databases are well-suited for unstructured, semi-structured, and rapidly changing data. This is because they offer flexibility in data storage and don't require a rigid schema like relational databases. They excel in handling large volumes of data, making them a good choice for big data applications and real-time web applications.  Here's a more detailed explanation:  Why non-relational databases are good for unstructured data:   * **Fl Flexibility:**   Non-relational databases can store data in various formats like documents (JSON, XML), key-value pairs, or graphs. They don't enforce a strict schema, allowing data structures to evolve without requiring database changes.   * **Sc Scalability:**   Non-relational databases are designed to scale horizontally by adding more servers to the database cluster. This allows them to handle large volumes of data and high traffic loads.  **High Performance:**  They are optimized for fast queries and can handle large data volumes, making them suitable for real-time applications.   * **Ea Ease of Development:**   Non-relational databases often have simpler data models and APIs, which can make development faster and easier.  Examples of data that benefits from non-relational databases:   * **S Social media data:**   Social media platforms generate large amounts of diverse data, including text posts, images, videos, and user metadata, which is well-suited for document-oriented NoSQL databases.   * **Bi Big data analytics :**   Non-relational databases can handle large datasets with unstructured information,  ma making them useful for analytics and machine learning.   * **E- E-commerce data:**   They can store product catalogs, user behavior data, and other unstructured data that might not fit well into a relational database.   * **IoI IOT data:**   The Internet of Things generates a wide range of data from various devices, which can be stored in flexible formats using non-relational databases.   * **CoContent management systems:**   They can store and manage unstructured content like articles, images, and videos. |

# 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 | **1. Self-Join**   * A **self-join** is when a table is joined **to itself**. * You use aliases to treat the table as if it were two separate tables. * **Example**: Employees table where you want to find each employee's manager (both are employees).   SELECT A.name AS Employee, B.name AS Manager  FROM Employees A  JOIN Employees B ON A.manager\_id = B.id; |
| Right join | **2. Right Join (Right Outer Join)**   * Returns **all rows from the RIGHT table** and the **matched rows from the LEFT table**. * If no match exists, NULLs are shown for left table columns.   SELECT \*  FROM Orders  RIGHT JOIN Customers ON Orders.customer\_id = Customers.id; |
| Full join | **3. Full Join (Full Outer Join)**   * Returns **all rows when there is a match in either** left or right table. * If there is no match, NULLs fill the missing side.   SELECT \*  FROM Employees  FULL JOIN Departments ON Employees.dept\_id = Departments.id; |
| Inner join | **4. Inner Join**   * Returns **only the rows that have matching values in both tables**. * This is the most common type of join.   SELECT \*  FROM Students  INNER JOIN Classes ON Students.class\_id = Classes.id; |
| Cross join | **5. Cross Join**   * Returns **all possible combinations** (Cartesian product) of rows from both tables. * No ON condition is needed. * Useful sometimes, but can generate **very large result sets**.   SELECT \*  FROM Products  CROSS JOIN Suppliers; |
| Left join | **6. Left Join (Left Outer Join)**   * Returns **all rows from the LEFT table**, and the **matched rows from the RIGHT table**. * If no match exists, NULLs are shown for right table columns.   SELECT \*  FROM Customers  LEFT JOIN Orders ON Customers.id = Orders.customer\_id;    **Quick Visualization of All Join types**   | **Type** | **Keeps unmatched rows from** | | --- | --- | | Inner Join | None (only matched rows) | | Left Join | Left table | | Right Join | Right table | | Full Join | Both tables | | Cross Join | All combinations (no match needed) | | Self-Join | Same table | |

# 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**:  The database needs to store information about inventory (products), sales transactions, customers, and loyalty programs. Key data includes product details (name, category, price, quantity), customer information (name, email, phone number, loyalty points), and sales data (date, time, items sold, amounts, and which customer made the purchase).  (products) – Product name, category, Price and Stock level  Customers(name, email, phone number, loyalty points),  sales data (date, time, items sold, amounts, and which customer made the purchase).  The users of this database will primarily be shop employees and managers. Employees will need to look up product availability, process sales, and enrol customers in the loyalty program. Managers will require access to reports on inventory levels, sales trends, and customer activity to make informed business decisions.  **2 . Designing the Database Schema**:  Database would be Normalised to reduce redundancy,  Ease of updates and accurate tracking of stock by deducting quantities on each sale. Tracking of stock would be implemtned by creating Triggers and Stored Procedures. (Not implemented in this Project design)  What relationships between tables are necessary (e.g., how sales relate to inventory and customers)?  a. Database Tables Structure  Here’s a basic schema with 3 main tables:  1. Products  Stores information about items available for sale.  Field Name Data Type Description  product\_id INT (PK) Unique identifier for the product  product\_name VARCHAR Name of the product  category VARCHAR Product category (e.g., dairy)  price DECIMAL Unit price  stock\_level INT Quantity available in stock  2. Customers  Stores customer contact and loyalty information.  Field Name Data Type Description  customer\_id INT (PK) Unique customer identifier  name VARCHAR Customer's name  email VARCHAR Email address  phone\_number VARCHAR Phone number  loyalty\_points INT Earned loyalty points  3. Sales  Records each sales transaction.  Field Name Data Type Description  sale\_id INT (PK) Unique sale identifier  customer\_id INT (FK) Link to the customer making the sale  sale\_date DATED Date of the sale  sale\_time TIME Time of the sale  total\_amount DECIMAL(10,2) Total cost of the sale    Fig 1 The ER Schema Diagram for the grocerystore database  **3.Implementing the Database**:   * 1. What SQL commands would you use to create the database and its tables?   Creation of the grocerystore database MySQL command  CREATE DATABASE grocerystore;  -- Products Table  CREATE TABLE Products (  Product\_ID INT AUTO\_INCREMENT PRIMARY KEY,  Product\_Name VARCHAR(100) NOT NULL,  Category VARCHAR(50),  Price DECIMAL(10,2),  Stock\_Level INT  );  -- Customers Table  CREATE TABLE Customers (  Customer\_ID INT AUTO\_INCREMENT PRIMARY KEY,  First\_Name VARCHAR(50),  Last\_Name VARCHAR(50),  Email VARCHAR(100),  Loyalty\_Points INT DEFAULT 0  );  -- Sales Table  CREATE TABLE Sales (  Sale\_ID INT AUTO\_INCREMENT PRIMARY KEY,  Product\_ID INT,  Customer\_ID INT,  total\_amount Decimal(10,2)  Date DATE,  Quantity INT,  FOREIGN KEY (Product\_ID) REFERENCES Products(Product\_ID),  FOREIGN KEY (Customer\_ID) REFERENCES Customers(Customer\_ID)   1. **Populating the Database**:   In a real system, sales staff would not have direct entry access into the database . A software client application form would be used for data entry E.G A web application form wriiten in Javascript with Nodejs at the backend that communicates with the database system.  However the underlying SQL statements for inputing data is executed using the sql INSERT/VALUES command and for the purpose of this project the following examples illustrates this.  -- Insert sample products  INSERT INTO Products (Product\_Name, Category, Price, Stock\_Level)  VALUES  ('Milk - 1L', 'Dairy', 1.20, 100),  ('Eggs - 12 Pack', 'Dairy', 2.50, 80);  -- Insert sample customers  INSERT INTO Customers (First\_Name, Last\_Name, Email, Loyalty\_Points)  VALUES  ('Emma', 'Wilson', 'emma.wilson@example.com', 45),  ('Liam', 'Brown', 'liam.brown@example.com', 80);  -- Insert sample sales  INSERT INTO Sales (Product\_ID, Customer\_ID, Date, Quantity) VALUES  (1, 1, '2025-04-20', 6), -- Emma bought 6 Bananas  (2, 2, '2025-04-21', 2); -- Liam bought 2 Milk);    Fig 1 SQL query to display some sales data    Fig 2 SQL query to display Customers info data.    Fig 3 SQL query to display Products form products table  **5. Maintaining the Database**  To keep the database accurate, regular audits of inventory and customer information should be performed. Constraints like CHECK and NOT NULL can be used to ensure data integrity.  Backups should be scheduled daily using automated scripts or tools like mysqldump:  mysqldump -u username -p grocerstore > backup.sql (Database on Linux System )  In a corporate environment, there will also be a centrally managed backups team that have automated scripts to take backups of data drives on a nightly basis which the DBA can check and reqularly check and carry out manual backup and restores.  Data security measures include granting user-specific access rights, encrypting sensitive data, and keeping the database server updated.  **Database Security Outline**  **1. User Access Control**   * **Role-Based Access Control (RBAC)**:   + Admin: Full access (owner/IT support).   + Manager: Inventory, sales reports, and customer loyalty.   + Cashier: Sales entry, view-only access to inventory.   + Customer Service: Access to customer and loyalty data only. * **Authentication & Password Policies**:   + Enforce strong passwords (min length, complexity).   + Require 2FA (e.g., SMS/email-based) for admin and manager accounts.   + Regular password expiration (e.g., every 90 days).   **2. Data Encryption**   * **At Rest**: Use AES-256 encryption for the database files. * **In Transit**: Enable SSL/TLS for all database connections. * **Sensitive Data Masking**:   + Mask credit card numbers, personal IDs, etc., when viewed.   **3. Network Security**   * **Firewall Rules**: Only allow known internal IP addresses or VPN access to the database server. * **Database Port Hardening**: Use non-default ports and disable unused services. * **Limit Remote Access**: Use SSH or VPN tunneling, and block public database access.   **4. Activity Logging & Monitoring**   * Log all access attempts and major changes (e.g., delete/update operations). * Alert on suspicious activity (e.g., access at odd hours, failed login attempts). * Keep logs for 6–12 months, depending on compliance needs.   **💾 Backup Strategy**  **1. Backup Frequency**   * **Full Backup**: Daily, outside business hours (e.g., 2 AM). * **Incremental Backup**: Hourly or every 2–4 hours during business hours. * **Transaction Log Backup** (if applicable): Every 15 minutes for point-in-time recovery.   **2. Storage Options**   * **Onsite Backup**: External hard drive or NAS device. * **Cloud Backup**:   + Use services like AWS S3, Google Cloud Storage, or Azure Blob Storage.   + Enable encryption and multi-region replication.   **3. Retention Policy**   * Keep daily backups for 7 days. * Weekly backups for 1 month. * Monthly backups for 1 year.   **4. Disaster Recovery Plan**   * Document recovery steps clearly (who, what, when). * Test restoration process quarterly to ensure backups work. * Store a printed and digital copy of recovery steps off-site.   **5. Automated Backup Scripts**   * Automate backups using scheduled tasks or cron jobs. * Include notifications on backup success/failure (e.g., email alerts).     **✅ Compliance and Best Practices**   * Ensure customer data privacy (e.g., GDPR if in the EU, or local equivalents). * Use database-level auditing tools if available. * Keep database software and server OS up-to-date with patches.   Here's a detailed **outline for backups and security** tailored to the **small retail business database** managing inventory, sales, customer information, and a loyalty program:  **Database Security Outline**  **1. User Access Control**   * **Role-Based Access Control (RBAC)**:   + Admin: Full access (owner/IT support).   + Manager: Inventory, sales reports, and customer loyalty.   + Cashier: Sales entry, view-only access to inventory.   + Customer Service: Access to customer and loyalty data only. * **Authentication & Password Policies**:   + Enforce strong passwords (min length, complexity).   + Require 2FA (e.g., SMS/email-based) for admin and manager accounts.   + Regular password expiration (e.g., every 90 days).   **2. Data Encryption**   * **At Rest**: Use AES-256 encryption for the database files. * **In Transit**: Enable SSL/TLS for all database connections. * **Sensitive Data Masking**:   + Mask credit card numbers, personal IDs, etc., when viewed.   **3. Network Security**   * **Firewall Rules**: Only allow known internal IP addresses or VPN access to the database server. * **Database Port Hardening**: Use non-default ports and disable unused services. * **Limit Remote Access**: Use SSH or VPN tunneling, and block public database access.   **4. Activity Logging & Monitoring**   * Log all access attempts and major changes (e.g., delete/update operations). * Alert on suspicious activity (e.g., access at odd hours, failed login attempts). * Keep logs for 6–12 months, depending on compliance needs.     **1. Database Firewall (DB Firewall)**  **Purpose**: To protect the database from unauthorized access and SQL-based attacks.  **🧱 Key Functions:**   * **Whitelist IPs**: Only allows traffic from the application server or trusted internal sources. * **SQL Injection Protection**: Blocks suspicious query patterns before they reach the database. * **Access Monitoring**: Logs and flags unusual DB access patterns.   **Diagram Role**: The DB Firewall sits **between the Application Server and the Database**, filtering every query sent.  **🖥️ 2. Application Server**  **Purpose**: Acts as the middle layer between users (e.g., store staff, POS system) and the database. |

# 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.

|  |
| --- |
|  |

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.

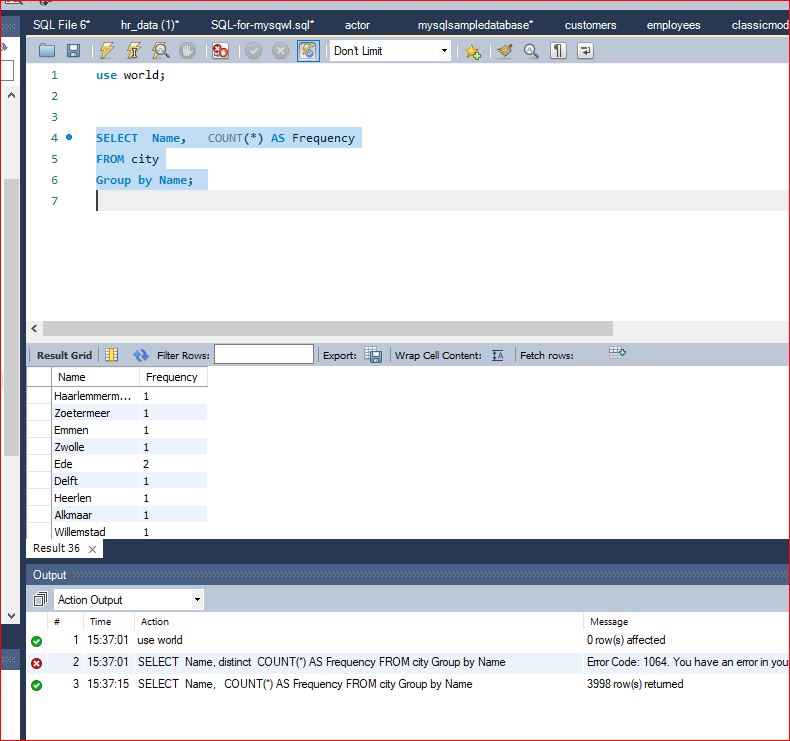
|  |
| --- |
|  |

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.

|  |
| --- |
|  |

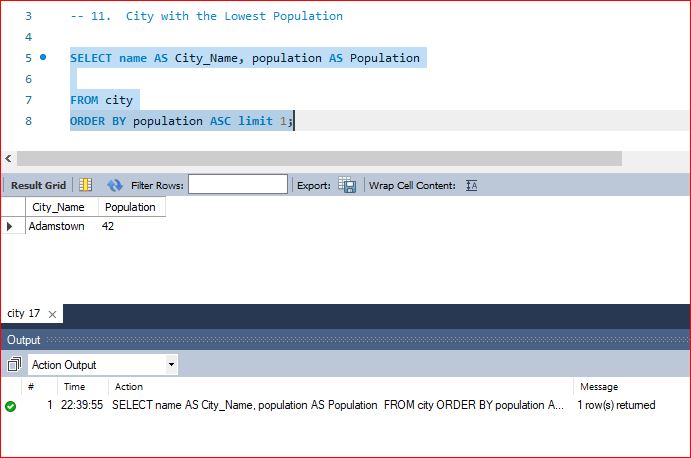
1. **Most Populated City:** *Scenario:* A real estate investment firm is interested in cities with significant population densities for potential development projects. You're tasked with identifying the most populated city from the database to guide their investment decisions and strategic planning.

|  |
| --- |
|  |

1. ****
2. **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.**