

**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 or set of fields in a database table that uniquely identifies each record in the table. Every table has one primary key, which ensures each row is unique.  Example: In a sales table, a unique OrderID could be a primary key because each order has a distinct identifier. |
| How does this differ from a secondary key? | A secondary key isn’t necessarily unique and can be used to reference or relate to other table.   |  |  |  | | --- | --- | --- | | Feature | Primary Key | Secondary Key | | Uniqueness | Must be unique for each record | Not necessarily unique | | Null values | Cannot have null values | Can contain null values | | Main purpose | To uniquely identify records | To enable relationships, lookups | | Example | OrderID(Unique Per Order) | ProductID(Reference to products) | |
| How are primary and foreign keys related? | The primary and foreign key relationship enforces referential integrity, meaning a record cannot exist in the foreign key table if it doesn’t have a matching record in the primary key table. |
| Provide a real-world example of a one-to-one relationship | Example:  Consider these 2 tables:  Table: Customers   |  |  | | --- | --- | | CustomerID(Primary Key) | Name | | 1 | John | | 2 | Bob | |
| Provide a real-world example of a one-to-many relationship | Table: Orders   |  |  |  | | --- | --- | --- | | OrderID | CustomerID(Foreign Key) | Amount | | 101 | 1 | 50 | | 102 | 2 | 75 | | 103 | 1 | 30 | |
| Provide a real-world example of a many-to-many relationship | * CustomerID in the Customers table is the primary key, identifying each customer. * CustomerID in the Orders table is a foreign key, linking each order to a customer in the Customers table. * The relationship between CustomerID in both tables enforces referential integrity, ensuring that every order is associated with an existing customer. |

# 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: Organises data into (rows and columns ) with predefined schemas. Tables are related through keys(Primary and foreign keys). Which establish relationships between them.  Example: SQL database like MySQL. |
| What type of data would benefit off the non-relational model?  Why? |  |

# 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 self join is a regular join, but the table is joined with itself.  SELECT A.CustomerName AS CustomerName1, B.CustomerName AS CustomerName2, A.City FROM Customers A, Customers B WHERE A.CustomerID <> B.CustomerID AND A.City = B.City ORDER BY A.City; |
| Right join | SQL RIGHT JOIN  The RIGHT JOIN keyword returns all records from the right table (table2), and the matching records from the left table (table1). The result is 0 records from the left side, if there is no match.  SELECT Orders.OrderID, Employees.LastName, Employees.FirstName FROM Orders RIGHT JOIN Employees ON Orders.EmployeeID = Employees.EmployeeID ORDER BY Orders.OrderID; |
| Full join | SQL FULL OUTER JOIN  The FULL OUTER JOIN keyword returns all records when there is a match in left (table1) or right (table2) table records.  Full outer join and outer join are the same. |
| Inner join | SQL INNER JOIN  An inner join is used to find related data in two tables. For example, suppose we need to retrieve data about a product and its category from the **SalesLT.Product** and **SalesLT.ProductCategory** tables. You can find the relevant product category record for a product based on its **ProductCategoryID** field; which is a foreign-key in the product table that matches a primary key in the product category table.  Example:  SELECT SalesLT.Product.Name AS ProductName, SalesLT.ProductCategory.Name AS Category  FROM SalesLT.Product  INNER JOIN SalesLT.ProductCategory  ON SalesLT.Product.ProductCategoryID = SalesLT.ProductCategory.ProductCategoryID; |
| Cross join | A *cross* join matches all possible combinations of rows from the tables being joined. In practice, it’s rarely used; but there are some specialized cases where it is useful.  SELECT p.Name, c.FirstName, c.LastName, c.EmailAddress  FROM SalesLT.Product AS p  CROSS JOIN SalesLT.Customer AS c; |
| Left join | SQL LEFT JOIN  The LEFT JOIN keyword returns all records from the left table (table1), and the matching records from the right table (table2). The result is 0 records from the right side, if there is no match.  SELECT Customers.CustomerName, Orders.OrderID FROM Customers LEFT JOIN Orders ON Customers.CustomerID = Orders.CustomerID ORDER BY Customers.CustomerName; |

# 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 requirement 2. The database will need to store the following types of data: 3. **Product Data:** 4. Product names, categories (e.g., groceries, domestic products), supplier details, cost and selling prices, stock levels, and reorder thresholds. 5. **Inventory Data:** 6. Stock quantities, purchase orders for restocking, and records of products sold. 7. **Sales Data:** 8. Individual sales transactions, total amounts, payment methods, and timestamps. 9. **Customer Data:** 10. Customer details such as names, contact information, and participation in the loyalty program. 11. **Loyalty Program Data:** 12. Points earned or redeemed, loyalty card numbers, and associated customer transactions. 13. **Supplier Data:** 14. Suppliers contact details, addresses, and records of past transactions. 15. **User and Operational Data:** 16. Information on employees or system users who manage the database, logs of transactions for accountability, and access permissions. 17. 2 b) **Shop Owner/Manager:**     * **Tasks:**       + Monitor inventory levels and identify low-stock items.       + Analyse sales trends and performance.       + Track customer behaviour and loyalty program performance.       + Manage supplier relationships and restocking orders.     * **Needs:**       + Generate reports for sales, stock levels, and customer insights.       + Set reorder levels and update product details as needed. 18. **Sales Staff:**     * **Tasks:**       + Process sales quickly at the point of sale.       + Look up product details (e.g., price, availability).       + Add loyalty points for customers or process point redemptions.     * **Needs:**       + Simple and fast data retrieval for sales and product information.       + Minimal permissions, focusing on daily operations. 19. **Customers:**     * **Tasks (Indirectly through loyalty program):**       + Earn points on purchases and redeem them.       + Access their loyalty account details (if integrated into a customer-facing system or app).     * **Needs:**       + Accuracy in points tracking and redemption.       + Confidence in product availability and prices. 20. **Accountant/Bookkeeper:**     * **Tasks:**       + Reconcile sales with payments received.       + Track costs, profits, and taxes.     * **Needs:**       + Detailed financial reports linking sales, costs, and inventory. 21. **System Administrator/IT Support:**     * **Tasks:**       + Maintain the database's integrity, security, and performance.       + Manage user permissions and troubleshoot system issues.     * **Needs:**       + Access to system logs, backups, and configuration settings.   ***2. A) Designing the Database Schema****:*  Relationships between tables:   1. **Products ↔ Suppliers:**    * A supplier supplies one or more products.    * **Relationship:** One-to-Many (One supplier → Many products). 2. **Sales ↔ SalesDetails:**    * Each sale can involve multiple products (line items), and each product can appear in multiple sales.    * **Relationship:** One-to-Many (One sale → Many sale details). 3. **SalesDetails ↔ Products:**    * A sale detail refers to a specific product sold in a transaction.    * **Relationship:** Many-to-One (Many sale details → One product). 4. **Sales ↔ Customers:**    * A sale may or may not be associated with a specific customer (walk-ins vs loyalty program members).    * **Relationship:** Many-to-One (Many sales → One customer). 5. **LoyaltyTransactions ↔ Customers:**    * A customer earns or redeems loyalty points through various transactions.    * **Relationship:** One-to-Many (One customer → Many loyalty transactions).  * B) * **Primary and Foreign Keys:** Use these to establish relationships and maintain data integrity. * **Indexes:** Create indexes on frequently queried columns (e.g., ProductID, CustomerID, SaleID) for faster lookups. * **Data Types:** Use appropriate data types (e.g., DECIMAL for prices, DATETIME for timestamps). * **Constraints:** Implement constraints like NOT NULL, UNIQUE, and CHECK where applicable to enforce data quality.     3.**What SQL commands would you use to create the database and its tables?**  The SQL commands primarily involve:   1. **Database Creation:** CREATE DATABASE to set up the database. 2. **Table Creation:** CREATE TABLE with primary keys, foreign keys, and other constraints. 3. **Relationships:** Defining relationships using FOREIGN KEY constraints.   **b. Examples of SQL Statements for Creating Tables and Defining Relationships**  Here’s the SQL script to create the database and its tables:  **Step 1: Create the Database**  CREATE DATABASE RetailManagement;  USE RetailManagement;  **Step 2: Create the Suppliers Table**  CREATE TABLE Suppliers (  SupplierID INT AUTO\_INCREMENT PRIMARY KEY,  SupplierName VARCHAR(100) NOT NULL,  ContactNumber VARCHAR(15),  Email VARCHAR(100),  Address TEXT  );  **Step 3: Create the Products Table**  CREATE TABLE Products (  ProductID INT AUTO\_INCREMENT PRIMARY KEY,  ProductName VARCHAR(100) NOT NULL,  Category VARCHAR(50),  SupplierID INT,  CostPrice DECIMAL(10, 2) NOT NULL,  SellingPrice DECIMAL(10, 2) NOT NULL,  StockQuantity INT NOT NULL DEFAULT 0,  ReorderLevel INT NOT NULL DEFAULT 0,  FOREIGN KEY (SupplierID) REFERENCES Suppliers(SupplierID)  ON DELETE SET NULL  ON UPDATE CASCADE  );  **Step 4: Create the Customers Table**  CREATE TABLE Customers (  CustomerID INT AUTO\_INCREMENT PRIMARY KEY,  FirstName VARCHAR(50) NOT NULL,  LastName VARCHAR(50) NOT NULL,  PhoneNumber VARCHAR(15),  Email VARCHAR(100),  LoyaltyCardNumber VARCHAR(20) UNIQUE,  PointsBalance INT NOT NULL DEFAULT 0  );  **Step 5: Create the Sales Table**  CREATE TABLE Sales (  SaleID INT AUTO\_INCREMENT PRIMARY KEY,  Date DATETIME NOT NULL DEFAULT CURRENT\_TIMESTAMP,  TotalAmount DECIMAL(10, 2) NOT NULL,  PaymentMethod ENUM('Cash', 'Card', 'Digital Wallet') NOT NULL,  CustomerID INT,  FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)  ON DELETE SET NULL  ON UPDATE CASCADE  );  **Step 6: Create the SalesDetails Table**  CREATE TABLE SalesDetails (  SaleDetailID INT AUTO\_INCREMENT PRIMARY KEY,  SaleID INT NOT NULL,  ProductID INT NOT NULL,  QuantitySold INT NOT NULL,  PriceAtSale DECIMAL(10, 2) NOT NULL,  FOREIGN KEY (SaleID) REFERENCES Sales(SaleID)  ON DELETE CASCADE  ON UPDATE CASCADE,  FOREIGN KEY (ProductID) REFERENCES Products(ProductID)  ON DELETE CASCADE  ON UPDATE CASCADE  );  **Step 7: Create the LoyaltyTransactions Table**  CREATE TABLE LoyaltyTransactions (  LoyaltyTransactionID INT AUTO\_INCREMENT PRIMARY KEY,  CustomerID INT NOT NULL,  TransactionType ENUM('Earned', 'Redeemed') NOT NULL,  Points INT NOT NULL,  Date DATETIME NOT NULL DEFAULT CURRENT\_TIMESTAMP,  FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)  ON DELETE CASCADE  ON UPDATE CASCADE  );  **Explanation of Key Features**   1. **Primary Keys:** AUTO\_INCREMENT ensures each record has a unique identifier. 2. **Foreign Keys:** Establish relationships between tables and enforce referential integrity. 3. **Constraints:**    * NOT NULL ensures essential fields are always filled.    * DEFAULT provides initial values for certain columns (e.g., StockQuantity, PointsBalance).    * ENUM restricts values for specific columns (e.g., PaymentMethod, TransactionType). 4. **On Delete/Update Actions:**    * Ensures cascading changes or sets fields to NULL when referenced records are updated or deleted.   **1. Reconcile Sales with Payments Received**  **Objective:** Ensure total sales match payments.  SELECT  SaleID,  TotalAmount,  PaymentMethod,  Date  FROM  Sales  WHERE  TotalAmount IS NULL  OR PaymentMethod IS NULL;  This query checks for missing or incomplete payment details.  **2. Track Costs, Profits, and Taxes**  **Objective:** Calculate revenue, COGS, profit, and tax.  **(a) Calculate Revenue and Profit per Sale**  SELECT  s.SaleID,  s.Date,  s.TotalAmount AS Revenue,  SUM(sd.QuantitySold \* p.CostPrice) AS CostOfGoodsSold,  (s.TotalAmount - SUM(sd.QuantitySold \* p.CostPrice)) AS Profit  FROM  Sales s  JOIN  SalesDetails sd ON s.SaleID = sd.SaleID  JOIN  Products p ON sd.ProductID = p.ProductID  GROUP BY  s.SaleID, s.Date, s.TotalAmount;  This query shows:   * Total revenue for each sale. * Cost of goods sold (COGS). * Profit (Revenue - COGS).   **(b) Calculate Taxes Collected (if applicable)**  Assuming a 10% tax rate on sales:  SELECT  SaleID,  TotalAmount,  (TotalAmount \* 0.10) AS TaxCollected  FROM  Sales;  **3. Inventory Movement and Costs**  **Objective:** Track inventory changes based on sales and restocking.  **(a) Inventory Reduction Report**  SELECT  p.ProductName,  SUM(sd.QuantitySold) AS TotalSold,  p.StockQuantity AS RemainingStock  FROM  SalesDetails sd  JOIN  Products p ON sd.ProductID = p.ProductID  GROUP BY  p.ProductName, p.StockQuantity  ORDER BY  TotalSold DESC;  This shows:   * Total quantity sold per product. * Remaining stock for each product.   **(b) Restocking Costs**  To analyze restocking costs:  SELECT  SupplierID,  SUM(CostPrice \* StockQuantity) AS TotalRestockingCost  FROM  Products  WHERE  StockQuantity < ReorderLevel  GROUP BY  SupplierID;  This helps calculate the cost of restocking low-stock items.  **Automated Reports for the Accountant**   * Create **views** for frequently needed reports, e.g.,:   CREATE VIEW FinancialReport AS  SELECT  s.SaleID,  s.Date,  s.TotalAmount AS Revenue,  SUM(sd.QuantitySold \* p.CostPrice) AS CostOfGoodsSold,  (s.TotalAmount - SUM(sd.QuantitySold \* p.CostPrice)) AS Profit,  (s.TotalAmount \* 0.10) AS TaxCollected  FROM  Sales s  JOIN  SalesDetails sd ON s.SaleID = sd.SaleID  JOIN  Products p ON sd.ProductID = p.ProductID  GROUP BY  s.SaleID, s.Date, s.TotalAmount;  Accountants can query this view directly:  SELECT \* FROM FinancialReport WHERE Date BETWEEN '2024-11-01' AND '2024-11-15'; |
|  |  |

# 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)** [**here**](https://justit831-my.sharepoint.com/:u:/g/personal/danpe_justit_co_uk/Ef6vAaaYVi5FhHqKGxqnn60B9g2khoYekEIO3Y7J00UcJQ?e=pv9NNE)
2. **Follow each step to create your database** [**here**](https://justit831-my.sharepoint.com/:b:/g/personal/danpe_justit_co_uk/EdeCKl2Sas1Hl7u9amDy0fIB9jGVCKxSR0u2-lFOvS5rXw?e=xKv1U7)

**For each question I would like to see both the syntax used and the output.**

1. **Count Cities in USA:** *Scenario:* You've been tasked with conducting a demographic analysis of cities in the United States. Your first step is to determine the total number of cities within the country to provide a baseline for further analysis.

|  |
| --- |
| select count(\*) from city  where CountryCode = 'USA';  Result: 274 |

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

|  |
| --- |
| select Name from country  order by LifeExpectancy Desc;  Result: Andorra |

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 from country  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 from City  where Population > 2000000;   |  | | --- | | Alger | | Luanda | | Buenos Aires | | Sydney | | Melbourne | | Dhaka | | São Paulo | | Rio de Janeiro | | Salvador | | Belo Horizonte | | Fortaleza | | London | | Santiago de Chile | | Guayaquil | | Cairo | | Alexandria | | Giza | | Madrid | | Cape Town | | Addis Abeba | | Quezon | | Jakarta | | Surabaya | | Bandung | | Mumbai (Bombay) | | Delhi | | Calcutta [Kolkata] | | Chennai (Madras) | | Hyderabad | | Ahmedabad | | Bangalore | | Baghdad | | Teheran | | Roma | | Tokyo | | Jokohama [Yokohama] | | Osaka | | Nagoya | | Nairobi | | Shanghai | | Peking | | Chongqing | | Tianjin | | Wuhan | | Harbin | | Shenyang | | Kanton [Guangzhou] | | Chengdu | | Nanking [Nanjing] | | Changchun | | Xi´an | | Dalian | | Qingdao | | Jinan | | Hangzhou | | Zhengzhou | | Shijiazhuang | | Santafé de Bogotá | | Cali | | Kinshasa | | Pyongyang | | Seoul | | Pusan | | Inchon | | Taegu | | La Habana | | Casablanca | | Ciudad de México | | Rangoon (Yangon) | | Abidjan | | Karachi | | Lahore | | Lima | | Paris | | Bucuresti | | Berlin | | Riyadh | | Jedda | | Singapore | | Taipei | | Bangkok | | Istanbul | | Ankara | | Izmir | | Kyiv | | Toskent | | Moscow | | St Petersburg | | Ho Chi Minh City | | New York | | Los Angeles | | Chicago | |

1. **Cities Beginning with 'Be' Prefix:** *Scenario:* A travel blogger is planning a series of articles featuring cities with unique names. You're tasked with compiling a list of cities from the database that start with the prefix 'Be' to assist in the blogger's content creation process.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| select Name from City  where Name like 'Be%';   |  | | --- | | Béjaïa | | Béchar | | Benguela | | Berazategui | | Belize City | | Belmopan | | Belo Horizonte | | Belém | | Belford Roxo | | Betim | | Bento Gonçalves | | Belfast | | Benoni | | Bekasi | | Bengkulu | | Belgaum | | Bellary | | Berhampore (Baharampur) | | Beawar | | Bettiah | | Beerseba | | Bene Beraq | | Bergamo | | Beppu | | Beograd | | Benxi | | Bengbu | | Bei´an | | Beipiao | | Beihai | | Bello | | Beirut | | Bengasi | | Beni-Mellal | | Beau Bassin-Rose Hill | | Benito Juárez | | Bender (Tîghina) | | Beira | | Benin City | | Bergen | | Besançon | | Berlin | | Bergisch Gladbach | | Bern | | Berdjansk | | Berdytšiv | | Belgorod | | Berezniki | | Beaumont | | Bellevue | | Berkeley | |

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 City;  where Population between 50000 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 City order by Name asc;   |  | | --- | | [San Cristóbal de] la Laguna | | ´s-Hertogenbosch | | A Coruña (La Coruña) | | Aachen | | Aalborg | | Aba | | Abadan | | Abaetetuba | | Abakan | | Abbotsford | | Abeokuta | | Aberdeen | | Abha | | Abidjan | | Abiko | | Abilene | | Abohar | | Abottabad | | Abu Dhabi | | Abuja | | Acámbaro | | Acapulco de Juárez | | Acarigua | | Accra | | Achalpur | | Acheng | | Acuña | | Adamstown | | Adana | | Addis Abeba | | Adelaide | | Aden | | Adiyaman | | Ado-Ekiti | | Adoni | | Afyon | | Agadir | | Agaña | | Agartala | | Agege | | Ageo | | Agra | | Águas Lindas de Goiás | | Aguascalientes | | Ahmadnagar | | Ahmadpur East | | Ahmedabad | | Ahome | | Ahvaz | | Aix-en-Provence |   .  .  . |

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 name, Count(\*) as 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 City  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 capital  FROM 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 Name, LifeExpectancy  FROM 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.

|  |
| --- |
| select city.name, country.continent FROM City inner join country on city.CountryCode = country.Code 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 avg(population)as averagePopulation, countryCode from city group by countryCode; |

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 city.name, city.population from city inner join country on city.id = country.capital order by city.population asc;    .  .  . |

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 from country where population <> 0 order by population 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, country.GNP from country inner join city on city.countryCode = country.Code where country.gnp > (select avg(gnp) from country); |

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 \* from (  select ROW\_NUMBER() over (order by population) as ranking, name, population from city) as subquery  where ranking between 30 and 40; |

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