

**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? | Primary key is a column or group of columns that uniquely identities the record in that table. It cannot be a null values and it’s often used to create **relationships** with other tables. |
| How does this differ from a secondary key? | A secondary key (also known as an alternate key or secondary index ) is a column that is not the primary identifier but is still used to retrieve data efficiently. Secondary keys do not have to be unique and can contain null values. They are mainly used to support searching, sorting, or filtering operations on the table, often improving query performance.  Primary key is essential for uniquely identifying and linking records whereas a secondary key is primarily used for optimizing data access and queries. |
| How are primary and foreign keys related? | A foreign key is a column in one table that refers to the primary key in another table. It ensures that the value in the foreign key column matches a value in the referenced primary key column.  They are closely related because they help establish relationships between tables. |
| Provide a real-world example of a one-to-one relationship | Employee -> Security Clearance is the best example for one-to-one relationship.  Each employee may be assigned exactly one unique security clearance. Similarly, each security clearance record belongs to only one employee.  You can’t assign the same clearance to multiple employees. |
| Provide a real-world example of a one-to-many relationship | One airport serves many passengers. But each passenger typically checks in or departs from one airport at a time. So it’s a one-to-many relationship:  One Airport → Many Passengers |
| Provide a real-world example of a many-to-many relationship | Flights and Passengers.   * A passenger can take multiple flights. * A flight can have many passengers.   So:   * One passenger → many flights * One flight → many passengers |

# 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 databases** require **structured data**, organized into **tables with rows and columns**, and follow a **strict schema.**  Non-relational (NoSQL) databases can handle unstructured or semi-structured data, such as JSON, XML, images, audio, video, or CSV files.  Relational databases typically use vertical scalability (scale up by adding more power to one server — CPU, RAM).  Non-relational databases are designed for horizontal scalability (scale out by adding more servers or nodes), which is ideal for handling large, distributed datasets.  Non-relational databases like MongoDB, Cassandra, or Redis are often used for real-time analytics, especially in big data and IoT contexts due to their speed and scalability.  In relational databases, data analysis often happens on well-prepared (structured and clean) data — this means data is cleaned and organized first (data onboarding), and then analysis follows using SQL queries, joins, and aggregations.  In non-relational databases, data can be ingested quickly, even if it’s not clean or structured, and analysis can happen afterward using flexible tools. |
| What type of data would benefit off the non-relational model?  Why? | Unstructured and Semi-Structured Data-  JSON, XML files, logs , social media posts and whatsapp/messager message, videos, photos and audio.  Because it doesn't require a predefined schema. Also,you can choose the best structure (document, key-value, graph) based on your use case. |

# 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 join where a table is joined **with itself** to compare rows within the same table.  It’s useful in **hierarchical** data like employees and managers, or comparing records.  Aliases (e.g., A, B) are used to distinguish the same table.  SELECT A.col, B.col   FROM table A   JOIN table B ON A.common\_field = B.common\_field;  Example: Match employees with their managers using ManagerID = EmployeeID.  Supports INNER, LEFT, or other join types depending on requirements.  Ideal for tasks like identifying relationships, comparing duplicates, or building org charts. |
| Right join | A **RIGHT OUTER JOIN** returns **all rows from the right table** and matching rows from the left table.If no match is found in the left table, **NULLs** are shown for its columns.  SELECT A.col, B.col  FROM tableA A  RIGHT JOIN tableB B ON A.key = B.key;  Use it when you want to keep **all records from the right table** regardless of matches.  Example: List all departments (right table) and employees (left table), even if a department has no employees.  It’s the **opposite of LEFT JOIN** and helps ensure completeness from the right side.  Useful in **reporting**, **data auditing**, and **finding unmatched entries** |
| Full join | A **FULL OUTER JOIN** returns **all rows from both tables**, matching them where possible.If there’s **no match**, NULLs are shown for missing columns from either side.  SELECT A.col, B.col  FROM tableA A  FULL OUTER JOIN tableB B ON A.key = B.key;  Combines the effect of **LEFT JOIN** and **RIGHT JOIN**.  Example: Show all customers and all orders, even if some customers haven’t ordered and some orders aren’t linked.  Useful for **merging datasets**, **comparing records**, and ensuring **no data is lost**.  Not all SQL databases support FULL JOIN directly (e.g., MySQL), but it can be emulated with UNION. |
| Inner join | An **INNER JOIN** returns **only the matching rows** from both tables based on a condition.It excludes rows that don’t have matches in both tables.  SELECT A.col, B.col  FROM tableA A  INNER JOIN tableB B ON A.key = B.key;  It’s the **most common join type** used in SQL queries.  Example: Get employees with their department names by joining on DepartmentID.  Improves efficiency by filtering out unrelated data.  Can be written simply as JOIN (default is INNER JOIN). |
| Cross join | A **CROSS JOIN** returns the **Cartesian product** of two tables — every row of one table is combined with every row of the other.  If table A has m rows and table B has n rows, the result will have m × n rows.  SELECT A.col, B.col  FROM tableA A  CROSS JOIN tableB B;  **No ON condition** is used (unlike other joins).It’s useful for **combinations**, like generating test data or all possible pairings.  Use cautiously — output size grows quickly with large tables.  Note :Also called a **Cartesian Join.** |
| Left join | A **LEFT OUTER JOIN** returns **all rows from the left table**, and matching rows from the right table.  If there's **no match** in the right table, NULLs are returned for its columns.  SELECT A.col, B.col  FROM tableA A  LEFT JOIN tableB B ON A.key = B.key;  Ensures no data is lost from the **left table**.  Example: Show all employees (left) and their assigned projects (right), even if some have none.  Common in **reporting**, **data reconciliation**, and finding **unmatched** entries.Also called **LEFT JOIN** (the word "OUTER" is optional). |

# Day 4: Task 1: 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.

|  |
| --- |
| -- No of cities in USA  select count(distinct city.name) as CitiesinUSA,country.Name as CountryName  from city join country on city.CountryCode = country.Code  where country.Name='United States';  -- Another way with subquery  select count(distinct name) as CitiesinUSA from city  where CountryCode IN( select distinct Code from country where Name='United States');  **Output:** |

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 as CountryName, LifeExpectancy  FROM country  where LifeExpectancy = (SELECT MAX(LifeExpectancy) FROM country);  Another method:  select max(LifeExpectancy) as HighestLifeExpectancy, Name as CountryName  from country  group by Name  order by HighestLifeExpectancy desc;  Output: |

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 \*  FROM world.city  order by Population desc  limit 10; |

1. **Cities with Population Larger than 2,000,000:** *Scenario:* A real estate developer is interested in cities with substantial population sizes for potential investment opportunities. You're tasked with identifying cities from the database with populations exceeding 2 million to focus their research efforts.

|  |
| --- |
| SELECT \*  FROM world.city  WHERE Population > 2000000  ORDER BY Population DESC; |

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

|  |
| --- |
| SELECT \*  FROM world.city  where Name like 'Be%'  order by Population desc; |

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, CountryCode, Population  FROM world.city  WHERE Population BETWEEN 500000 AND 1000000  ORDER BY Population DESC; |

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 AS CityName, CountryCode, Population  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 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 AS CityName, COUNT(Name) AS OccurrenceCount  FROM world.city  GROUP BY Name  -- having count(Name) >3  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 AS CityName, CountryCode, Population  FROM world.city  ORDER BY Population ASC  LIMIT 10; |

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 AS CountryName, Population  FROM world.country  ORDER BY Population DESC  LIMIT 1; |

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

|  |
| --- |
| SELECT c.Name AS Country, ci.Name AS CapitalCity  FROM world.country c  JOIN world.city ci ON c.Capital = ci.ID  WHERE c.Name = 'Spain'; |

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 AS City, country.Name AS Country, country.Continent  FROM world.country  JOIN world.city ON city.CountryCode = country.Code  WHERE country.Continent = 'Europe'  ORDER BY country.Name, city.Name; |

1. **Average Population by Country:** *Scenario:* A demographic research team is conducting a comparative analysis of population distributions across countries. You're tasked with calculating the average population for each country from the database to provide valuable insights into global population trends.

|  |
| --- |
| SELECT country.Name AS CountryName, AVG(city.Population) AS AvgCityPopulation  FROM world.city  JOIN world.country ON city.CountryCode = country.Code  GROUP BY country.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 country.Name AS CountryName,city.Name AS CapitalCity,city.Population AS CapitalPopulation  FROM world.country AS country  JOIN world.city AS city ON country.Capital = city.ID  ORDER BY city.Population DESC; |

1. **Countries with Low Population Density:** *Scenario:* An agricultural research institute is studying countries with low population densities for potential agricultural development projects. You're tasked with identifying countries with sparse populations from the database to support the institute's research efforts.

|  |
| --- |
| SELECT Name AS CountryName,Population,SurfaceArea,  (Population / SurfaceArea) AS PopulationDensity  FROM world.country  -- WHERE (Population / SurfaceArea) > 100  ORDER BY PopulationDensity 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 AS CityName,country.Name AS CountryName,country.GNP,country.Population AS CountryPopulation,(country.GNP / country.Population) AS GNP\_per\_Capita FROM world.city AS city JOIN world.country AS country ON city.CountryCode = country.Code WHERE (country.GNP / country.Population) > (SELECT AVG(GNP / Population) FROM world.country WHERE GNP IS NOT NULL AND Population > 0) ORDER BY GNP\_per\_Capita DESC; |

1. **Display Columns with Limit (Rows 31-40):** *Scenario:* A market research firm requires detailed information on cities beyond the top rankings for a comprehensive analysis. You're tasked with providing data on cities ranked between 31st and 40th by population to ensure a thorough understanding of urban demographics.

|  |
| --- |
| SELECT \*  FROM world.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.**