

**Data Technician**

|  |
| --- |
|  |

|  |
| --- |
| Name: |
| Course Date: |
|  |

**Table of contents**

[Day 1: Task 1 3](#_Toc193099469)

[Day 1: Task 2 4](#_Toc193099470)

[Day 3: Task 1 4](#_Toc193099471)

[Day 4: Task 1: SQL Practical 6](#_Toc193099472)

[Day 4: Task 2: Written (Optional) 14](#_Toc193099473)

[**Course Notes** 18](#_Toc193099474)

[**Additional Information** 19](#_Toc193099475)

# Day 1: Task 1

Please research and complete the below questions relating to key concepts of databases.

|  |  |
| --- | --- |
| What is a primary key? | A unique identifier in a specific column in a table to identify each row |
| How does this differ from a secondary key? | Primary cannot contain NULL, secondary can. Only one primary allowed, secondary: multiple allowed. Ensures uniqueness for specific columns (other than primary column) |
| How are primary and foreign keys related? | Foreign keys refer to the primary key of another table creating a relationship between the two tables |
| Provide a real-world example of a one-to-one relationship | A first name in one table may relate to only one last name in another table |
| Provide a real-world example of a one-to-many relationship | One cooking ingredient in one table may relate to many recipes in another table |
| Provide a real-world example of a many-to-many relationship | Students in one table can enrol in multiple courses in another table and one course can have multiple students enrolled in it |

# 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? | A relational database contains structured data while a non-relational database contains unstructured data. |
| What type of data would benefit off the non-relational model?  Why? | Unstructured data |

# 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 | Joins a table to itself to compare rows within the same table. Example: Find employees and their managers in an employee hierarchy |
| Right join | Returns all rows from the right table and matching rows from the left table. If no match, NULL is returned for left table columns. Example: List all orders and their associated customers |
| Full join | Returns all rows if there is a match in either table. If no match, NULL is returned for non-matching rows. Example: Find all customers and orders, including records from both tables. |
| Inner join | Returns only rows with matching values in both tables. Example: Find customers who have placed orders |
| Cross join | Returns the Cartesian product of both tables (all possible combination of rows). Example: All possible combinations of products and sizes for inventory planning. |
| Left join | Returns all rows from the left table and matching rows from the right table. If no match, NULL is returned for right table columns. Example: List all orders and their associated customers |

# 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.

|  |
| --- |
| select count(distinct name) from city where countrycode='usa';  result = 264 |

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 \* from country where lifeexpectancy = (select max(lifeexpectancy) from country);  select max(lifeexpectancy) from country;  Result: 83.5 |

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 \* 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 \* from 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 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 \* from 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 \* from 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 \* from 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 \* FROM city WHERE population = (SELECT Max(population) FROM city); |

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 count(c.name) as Count, c.name  from city c  group by c.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 c.name, c.population  from city c  order by c.population asc; |

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 c.name, c.population  from country c  where c.population = (select max(c.population) from country)  order by c.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 \*  from city  join country  on country.code = city.countrycode  where country.name = 'spain' and country.capital = city.id; |

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  from city  join country  on country.code = city.countrycode  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.

|  |
| --- |
| -- average population of cities per country  select avg(ci.population) as AvgPop, co.name as 'Country'  from country co  join city ci  on co.code = ci.countrycode  group by co.name  order by AvgPop desc;  -- average population of all countries  select avg(population)  from country; |

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 co.name as 'Country', ci.name as 'Capital', ci.population as 'City Pop'  from city ci  join country co  on co.code = ci.countrycode  where co.capital = ci.id; |

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 'Country', surfacearea as 'Area', population as 'Population', population/surfacearea as 'Population Density'  from country  where population>0  order by population/surfacearea;  -- Because I wanted to name the Density Column with a space, I had to make it into a string and then could not use it with ‘order by’ so had to recalculate the density within ‘order by’. The alternative would have been to use PopDense and then call ‘order by PopDense’ |

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.

|  |
| --- |
| -- for this task I got the logic correct but struggled with the “where” syntax comparing GNP per Cap of each city to average GNP per Cap. Asking AI for help, after a while, it gave me 2 interesting solutions that helped me learn new ways of writing queries. I’ve included them below.  My version:  select ci.name as 'City', co.gnp as 'GDP', ci.population as 'Pop', co.gnp/ci.population as gdpcap #'GDP/Cap' -- experimenting with and without ‘ ‘ trying to make it work  from city ci  join country co  on co.code = ci.countrycode  where co.gnp/ci.population > gdpcap = select avg(gdpcap);  #select avg(co.gnp/ci.population); -- unsure of the syntax after trying different options and nothing working  Ai version 1, nested:  SELECT ci.name AS 'City', co.gnp AS 'GNP', ci.population AS 'Population', (co.gnp / ci.population) AS 'GNP\_per\_capita'  FROM city ci  JOIN country co ON co.code = ci.countrycode  WHERE (co.gnp / ci.population) > ( SELECT AVG(gnp\_capita) FROM ( SELECT (co2.gnp / ci2.population) AS gnp\_capita FROM city ci2 JOIN country co2 ON co2.code = ci2.countrycode WHERE ci2.population > 0 -- Avoid division by zero ) AS avg\_calc ) ORDER BY GNP\_per\_capita DESC;  AI version 2, cleaner version by pre-calculating city gdp (gnp):  WITH city\_gnp AS ( SELECT ci.name AS city\_name, co.gnp, ci.population, (co.gnp / NULLIF(ci.population, 0)) AS gnp\_per\_capita FROM city ci JOIN country co ON co.code = ci.countrycode )  SELECT city\_name AS 'City', gnp AS 'GNP', population AS 'Population', gnp\_per\_capita AS 'GNP\_per\_capita' FROM city\_gnp WHERE gnp\_per\_capita > (SELECT AVG(gnp\_per\_capita) FROM city\_gnp) ORDER BY gnp\_per\_capita DESC;  -- AI even added zero division prevention which I had not considered in this instance |

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  name AS 'City',  population AS 'Pop',  DENSE\_RANK() OVER (ORDER BY population DESC) AS 'Rank'  FROM city  ) AS ranked\_cities  WHERE `Rank` BETWEEN 31 AND 40  ORDER BY `Rank`;  -- my own version only showed the results but not the ranking so i enquired and learned some new SQL Syntax:  -- learned about dense\_rank vs row\_number  -- learned about select version() to find out my version with regard to new syntax like dense\_rank, etc vs mysql5.7  -- learned that using select \* in outer query and then selecting visible columns in inner query is a common sql pattern for clarity and flexibility  -- can even rank within groups: DENSE\_RANK() OVER(PARTITION BY countrycode ORDER BY population DESC) AS 'Rank' |

# Day 4: Task 2: Written (Optional)

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 | Creating a sophisticated Database for your local Cornershop would benefit the owners of the shop as it would provide insights into products, customers and sales. The types of data the Database would be handling would range from strings and integers to floats and dates/times depending on what is available to the owners. Assuming that the shop is equipped with modern tils (and assuming the owners are able to extract that data from the tils which can be challenging), we can suggest a complex Database design.  One of the main tables in the Database would be the Categories table with columns such as food, toys, stationary, drinks, tobacco, pets, care, newsagents and other (for any non-fitting products). Each category would link to another table with subcategories e.g. food would contain frozen, fresh, sweets, cereal. These will then link to further subcategory tables e.g. ‘frozen’ containing meat, pizza, veg, ice-cream, etc. Those would then hold the item lists.  The products table (as the primary table) would hold information for each individual product with regard to sales, number of items in inventory and number of items sold. This would allow the owners to see profitability and margins per product. Furthermore, perishables could have their use-by dates stored and alerts (CREATE TRIGGER, CREATE EVENT) could be created for the owners to start offering reduced prices on products going off soon (or throwing products that have gone off). Commercial impact of gone-off products could be visualised and compared to profitability of the product in question. Buy and Sell dates could be stored and shelf time calculated to show which products take a long time to be sold and take up space on the shelves. In addition, missing items would stand out as discrepancies between items bought, stored and sold and could point to unrecognised theft. This would then be linked to the category hierarchy and patterns may emerge for certain categories which could relate to areas within the shop that attract more theft and are less visible and the financial impact of each allowing the owners to prioritize security measures.  A customer table would hold all available information about the punters. From name and age to their shopping behaviour: How much do they spend per visit or in total? How much profit do they contribute? What time of day and day of the week do they visit? These metrics could then be generalised and show what age group shops when and how much they actually contribute to the overall revenue. Customer addresses could be used to calculate distance to the shop and show the effective catchment area. Peak and quiet times would allow the owners to adjust staff levels throughout the day and the week as well as seasonality. This is also where a threshold could be calculated that triggers the loyalty program. This could depend on money spent or quantity of items purchased by the customers. Alerts could be triggered when staff should let the customer know when they are getting closer to a free product or a percentage discount, to entice the customer to shop more in the run up to fulfil the promotional requirements.  An employee Table would keep track of hours worked, start date of employment, hourly pay and maybe even a personal scoring system that determines the Christmas bonus.  In order to create the Database, we use the commands:  CREATE DATABASE CornershopDB;  USE CornershopDB; -- to start using the new Database  -- We then continue to create the tables:  -- Parent Table  CREATE TABLE Products (  Product\_id INT PRIMARY KEY AUTO\_INCREMENT,  Product\_name VARCHAR(50) NOT NULL,  Product\_Category VARCHAR(50) NOT NULL,  Product\_SubCategory VARCHAR(50) NOT NULL  -- etc  );  -- Child Table  CREATE TABLE Customers (  Customer\_id INT PRIMARY KEY AUTO\_INCREMENT,  Customer\_firstname VARCHAR(50) NOT NULL,  Customer\_lastname VARCHAR(50) NOT NULL,  Customer\_age INT,  -- Link to Products Table  Product\_id INT,  FOREIGN KEY (product\_id) REFERENCES Products(product\_id)  Ent\_id changes  );  -- Insert Data in Parent Table  INSERT INTO Products (product\_name, product\_category, product\_subcategory)  VALUES (‘Bounty’, ‘Sweets’,’Bar’), (‘Frosties’,’Cereal’,’Flakes’), (‘Peroni’,’Drinks\_Alc’,’Beer’);  -- Insert Data in Child Table  INSERT INTO Customers (customer\_firstname, customer\_lastname, customer\_age)  VALUES (‘Napoleon’, ‘Dynamite’,’18’), (‘Summer’,’Wheatly’,’18’), (‘Pedro’,’Sanchez’,’18’) (‘Rex’,’Kwando’,’35’);  Initially, the data may be in the form of CSV, xlsx, json or text logs. In order to import that data, we can import via the MySQL Workbench menu or use the following code:  LOAD DATA LOCAL INFILE ‘/path/.../ProductList.csv’  INTO TABLE Products  FIELDS TERMINATED BY ‘,’  ENCLOSED BY ‘ “ ‘  LINES TERMINATED BY ‘\n’  IGNORE 1 ROWS; -- Skip Header Row  To maintain accuracy and security, we would need to ideally automate the process of appending new data to the Database. In regular intervals we would back-up the existing Database before adding new data from the till (making sure the data comes in clean and appropriately structured) and checking for consistency by querying the data. As an additional backup, we can use constraints in the code (e.g. a product name must not be empty, or a product inventory must not be negative, etc. Constraints, checks and backups can all be automated with SQL. Further Security measures are encryption and access control (i.e. password protection). Alerts can be set for failed logins i.e. in case someone was trying to get past the password protection. Finally, a Disaster Recovery Plan should be in place in case of server failure, hacking or accidental deletion. |
|  |  |

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