

**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 uniquely identifies each record in a table. * And no two rows can have the same primary key, which helps to maintain data integrity. * It should be stable, meaning it shouldn't change over time. |
| How does this differ from a secondary key? | A secondary key is used for searching key and is generally used for searching , sorting or indexing data based on columns and that are not guaranteed to be unique.  A Secondary key as key that is used to access records in a database on columns that are not the primary key.   |  |  | | --- | --- | | **Primary Key** | **Secondary Key** | | **Unique** - Ensures each row in the table is uniquely identifiable | **Non-unique** - Secondary keys can allow duplicate values (e.g., multiple records can have the same value in the column associated with the secondary key) | | **Non-null:** Cannot have NULL values. | **Nullable**: Can allow NULL values, depending on how the database is set up. | | **Single or Composite**: It can be made of one or more columns (composite primary key). | **Multiple**: A table can have multiple secondary keys (indexes) to allow more efficient searching or sorting based on different columns. | | **One per table**: There can only be one primary key in a table. | **Multiple**: A table can have multiple secondary keys (indexes) to allow more efficient searching or sorting based on different columns. | |
| How are primary and foreign keys related? | * Primary key and foreign keys are both used to establish relationships between tables. * As mentioned earlier, a **primary key** uniquely identifies a record within a table. It ensures that each record can be distinguished from others and typically prevents duplicates or null values. * A **foreign key** is a column (or set of columns) in a table that creates a link to the **primary key** of another table. It is used to enforce referential integrity between two tables, ensuring that the value in the foreign key column corresponds to an existing record in the referenced table. * **Relationship Between Primary and Foreign Keys:**  1. **Primary Key**: The primary key in a table uniquely identifies each record within that table. It’s a way of ensuring that each row is distinct. 2. **Foreign Key**: The foreign key in another table refers to the primary key of the first table. The foreign key value must either match a primary key value from the referenced table or be NULL (if allowed by the database design). |
| Provide a real-world example of a one-to-one relationship | E.g. for one-one relationship  1. A person can have one passport. A passport is typically assigned to a single person and each person is generally issued only one passport. In this example person and the passport entities have one-to-one relationship, where each person linked to exactly one passport and each passport is linked to exactly one person.  2. Each person has one unique NI number assigned to each person. |
| Provide a real-world example of a one-to-many relationship | E.g. for one-to-many relationships  1. Customers can order many products and each order is associated with one customer.  2. A director can shoot many movies, but one movie is directed by one director only (may be some movies has multiple directors)  3. A Class belongs to many students , but each student belongs particular Class in school. |
| Provide a real-world example of a many-to-many relationship | E.g. for many-many relationships  1. A student can learn many courses and one course can have many students.  2. A movie can have many actors and an actor can act in many movies.  3. Sports club has many members and members can join different clubs for different sports. |

# 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 management systems (RDBMS):** This is the most common database type which allows us to organize the data into table with rows and columns. The popular RDBMS are PostgreSQL, MySQL, Oracle Database, SQL Server and IBM Db2.  **When to use an SQL database:**   * You need highly structured data distributed across multiple tables. You need your data to adhere to a strict, predictable, predefined, and already planned schema. * Your data will remain relatively the same. SQL databases are convenient if you don't plan on frequently changing the structure of the database and don't need to regularly update items. Keep in mind that they offer little flexibility. * You need consistent data. * Data integrity and security are a priority. * You want accurate results for complex queries. * A disadvantage of SQL databases is that they scale vertically. * And an increase in processing power and memory storage is needed to handle an increase in load to improve performance.   **Non- relational Database systems (NoSQL databases):** These types of databases stores data as documents. The popular document databases are MongoDB, Databricks and Amazon Dynamo DB.  There are four major types of non-relational databases:   * Column oriented databases * Key - value data stores * Document - oriented stores * Graph oriented databases   **When to use a NoSQL database:**   * You are working in a fast development environment that requires frequent adaptations of requirements and constant changes to the database structure. * You are working with large amounts of data that are diverse in nature but do not require a lot of structure or accuracy. * You are working with data that needs frequent updates. NoSQL databases offer a loose, flexible, and dynamic schema that allows for regular changes to the data. * You want speedy query results and continuous availability of the system. * You don't want to perform any upfront planning, preparing, or designing of the database, but want to immediately start building instead. * A big advantage of NoSQL databases is that they scale horizontally and a disadvantage of NoSQL databases is that they do not ensure data integrity and consistency. |
| What type of data would benefit off the non-relational model?  Why? | When to use these non-relations DB’s are when we have flexible data models to handle unstructured or semi-structure data , Scalability (can distribute data across multiple servers or clusters) , High Performance with simple queries, Rapid Development.   * For example Social media posts, multimedia content (images, videos), sensor data etc.,   Why because non-relational database like document based or key value stores are flexible when it comes to storing unstructured or semi-structured data. These systems don’t require a predefined schema, making it easier to store diverse data types (example text, images, JSON objects)   * And Internet of Things (IoT) data and big data applications, why because these are designed to scale horizontally across multiple servers, which is crucial for handling large volumes of 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 | A **Self Join** allows to join a table with itself. This allows us to compare rows within the same table.  A Self Join is often used in scenarios where there is hierarchical or relational data within the same table like comparing related rows from a single table, such as when one employee reports to another in an organizational structure.  **Example:** SQL Self Join to retrieve Employees and their Managers. |
| Right join | A **Right Join** returns all records from the right table (table2), and the matching records from the left table (table1).  If there is no match, the result is NULL from the left side.  **Example :**   * Join Customers and Orders tables * Sharing customer\_id columns * Customers is the left table * Orders is the right table   SELECT Customers.customer\_id, Customers.first\_name, Orders.item  FROM Customers  RIGHT JOIN Orders  ON Customers.customer\_id = Orders.customer\_id;  In this example, the query returns all customers and any orders they might have placed. If a customer has not placed any orders, their details will still be included, but with NULL values for order-related columns. |
| Full join | The **Full Join** joins two tables based on a common column. It selects records that have matching values in these columns and the remaining rows from both of the tables.  **Example:** full join Customers and Orders tables  -- based on their shared customer\_id columns  -- Customers is the left table  -- Orders is the right table  SELECT Customers.customer\_id, Customers.first\_name, Orders.item  FROM Customers  FULL OUTER JOIN Orders  ON Customers.customer\_id = Orders.customer\_id; |
| Inner join | The **INNER JOIN** keyword selects records that have matching values in both tables.  **Example:**  - join Customers and Orders tables with their matching fields customer\_id  - join the Customers and Orders tables  - with customer\_id and customer fields  SELECT Customers.customer\_id, Customers.first\_name, Orders.amount  FROM Customers  INNER JOIN Orders  ON Customers.customer\_id = Orders.customer;  Returns the customer id’s who are matching in both tables with first name and order amount. |
| Cross join | The **CROSS JOIN** operation allows us to combine rows from two or more tables without any specific relationship between them.  **Example:** Take the Customers and Orders tables.  SELECT Customers.customer\_id, Customers.first\_name, Orders.order\_id  FROM Customers  CROSS JOIN Orders;  This creates a Cartesian product of all customer IDs and first names with order IDs.  The result is a combination of every customer with every order. |
| Left join | The **LEFT JOIN**  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.  **Example:**  -- left join Customers and Orders tables based on their shared customer\_id columns  -- Customers is the left table  -- Orders is the right table  SELECT Customers.customer\_id, Customers.first\_name, Orders.item  FROM Customers  LEFT JOIN Orders  ON Customers.customer\_id = Orders.customer\_id;  The result selects the customer\_id and first\_name from Customers and the amount from Orders.  Hence, the result includes rows where customer\_id from Customers matches customer from Orders. |

# 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 data that needs to be stored by the business is data regarding the products, sales and customer information. Data about the products should include current stock of the product, a product description (name and brand), pricing, as well as the category of the product. For example, fruit and veg or meats. The sales data should include the number of times a product has been sold to enable the tracking of stock levels, this would also allow calculations to work out the total sales from the number of stock sold times their price. Data for customers should include customer information such as Name, contact details such as a number or email to keep in contact and ID for the loyalty program that the business wants to set up. The user of the database will most likely be the owner of the business who oversees most things such as stock management as well as managing pricing of products. Their job would also be to update the sales table and any relevant information to keep track of stock sold and see performance numbers for their profit.   1. **Designing the Database Schema**:   For structure I would create 5 tables 1. Customers Table We need to store customer details like personal information and loyalty program data. customer\_id (Primary Key): A unique identifier for each customer. first\_name last\_name email: Used for communication or promotional purposes. phone\_number loyalty\_points: Tracks how many loyalty points the customer has earned. date\_joined: When they joined the store's loyalty program. This table is focused on customer demographics and loyalty data. 2. Products Table This is where we track the inventory of products, including stock quantities, price, and product details. product\_id (Primary Key): Unique identifier for each product. product\_name category: Categorizes the product (e.g., groceries, beverages, domestic products). description: A description of the product. price: The unit price of the product. stock\_quantity: Tracks how many units are available. supplier\_id (Foreign Key, optional): If the store buys products from suppliers, this would link to a Supplier table. 3. Sales Table This table captures overall sales transactions. Each sale will link to a customer (if applicable), and track the total amount. sale\_id (Primary Key): A unique identifier for each sale. customer\_id (Foreign Key to Customers): This links each sale to a customer. sale\_date: The date and time when the sale took place. total\_amount: The total price for this sale transaction. 4. Sale Details Table This table records each individual product sold as part of a sale. This allows us to track the line items in a sale. sale\_detail\_id (Primary Key): Unique identifier for each line item. sale\_id (Foreign Key to Sales table): Links to the specific sale. product\_id (Foreign Key to Products): Links to the product sold. quantity: How many units of the product were sold. price\_per\_unit: The price of the product at the time of sale. 5. Inventory Transactions Table This table tracks inventory adjustments, such as restocking, returns, or sales-related stock changes. transaction\_id (Primary Key): Unique identifier for each inventory adjustment. product\_id (Foreign Key to Products table): Links to the product whose stock is adjusted. transaction\_type: Type of transaction (e.g., "restock," "sale," "return"). quantity: The number of units affected. transaction\_date: When the transaction occurred.   1. **Implementing the Database**   **Creating the database:**  CREATE DATABASE RetailStoreDB;  Before creating tables, ensure you are working within the RetailStoreDB database:  **USE RetailStoreDB;**  **Creating Tables** **Products, Sales, SaleDetails, Customers, InventoryTranmissons**, and **Suppliers.**  **CREATE TABLE Customers** ( Customer\_Id INT PRIMARY KEY AUTO\_INCREMENT,  First\_Name VARCHAR(50),  Last\_Name VARCHAR(50),  email VARCHAR(100) UNIQUE,  Phone\_Number VARCHAR(15),  Address VARCHAR(20) ,  Loyalty Points INT,  Date\_Joined DATETIME DEFAULT CURRENT\_TIMESTAMP);  **CREATE TABLE Products(** Product\_Id PRIMARY KEY AUTO\_INCREMENT,  P\_Name VARCHAR(20) NOT NULL,  P\_Desc TEXT,  Price DOUBLE NOT NULL,  Quantity\_in\_stock INT NOT NULL,  Category VARCHAR(100),  Supplier\_Id INT FOREIGN KEY (supplier\_id) REFERENCES Suppliers(supplier\_id  );  **CREATE TABLE Sales** ( Sale\_Id INT PRIMARY KEY AUTO\_INCREMENT,  Customer\_Id INT,  Sale\_Date DATETIME DEFAULT CURRENT\_TIMESTAMP,  Total\_Amout DECIMAL(10, 2) NOT NULL,  FOREIGN KEY (Customer\_Id) REFERENCES Customers(Customer\_Id)  );  **CREATE TABLE SaleDetails** ( Sale\_Detail\_Id INT PRIMARY KEY AUTO\_INCREMENT,  Sale\_Id INT,  Product\_Id INT, quantity INT NOT NULL,  Price\_Per\_Unit DECIMAL(10, 2) NOT NULL,  FOREIGN KEY (Sale\_Id) REFERENCES Sales(Sale\_Id),  FOREIGN KEY (Product\_Id) REFERENCES Products(Product\_Id) );  **CREATE TABLE** **InventoryTransactions** ( Transaction\_Id INT PRIMARY KEY AUTO\_INCREMENT,  Product\_Id INT, Transaction\_Type VARCHAR(50) NOT NULL,  Quantity INT NOT NULL,  Transaction\_Date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,  FOREIGN KEY (Product\_id) REFERENCES Products(Product\_id) );   1. **Populating the Database**   Inserting values into Tables:  **INSERT INTO** **Customers** (First\_Name, Last\_Name, email, Phone\_Number, Loyalty\_Points, Date\_Joined ) VALUES ('John', 'Doe', 'john.doe@example.com', '555-1234', 150 , '20-01-2025 10:05:00'), ('Daniel', 'Smith', 'jane.smith@example.com', '555-5678', 230, ‘21-01-2025 16:30:00'), ('George', 'Jone', 'alice.johnson@example.com', '555-9876', 320, ‘25-02-2025 07:30:00');  **INSERT INTO Products** (Name, Brand, Category, Description, Price, Stock) VALUES ('Apple', 'GoGreen', 'Fruit', 'Fresh British Apples', 1.00, 100, 001), ('Chicken Breast', 'UKForm', 'Meat', 'Boneless chicken', 5.00, 50,002) (‘Carrots’, ‘FreshVeg’, ‘Vegetable’, ‘Organic Carrots’, 2.00, 30,003) ;  **INSERT INTO Sales** (Customer\_Id, Total\_Amount) VALUES (1, 25.50), (2, 15.00), (3,50,75);  **INSERT INTO SaleDetails** (Sale\_Id, Product\_Id, Quantity, Price\_Per\_Unit) VALUES (1, 1, 5, 1.50), (1, 2, 2, 0.99) (2,2,3,.89);  **Sale\_Id** is an auto-incrementing primary key. **Product\_Id** is a foreign key, linking to the **Products** table to identify which product was sold.  **INSERT INTO InventoryTransactions** (Product\_Id, Transaction\_Type, Quantity, Transaction\_Date) VALUES (1, 'sale', 5, '11-02-2025 14:00:00'), (2, 'sale', 2, ’12-01-2025 16:30:00'), (3, 'sale', 10, ’20-01-2025 18:00:00');   1. **Maintaining the Database :**   Maintaining a database involves various tasks to ensure its optimal performance, data integrity, and security.  **Data Integrity and Accuracy:**  **Data Validation**: Ensure that only accurate and valid data is entered into the database. For example:   * + Stock levels cannot be negative.   + Sales quantities should be a positive integer.   + Customer contact details should adhere to proper formats.   **Constraints**:   * + Use primary keys (like Product\_ID, Customer\_ID, and Sale\_ID) to ensure uniqueness.   + Use foreign keys (e.g., Product\_ID in the Sales table) to maintain relationships and data consistency.   + Use CHECK constraints for validation (e.g., ensure Stock\_Quantity >= 0).   **Updating Data:**   * + Updating Stock Levels: After every sale, the business owner needs to adjust the stock. For example:   + After a sale is recorded in the Sales table, the Stock\_Quantity in the Products table should be updated to reflect the remaining stock   **Regular Backups :**   * + Backing up your data ensures that in the event of failure, corruption, or loss, you can restore your database to a previous state.   + Schedule automatic daily backups of the database to ensure data is not lost due to system failures. Ensure both full backups and incremental backups are in place.   + Store backups offsite or in cloud storage to protect against physical disasters.   **Security :**   * + **Access Control**: Restrict access to the database by user roles. For example:   + The business owner may have full access (CRUD operations on all tables).   + Sales staff may only have access to the Sales table, and customer support may only have access to the Customers table.   + **Encryption**: Encrypt sensitive data (e.g., customer email addresses, phone numbers) at rest and during transmission. This will help prevent unauthorized access.   **Reporting and Analytics:**   * + **Automated Reports**:     - Set up automated reporting systems to track key business metrics such as:     - Sales performance (e.g., total sales, revenue, most sold products).     - Stock levels and reordering needs.     - Customer activity (e.g., loyalty points earned, customer retention).   + **Custom Queries**: The business owner can use custom SQL queries to generate insights.   **Monitoring and Alerts:**   * + **Performance Monitoring**: Use monitoring tools to track the performance of the database (e.g., query response times, server CPU/memory usage).   + **Alerts**: Set up alerts for certain conditions such as low stock levels or sales exceeding a threshold. This can be done through database triggers or external monitoring systems.   **Compliance:**   * + **GDPR Compliance**: If you're dealing with customer data, ensure compliance with regulations like GDPR, especially when handling personal data (e.g., email addresses, phone numbers).   + **Data Retention Policies**: Define and enforce data retention policies. For instance, customer records should only be retained for a certain period before being archived or deleted, depending on business needs and legal requirements.   **Scaling the Database:**   * + As the business grows, you may need to scale the database. Consider options like database clustering, read replicas, or sharding to handle increased data volume. |

# 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.

|  |
| --- |
|  |

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.

|  |
| --- |
| If we want to the count of cities with population> 200000 |

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.

|  |
| --- |
| If we want count the how many cities starts with ‘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.

|  |
| --- |
| Again if want the count of the above query: |

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. **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.

|  |
| --- |
|  |

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. **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.**