

Data Technician

Name:

Course Date:

Table of contents

Day 1: Task 1 3

Day 1: Task 2 4

Day 3: Task 1 4

Day 4: Task 1: Written..... 6

Day 4: Task 2: SQL Practical..... 10

Course Notes..... 31

Additional Information..... 32



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 column or set of columns in a database table that uniquely identifies each row, ensuring each record is distinct and prevents duplicate data |
| How does this differ from a secondary key? | A primary key is a special column (or set of columns) that uniquely identifies every row in a table, with only one primary key per table, and it cannot contain NULL values. A secondary key (or alternate key) is any other field or combination of fields in the table that could also uniquely identify rows but wasn't chosen as the primary key |
| How are primary and foreign keys related? | Primary and foreign keys are related because a foreign key in one table points to a primary key in another table, establishing a link between the two and ensuring data integrity. |
| Provide a real-world example of a one-to-one relationship | A one-to-one relationship example is a school database, each student has only one student ID, and each student ID is assigned to only one person. |
| Provide a real-world example of a one-to-many relationship | A one-to-many relationship example is a blog can have many associated posts, but each Post is associated with only one Blog. |
| Provide a real-world example of a many-to-many relationship | A many-to many relationship example is one between students and classes. A student can register for many classes, and a class can include many students. |



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 store data in structured tables with predefined schemas, ensuring strong consistency and integrity (like in SQL databases), while non-relational (NoSQL) databases use more flexible, non-tabular formats like documents or graphs to store varied data types |
| What type of data would benefit off the non-relational model? Why? | <p>Non-relational databases (NoSQL) are best suited for unstructured, semi-structured, or rapidly changing data that doesn't fit neatly into tables and rows.</p> <p>Social media posts, emails, sensor data, logs, documents (JSON, XML), photos, and videos are a few examples of unstructured or semi-structured data.</p> <p>Why: There is no set schema for these data types. Flexible, schema-less documents can be stored in NoSQL databases (such as Couchbase or MongoDB), making it simple to add new fields or modify the data structure without affecting already-existing entries.</p> |

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 | <p>Self-Join is a type of a JOIN query used to compare rows within the same table.</p> <p>An example of an SQL self-join is a query that retrieves employees and their managers. By joining the Employees table to itself by using their aliases for the employee and managers roles and a left join on the manager column.</p> |
|------------------|---|



| | |
|-------------------|--|
| | |
| 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. |
| Full join | Full join is a join operation that retrieves all records from both tables, including matching and non-matching records. If there are no matching records in one or both tables, NULL values are included for those columns. |
| Inner join | Inner joins combine records from two tables whenever there are matching values in a field common to both tables. You can use Inner join with the Departments and Employees tables to select all the employees in each department. |
| Cross join | Cross join is a join operation that produces the Cartesian product of two tables. Unlike other Join operators, it does not let you specify a join clause. You may, however, specify a WHERE clause in the Select statement. |
| Left join | Left Join is a type of Join operation that retrieves all records from the left table (table1) and the matching records from the right table (table2). If there are no matching records in the right table, NULL values are included for those columns. |

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:**
 - a. What kind of data will the database need to store?
 - b. Who will be the users of the database, and what will they need to accomplish?
2. **Designing the Database Schema:**
 - a. How would you structure the database tables to efficiently store inventory, sales, and customer information?
 - b. What relationships between tables are necessary (e.g., how sales relate to inventory and customers)?
3. **Implementing the Database:**
 - a. What SQL commands would you use to create the database and its tables?
 - b. Provide examples of SQL statements for creating tables and defining relationships between them.
4. **Populating the Database:**
 - a. How would you input initial data into the database? Give examples of SQL INSERT statements.
5. **Maintaining the Database:**
 - a. What measures would you take to ensure the database remains accurate and up to date?
 - b. 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

1a) The types of data the database will need to store would be product/inventory data to track items in stock, item price and knowing when to re-order given the reorder level has been reached

2a) In terms of structuring the database, I will break this part into three sections relative to the tables

The first table will be centred around products sold by the retailer with fields such as

Product_id – INT – Unique identifier for each product

Name – VARCHAR – Product Name

Category – VARCHAR – “Beverages”, “Snacks”, “Toiletries” etc.

Brand – VARCHAR – Something that can help in terms of filtering

Unit_Price – DECIMAL – Sales price per unit

Stock_Quantity – INT – Current stock count

Reorder_Level – INT – Threshold to trigger reorder

Supplier_Id – INT – Links to Supplier Table

The Second table will be about Customer Information relative to The loyalty program implemented by the retailer

Customer_Id – INT – Unique Identifier

Name – VARCHAR – Customer name

Phone – VARCHAR – Useful for the lookup for loyalty program

email – VARCHAR – optional as an alternative form of contact

Join_Date – DATE – When loyalty membership started

Loyalty_Points – INT – Current points balance

The Third table will be a sales/transactions tailored where the fields contain the following

Sales_Id – INT – Unique Transaction Identifier

Customer_Id – INT – Nullable (supports guest sales)

Sale_Date – DATETIME – Timestamp of transaction

Total_Amount – DECIMAL – Total transaction value

Points_Added – INT – Points accrued from purchase



Points_Redeemed – INT – Points used in sale (If any)

The Fourth table will contain Information about Items on sale with the following fields

Sale_Item_Id – INT – Line Item Identifier

Sale_Id – INT - Links to sales table

Product_Id – INT – Product Sold

Quantity – INT – Number of units sold

Unit_Price_At_Sale – DECIMAL – Price at the time of transaction (may change later)

Subtotal – DECIMAL – Quantity x Unit_Price_At_Sale

b) The relationships between these tables disclosed above would be

- Sales to Customers which would be many to one as many sales can be made by one customer

- Sales to Sale Items where the relationship is one to many where one sale has multiple items included in the sale.

- Products and suppliers would be a many to one relationship where many products can be sent out by one supplier.

- Inventory transactions would be a many to one relationship since multiple stock movements can be related to a singular product.

A solid blue vertical bar is positioned on the left side of a white background. The bar is uniform in color and width, extending from the top to the bottom of the frame.

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](#)
2. Follow each step to create your database [here](#)

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(*) AS TotalRows  
FROM world.city  
WHERE CountryCode = 'USA'
```

The screenshot shows the SQL Studio interface. The top pane displays the SQL query: `SELECT COUNT(*) AS TotalRows FROM world.city WHERE CountryCode = 'USA'`. The bottom pane shows the execution results in a table with columns: Time, Action, Message, and Duration / Fetch. The results indicate that the query was executed successfully, returning 1 row and 1 message.

| Time | Action | Message | Duration / Fetch |
|--------------|--|-------------------|-----------------------|
| 00:00:00.000 | SELECT COUNT(*) AS TotalRows FROM world.city WHERE CountryCode = 'USA' | 1 row(s) returned | 0.000 sec / 0.000 sec |

2. **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 Country, LifeExpectancy
from country
order by LifeExpectancy DESC
limit 1;
```

The screenshot displays a SQL query execution environment. The query is as follows:

```
13 • select name as Country, LifeExpectancy
14   from country
15   order by LifeExpectancy DESC
16   limit 1;
```

The results are shown in a table with the following data:

| Country | LifeExpectancy |
|---------|----------------|
| Andorra | 83.5 |

Below the results, an 'Action Output' log shows the execution steps and their durations:

| # | Time | Action | Message | Duration / Fetch |
|----|----------|--|--|-----------------------|
| 72 | 15:48:25 | select name as Country, LifeExpectancy from country order LifeExpectancy DESC select 1 | Error Code: 1064. You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use ne... | 0.000 sec |
| 73 | 15:49:17 | Use world | 0 row(s) affected | 0.000 sec |
| 74 | 15:49:17 | select * from country LIMIT 0, 1000 | 239 row(s) returned | 0.000 sec / 0.000 sec |
| 75 | 15:49:17 | select * from city LIMIT 0, 1000 | 1000 row(s) returned | 0.000 sec / 0.000 sec |
| 76 | 15:49:17 | select COUNT(*) AS TotalRows from world.city where CountryCode = 'USA' LIMIT 0, 1000 | 1 row(s) returned | 0.000 sec / 0.000 sec |
| 77 | 15:49:17 | select name as Country, LifeExpectancy from country order by LifeExpectancy DESC limit 1 | 1 row(s) returned | 0.000 sec / 0.000 sec |



3. **"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, CountryCode
from city
where name like '%New%'
```

The screenshot displays a data tool interface with a SQL query editor at the top. Below the editor, a 'Result Grid' shows a list of cities and their country codes. The 'Output' section at the bottom provides a detailed log of the query execution, including timestamps, actions, messages, and durations.

| name | CountryCode |
|-------------------------|-------------|
| Newcastle | AUS |
| Newcastle upon Tyne | GBR |
| Newport | GBR |
| Newcastle | ZAF |
| Kowloon and New Kowloon | HKG |
| New Bombay | IND |
| New Delhi | IND |
| Khanewal | PAK |
| New York | USA |
| New Orleans | USA |
| Newark | USA |

| # | Time | Action | Message | Duration / Fetch |
|----|----------|--|----------------------|-----------------------|
| 78 | 15:54:17 | Use world | 0 row(s) affected | 0.000 sec |
| 79 | 15:54:17 | select * from country LIMIT 0, 1000 | 239 row(s) returned | 0.000 sec / 0.000 sec |
| 80 | 15:54:17 | select * from city LIMIT 0, 1000 | 1000 row(s) returned | 0.000 sec / 0.000 sec |
| 81 | 15:54:17 | select COUNTRY() AS TotalRows from world city where CountryCode = 'USA' LIMIT 0, 1000 | 1 row(s) returned | 0.000 sec / 0.000 sec |
| 82 | 15:54:17 | select name as Country, LifeExpectancy from country order by LifeExpectancy DESC limit 1 | 1 row(s) returned | 0.000 sec / 0.000 sec |
| 83 | 15:54:17 | select name, CountryCode from city wherename like "%New%" LIMIT 0, 1000 | 14 row(s) returned | 0.000 sec / 0.000 sec |

4. **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, CountryCode, Population
from city
order by Population DESC
limit 10;
```

| Name | CountryCode | Population |
|------------------|-------------|------------|
| Mumbai (Bombay) | IND | 10500000 |
| Seoul | KOR | 9981619 |
| São Paulo | BRA | 9661465 |
| Shanghai | CHN | 966300 |
| Jakarta | IDN | 9604900 |
| Karachi | PAK | 9263265 |
| Istanbul | TUR | 8787958 |
| Ciudad de México | MEX | 8591309 |
| Moscow | RUS | 8389200 |
| New York | USA | 8008278 |

| # | Time | Action | Message | Duration / Fetch |
|----|----------|--|----------------------|-----------------------|
| 92 | 16:03:49 | select * from country LIMIT 0, 1000 | 239 row(s) returned | 0.000 sec / 0.000 sec |
| 93 | 16:03:49 | select * from city LIMIT 0, 1000 | 1000 row(s) returned | 0.000 sec / 0.000 sec |
| 94 | 16:03:49 | select COUNT(*) AS TotalRows from world.city where CountryCode = 'USA' LIMIT 0, 1000 | 1 row(s) returned | 0.000 sec / 0.000 sec |
| 95 | 16:03:49 | select name as Country, LifeExpectancy from country order by LifeExpectancy DESC limit 1 | 1 row(s) returned | 0.000 sec / 0.000 sec |
| 96 | 16:03:49 | select name, CountryCode from city where name like 'UNew%' LIMIT 0, 1000 | 14 row(s) returned | 0.000 sec / 0.000 sec |
| 97 | 16:03:49 | select Name, CountryCode, Population from city order by Population DESC limit 10 | 10 row(s) returned | 0.000 sec / 0.000 sec |

5. **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, CountryCode, Population
from city
where Population > 2000000
order by Population desc;
```

```
22 • select Name, CountryCode, Population
23   from city
24   order by Population DESC
25   limit 10;
```

| Name | CountryCode | Population |
|------------------|-------------|------------|
| Mumbai (Bombay) | IND | 10500000 |
| Seoul | KOR | 9981619 |
| São Paulo | BRA | 9968485 |
| Shanghai | CHN | 9696300 |
| Jakarta | IDN | 9604900 |
| Karachi | PAK | 9269265 |
| Istanbul | TUR | 8767958 |
| Ciudad de México | MEX | 8591309 |
| Moscow | RUS | 8389200 |
| New York | USA | 8008278 |

| # | Time | Action | Message | Duration / Fetch |
|----|----------|--|----------------------|-----------------------|
| 92 | 16:03:49 | select * from country LIMIT 0, 1000 | 239 row(s) returned | 0.000 sec / 0.000 sec |
| 93 | 16:03:49 | select * from city LIMIT 0, 1000 | 1000 row(s) returned | 0.000 sec / 0.000 sec |
| 94 | 16:03:49 | select COUNT(*) AS TotalRows from world.city where CountryCode = 'USA' LIMIT 0, 1000 | 1 row(s) returned | 0.000 sec / 0.000 sec |
| 95 | 16:03:49 | select name as Country, LifeExpectancy from country order by LifeExpectancy DESC limit 1 | 1 row(s) returned | 0.000 sec / 0.000 sec |
| 96 | 16:03:49 | select name, CountryCode from city wherename like "%New%" LIMIT 0, 1000 | 14 row(s) returned | 0.000 sec / 0.000 sec |
| 97 | 16:03:49 | select Name, CountryCode, Population from city order by Population DESC limit 10 | 10 row(s) returned | 0.000 sec / 0.000 sec |



| Result Grid | | | | Filter Rows: | | Export: | Wrap Cell Content: |
|-------------|--------------------|-------------|------------|--------------|---------|---------|--------------------|
| | name | CountryCode | Population | | | | |
| ▶ | Mumbai (Bombay) | IND | 10500000 | | | | |
| | Seoul | KOR | 9981619 | | | | |
| | São Paulo | BRA | 9968485 | | | | |
| | Shanghai | CHN | 9696300 | | | | |
| | Jakarta | IDN | 9604900 | | | | |
| | Karachi | PAK | 9269265 | | | | |
| | Istanbul | TUR | 8787958 | | | | |
| | Ciudad de México | MEX | 8591309 | | | | |
| | Moscow | RUS | 8389200 | | | | |
| | New York | USA | 8008278 | | | | |
| | Tokyo | JPN | 7980230 | | | | |
| | Peking | CHN | 7472000 | | | | |
| | London | GBR | 7285000 | | | | |
| | Delhi | IND | 7206704 | | | | |
| | Cairo | EGY | 6789479 | | | | |
| | Teheran | IRN | 6758845 | | | | |
| | Lima | PER | 6464693 | | | | |
| | Chongqing | CHN | 6351600 | | | | |
| | Bangkok | THA | 6320174 | | | | |
| | Santafé de Bogotá | COL | 6260862 | | | | |
| | Rio de Janeiro | BRA | 5598953 | | | | |
| | Tianjin | CHN | 5286800 | | | | |
| | Kinshasa | COD | 5064000 | | | | |
| | Lahore | PAK | 5063499 | | | | |
| | Santiago de Chile | CHL | 4703954 | | | | |
| | St Petersburg | RUS | 4694000 | | | | |
| | Calcutta [Kolkata] | IND | 4399819 | | | | |
| | Wuhan | CHN | 4344600 | | | | |
| country 30 | city 31 | Result 32 | country 33 | city 34 | city 35 | city 36 | × |

6. **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, CountryCode
from city
where name like 'Be%';
```

```
32 • select name, CountryCode
33 from city
34 where name like 'Be%';
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: [

| name | CountryCode |
|-----------------|-------------|
| Béjaïa | DZA |
| Béchar | DZA |
| Benguela | AGO |
| Berazategui | ARG |
| Belize City | BLZ |
| Belmopan | BLZ |
| Belo Horizonte | BRA |
| Belém | BRA |
| Belford Roxo | BRA |
| Betim | BRA |
| Bento Gonçalves | BRA |
| Ralfort | GBD |



7. **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 city
where Population between 500000 and 1000000
order by Population desc;
```

```
36 • select name, CountryCode, Population
37   from city
38   where Population between 500000 and 1000000
39   order by Population desc;
```

| name | CountryCode | Population |
|-----------|-------------|------------|
| Amman | JOR | 1000000 |
| Mogadishu | SOM | 997000 |
| Volgograd | RUS | 993400 |
| Sendai | JPN | 989975 |
| Peshawar | PAK | 988005 |
| Baotou | CHN | 980000 |
| Adelaide | AUS | 978100 |
| Madurai | IND | 977856 |
| Mekka | SAU | 965700 |
| Köln | DEU | 962507 |
| Moscow | RUS | 950000 |

8. **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.

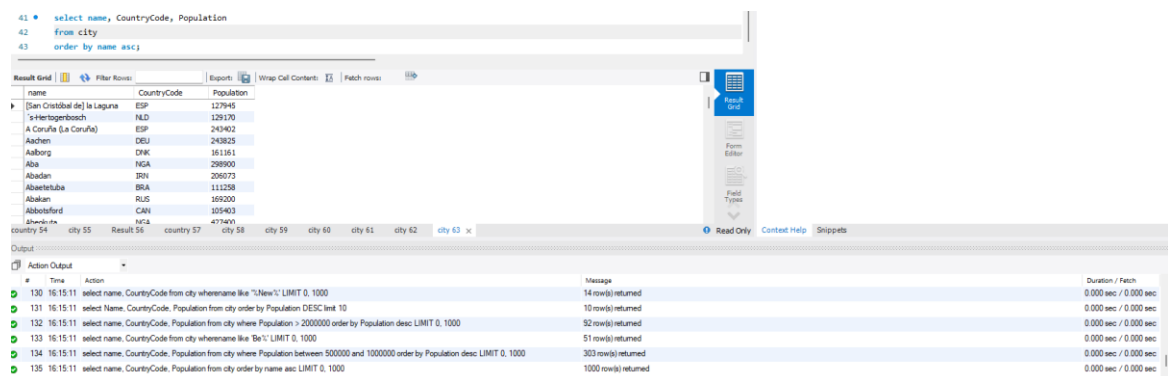
```
select name, CountryCode, Population
from city
order by name asc;
```

The screenshot displays a database query interface. At the top, the SQL query is entered: `select name, CountryCode, Population from city order by name asc;`. Below the query editor, a 'Result Grid' shows the first few rows of the query results. The columns are 'name', 'CountryCode', and 'Population'. The rows listed are: (San Cristobal de la Laguna, ESP, 127946), (San Hieronymus, NLD, 129370), (A Coruña (La Coruña), ESP, 245402), (Aachen, DEU, 243825), (Aalborg, DNK, 161161), (Alba, HUN, 206900), (Albadan, IRN, 206073), (Albaetfuba, BRA, 111258), (Abakun, RUS, 109200), (Abbotsford, CAN, 105403), and (Abundia, MEX, 47400). Below the result grid, an 'Output' section shows a log of actions and their results. The log includes the following entries:

| # | Time | Action | Message | Duration / Fetch |
|-----|----------|---|----------------------|-----------------------|
| 130 | 16:15:11 | select name, CountryCode from city where name like "%New%" LIMIT 0, 1000 | 14 row(s) returned | 0.000 sec / 0.000 sec |
| 131 | 16:15:11 | select Name, CountryCode, Population from city order by Population DESC limit 10 | 10 row(s) returned | 0.000 sec / 0.000 sec |
| 132 | 16:15:11 | select name, CountryCode, Population from city where Population > 2000000 order by Population desc LIMIT 0, 1000 | 92 row(s) returned | 0.000 sec / 0.000 sec |
| 133 | 16:15:11 | select name, CountryCode from city where name like "B%" LIMIT 0, 1000 | 51 row(s) returned | 0.000 sec / 0.000 sec |
| 134 | 16:15:11 | select name, CountryCode, Population from city where Population between 500000 and 1000000 order by Population desc LIMIT 0, 1000 | 303 row(s) returned | 0.000 sec / 0.000 sec |
| 135 | 16:15:11 | select name, CountryCode, Population from city order by name asc LIMIT 0, 1000 | 1000 row(s) returned | 0.000 sec / 0.000 sec |

9. **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, CountryCode, Population
from city
order by Population desc
limit 1;
```



The screenshot shows a database query interface. The SQL query is entered in the top text area: `select name, CountryCode, Population from city order by Population desc limit 1;`. Below the query, the 'Result Grid' displays a table with three columns: 'name', 'CountryCode', and 'Population'. The first row is highlighted, showing '(San Cristóbal de) La Laguna' with CountryCode 'ESP' and Population '127945'. To the right of the table is a 'Fetch Size' dropdown menu. Below the table, the 'Output' section shows a log of actions and their results, including the execution of the query and the number of rows returned.

| # | Time | Action | Message | Duration / Fetch |
|-----|----------|---|----------------------|-----------------------|
| 130 | 16:15:11 | select name, CountryCode from city where name like "%New%" LIMIT 0, 1000 | 14 row(s) returned | 0.000 sec / 0.000 sec |
| 131 | 16:15:11 | select Name, CountryCode, Population from city order by Population DESC limit 10 | 10 row(s) returned | 0.000 sec / 0.000 sec |
| 132 | 16:15:11 | select name, CountryCode, Population from city where Population > 2000000 order by Population desc LIMIT 0, 1000 | 92 row(s) returned | 0.000 sec / 0.000 sec |
| 133 | 16:15:11 | select name, CountryCode from city where name like "%a%"; LIMIT 0, 1000 | 51 row(s) returned | 0.000 sec / 0.000 sec |
| 134 | 16:15:11 | select name, CountryCode, Population from city where Population between 500000 and 1000000 order by Population desc LIMIT 0, 1000 | 303 row(s) returned | 0.000 sec / 0.000 sec |
| 135 | 16:15:11 | select name, CountryCode, Population from city order by name asc LIMIT 0, 1000 | 1000 row(s) returned | 0.000 sec / 0.000 sec |

- ```
SELECT Name
FROM city
GROUP BY Name
HAVING COUNT(*) = 1
ORDER BY Name ASC;
```



11. **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, CountryCode, Population
FROM city
ORDER BY Population ASC
LIMIT 1;
```

The screenshot displays the SQL Enterprise Manager interface. The central pane shows the execution of a query. The 'Result Grid' tab is active, displaying a single row of data:

| Name      | CountryCode | Population |
|-----------|-------------|------------|
| Adamstown | PCN         | 42         |

The 'Output' pane at the bottom shows the execution log with the following entries:

| #    | Time     | Action                                                                                                              | Message              | Duration / Fetch      |
|------|----------|---------------------------------------------------------------------------------------------------------------------|----------------------|-----------------------|
| 5368 | 01:19:23 | select name, CountryCode from city where name like 'A%'                                                             | 51 row(s) returned   | 0.000 sec / 0.000 sec |
| 5369 | 01:19:23 | select name, CountryCode, Population from city where Population between 500000 and 1000000 order by Population desc | 363 row(s) returned  | 0.000 sec / 0.000 sec |
| 5370 | 01:19:23 | select name, CountryCode, Population from city order by name asc                                                    | 1000 row(s) returned | 0.000 sec / 0.000 sec |
| 5371 | 01:19:23 | select name, CountryCode, Population from city order by Population desc                                             | 1 row(s) returned    | 0.000 sec / 0.000 sec |
| 5372 | 01:19:23 | SELECT Name FROM city GROUP BY Name HAVING COUNT(*) = 1 ORDER BY Name ASC                                           | 1000 row(s) returned | 0.000 sec / 0.000 sec |
| 5373 | 01:19:23 | SELECT Name, CountryCode, Population FROM city ORDER BY Population ASC                                              | 1 row(s) returned    | 0.000 sec / 0.000 sec |

12. **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 country
ORDER BY Population DESC
LIMIT 5;
```

The screenshot displays the SQL Server Enterprise Manager interface. The central pane shows the execution results of a query. The query text is as follows:

```
52 GROUP BY Name
53 HAVING COUNT(*) = 1
54 ORDER BY Name ASC;
55
56 SELECT Name, CountryCode, Population
57 FROM city
58 ORDER BY Population ASC
59 LIMIT 1;
60
61 SELECT Name, Population
62 FROM country
63 ORDER BY Population DESC
64 LIMIT 5;
```

The result grid shows the following data:

| Name          | Population |
|---------------|------------|
| China         | 1277518000 |
| India         | 1013662000 |
| United States | 278357000  |
| Indonesia     | 212107000  |
| Brazil        | 170115000  |

The bottom pane shows the output of the query execution, including the time taken and the number of rows returned for each step.

| Action | Time     | Message                                                                                                                           | Duration / Fetch   |                       |
|--------|----------|-----------------------------------------------------------------------------------------------------------------------------------|--------------------|-----------------------|
| 5383   | 01:22:04 | select name, CountryCode, Population from city where Population between 500000 and 1000000 order by Population desc LIMIT 0, 1000 | 303 rows returned  | 0.000 sec / 0.000 sec |
| 5384   | 01:22:04 | select name, CountryCode, Population from city order by name asc LIMIT 0, 1000                                                    | 1000 rows returned | 0.000 sec / 0.000 sec |
| 5385   | 01:22:04 | select name, CountryCode, Population from city order by Population desc limit 1                                                   | 1 row returned     | 0.000 sec / 0.000 sec |
| 5386   | 01:22:04 | SELECT Name FROM city GROUP BY Name HAVING COUNT(*) = 1 ORDER BY Name ASC LIMIT 0, 1000                                           | 1000 rows returned | 0.000 sec / 0.000 sec |
| 5387   | 01:22:04 | SELECT Name, CountryCode, Population FROM city ORDER BY Population ASC LIMIT 1                                                    | 1 row returned     | 0.000 sec / 0.000 sec |
| 5388   | 01:22:04 | SELECT Name, Population FROM country ORDER BY Population DESC LIMIT 5                                                             | 5 rows returned    | 0.000 sec / 0.000 sec |

13. **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 co.Name AS Country, ci.Name AS Capital
FROM country co
JOIN city ci ON co.Capital = ci.ID
WHERE co.Name = 'Spain';
```

The screenshot displays the SQL Server Enterprise Manager interface. The left pane shows the 'Navigation' tree with 'Server Status', 'Client Connections', 'Users and Privileges', 'Status and System Variables', 'Data Export', 'Data Import/Restore', 'INSTANCE', 'StartUp / Shutdown', 'Server Logs', 'Options File', 'PERFORMANCE', 'Dashboard', 'Performance Reports', and 'Performance Schema Setup'. The central pane shows the query editor with the following SQL code:

```
57 FROM city
58 ORDER BY Population ASC
59 LIMIT 1;
60
61 SELECT Name, Population
62 FROM country
63 ORDER BY Population DESC
64 LIMIT 1;
65
66 SELECT co.Name AS Country, ci.Name AS Capital
67 FROM country co
68 JOIN city ci ON co.Capital = ci.ID
69 WHERE co.Name = 'Spain';
70
```

The right pane shows the 'SQL Additions' tab with a message: 'Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.'

The bottom pane shows the 'Result Grid' with the following data:

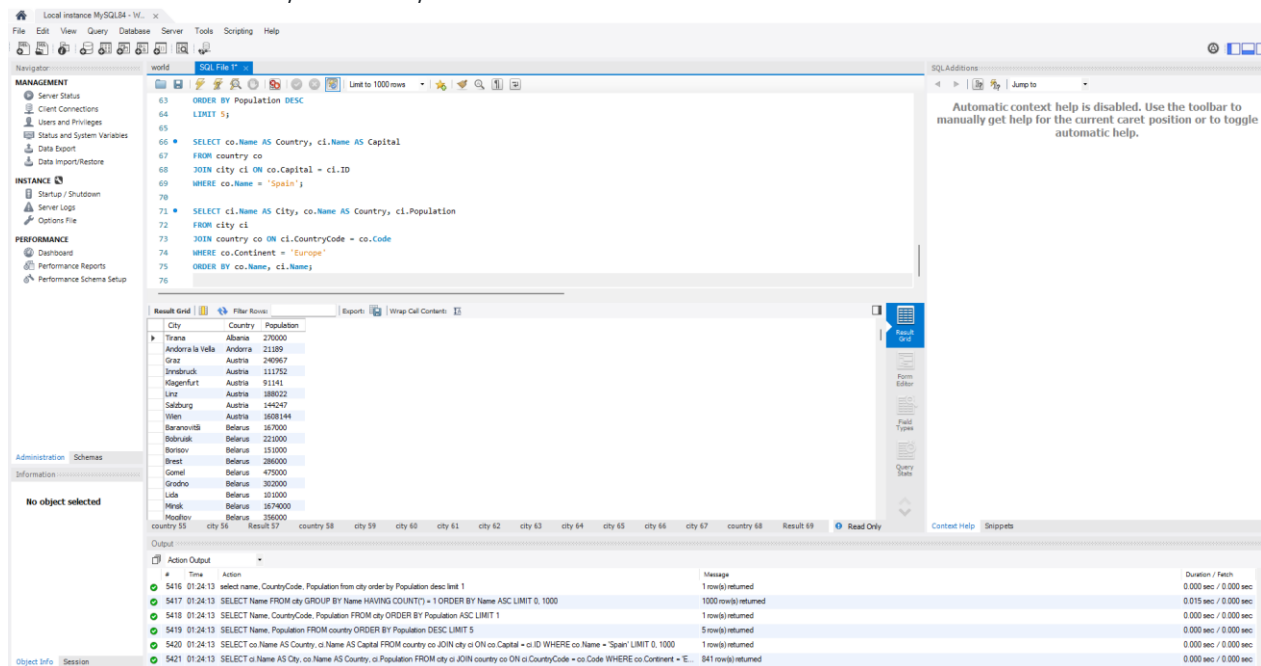
| Country | Capital |
|---------|---------|
| Spain   | Madrid  |

The bottom pane also shows the 'Action Output' tab with the following data:

| Time          | Action                                                                                                                                 | Message              | Duration / Fetch      |
|---------------|----------------------------------------------------------------------------------------------------------------------------------------|----------------------|-----------------------|
| 5399 01:23:23 | select name, CountryCode, Population from city order by name asc LIMIT 0, 1000                                                         | 1000 row(s) returned | 0.000 sec / 0.000 sec |
| 5400 01:23:23 | select name, CountryCode, Population from city order by Population desc limit 1                                                        | 1 row(s) returned    | 0.000 sec / 0.000 sec |
| 5401 01:23:23 | SELECT Name, CountryCode, Population FROM city GROUP BY Name HAVING COUNT(*) = 1 ORDER BY Name ASC LIMIT 0, 1000                       | 1000 row(s) returned | 0.000 sec / 0.000 sec |
| 5402 01:23:23 | SELECT Name, CountryCode, Population FROM city ORDER BY Population ASC LIMIT 1                                                         | 1 row(s) returned    | 0.000 sec / 0.000 sec |
| 5403 01:23:23 | SELECT Name, CountryCode, Population FROM country ORDER BY Population DESC LIMIT 5                                                     | 5 row(s) returned    | 0.000 sec / 0.000 sec |
| 5404 01:23:23 | SELECT co.Name AS Country, ci.Name AS Capital FROM country co JOIN city ci ON co.Capital = ci.ID WHERE co.Name = 'Spain' LIMIT 0, 1000 | 1 row(s) returned    | 0.000 sec / 0.000 sec |

14. **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 ci.Name AS City, co.Name AS Country, ci.Population
FROM city ci
JOIN country co ON ci.CountryCode = co.Code
WHERE co.Continent = 'Europe'
ORDER BY co.Name, ci.Name;
```



The screenshot shows the MySQL Workbench interface. The SQL Editor contains the following query:

```
SELECT ci.Name AS City, co.Name AS Country, ci.Population
FROM city ci
JOIN country co ON ci.CountryCode = co.Code
WHERE co.Continent = 'Europe'
ORDER BY co.Name, ci.Name;
```

The Results Grid displays the following data:

| City             | Country | Population |
|------------------|---------|------------|
| Tirana           | Albania | 270000     |
| Andorra la Vella | Andorra | 21189      |
| Graz             | Austria | 240967     |
| Innsbruck        | Austria | 111752     |
| Klagenfurt       | Austria | 91141      |
| Linz             | Austria | 188022     |
| Salzburg         | Austria | 142477     |
| Wien             | Austria | 1601146    |
| Baranovits       | Belarus | 167000     |
| Bobruisk         | Belarus | 221000     |
| Barnov           | Belarus | 151000     |
| Brest            | Belarus | 286000     |
| Gomel            | Belarus | 475000     |
| Grodno           | Belarus | 302000     |
| Lida             | Belarus | 101000     |
| Minsk            | Belarus | 1674000    |
| Mogilev          | Belarus | 356000     |

The Output tab shows the execution log with the following messages:

| #    | Time     | Action                                                                                                                                        | Message              | Duration / Fetch      |
|------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------|----------------------|-----------------------|
| 5416 | 01:24:13 | select name, CountryCode, Population from city order by Population desc limit 1                                                               | 1 row(s) returned    | 0.000 sec / 0.000 sec |
| 5417 | 01:24:13 | SELECT Name FROM city GROUP BY Name HAVING COUNT(*) = 1 ORDER BY Name ASC LIMIT 0, 1000                                                       | 1000 row(s) returned | 0.015 sec / 0.000 sec |
| 5418 | 01:24:13 | SELECT Name, CountryCode, Population FROM city ORDER BY Population ASC LIMIT 1                                                                | 1 row(s) returned    | 0.000 sec / 0.000 sec |
| 5419 | 01:24:13 | SELECT Name, Population FROM country ORDER BY Population DESC LIMIT 5                                                                         | 5 row(s) returned    | 0.000 sec / 0.000 sec |
| 5420 | 01:24:13 | SELECT co.Name AS Country, ci.Name AS Capital FROM country co JOIN city ci ON co.Capital = ci.ID WHERE co.Name = 'Spain' LIMIT 0, 1000        | 1 row(s) returned    | 0.000 sec / 0.000 sec |
| 5421 | 01:24:13 | SELECT ci.Name AS City, co.Name AS Country, ci.Population FROM city ci JOIN country co ON ci.CountryCode = co.Code WHERE co.Continent = 'E... | 541 row(s) returned  | 0.000 sec / 0.000 sec |



15. **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 co.Name AS Country, AVG(ci.Population) AS Average_City_Population
FROM city ci
JOIN country co ON ci.CountryCode = co.Code
GROUP BY co.Name
ORDER BY Average_City_Population DESC;
```

The screenshot displays a SQL IDE interface with a query editor, a result grid, and an output pane. The query editor contains the following SQL code:

```
69 WHERE co.Name = 'Spain';
70
71 SELECT ci.Name AS City, co.Name AS Country, ci.Population
72 FROM city ci
73 JOIN country co ON ci.CountryCode = co.Code
74 WHERE co.Continent = 'Europe'
75 ORDER BY co.Name, ci.Name;
76
77 SELECT co.Name AS Country, AVG(ci.Population) AS Average_City_Population
78 FROM city ci
79 JOIN country co ON ci.CountryCode = co.Code
80 GROUP BY co.Name
81 ORDER BY Average_City_Population DESC;
82
```

The result grid shows the following data:

| Country       | Average_City_Population |
|---------------|-------------------------|
| Singapore     | 4017733.0000            |
| Hong Kong     | 3693738.0000            |
| Uruguay       | 3268000.0000            |
| Guinea        | 3090633.0000            |
| Uganda        | 2958000.0000            |
| Liberia       | 2500000.0000            |
| Serra Leone   | 2500000.0000            |
| Mali          | 2000000.0000            |
| Australia     | 2001100.0000            |
| Mongolia      | 2000000.0000            |
| Congo         | 2000000.0000            |
| Libyan Arab.  | 674251.7500             |
| Lebanon       | 670000.0000             |
| Thailand      | 662763.4167             |
| Côte d'Ivoire | 638227.4000             |
| Azerbaijan    | 616000.0000             |
| Iran          | 595068.4000             |

The output pane shows the execution of the query, including the following messages:

- 5434 01:25:16 SELECT Name FROM city GROUP BY Name HAVING COUNT(\*) = 1 ORDER BY Name ASC LIMIT 0, 1000 1000 rows returned 0.000 sec / 0.000 sec
- 5435 01:25:16 SELECT Name, CountryCode, Population FROM city ORDER BY Population ASC LIMIT 1 1 row(s) returned 0.000 sec / 0.000 sec
- 5436 01:25:16 SELECT Name, Population FROM country ORDER BY Population DESC LIMIT 5 5 row(s) returned 0.000 sec / 0.000 sec
- 5437 01:25:16 SELECT co.Name AS Country, ci.Name AS Capital FROM country co JOIN city ci ON co.Capital = ci.ID WHERE co.Name = 'Spain' LIMIT 0, 1000 1 row(s) returned 0.000 sec / 0.000 sec
- 5438 01:25:16 SELECT co.Name AS City, co.Name AS Country, ci.Population FROM city ci JOIN country co ON ci.CountryCode = co.Code WHERE co.Continent = 'E' 811 row(s) returned 0.000 sec / 0.000 sec
- 5439 01:25:16 SELECT co.Name AS Country, AVG(ci.Population) AS Average\_City\_Population FROM city ci JOIN country co ON ci.CountryCode = co.Code GROUP BY co.Name 232 row(s) returned 0.016 sec / 0.000 sec

16. **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_City,
 ci.Population AS Capital_Population
FROM country co
JOIN city ci ON co.Capital = ci.ID
ORDER BY ci.Population DESC;
```

The screenshot displays a SQL IDE interface with a query editor, a results grid, and an output pane. The query editor contains the following SQL code:

```
SELECT
 co.Name AS Country,
 ci.Name AS Capital_City,
 ci.Population AS Capital_Population
FROM country co
JOIN city ci ON co.Capital = ci.ID
ORDER BY ci.Population DESC;
```

The results grid shows the following data:

| Country            | Capital_City      | Capital_Population |
|--------------------|-------------------|--------------------|
| South Korea        | Seoul             | 9801819            |
| Indonesia          | Jakarta           | 9604600            |
| Mexico             | Ciudad de México  | 891309             |
| Russian Federation | Moscow            | 839200             |
| Japan              | Tokyo             | 7980230            |
| China              | Peking            | 7472000            |
| United Kingdom     | London            | 720500             |
| Egypt              | Cairo             | 6769479            |
| Iran               | Tehran            | 670846             |
| Peru               | Lima              | 6464613            |
| Thailand           | Bangkok           | 6321174            |
| Colombia           | Santafé de Bogotá | 620862             |
| Congo, The Demo... | Kinshasa          | 500400             |
| Chile              | Santiago de Chile | 4703954            |
| Iraq               | Baghdad           | 420600             |
| Singapore          | Singapore         | 4617733            |
| Bangladesh         | Dhaka             | 4612850            |

The output pane shows the execution of the query, with the following messages:

```
5453 01:28:03 SELECT Name, CountryCode, Population FROM city ORDER BY Population ASC LIMIT 1
5454 01:28:03 SELECT Name, Population FROM country ORDER BY Population DESC LIMIT 5
5455 01:28:03 SELECT co.Name AS Country, ci.Name AS Capital FROM country co JOIN city ci ON co.Capital = ci.ID WHERE co.Name = 'Spain' LIMIT 0, 1000
5456 01:28:03 SELECT co.Name AS Country, ci.Population FROM city ci JOIN country co ON ci.CountryCode = co.Code WHERE co.Continent = 'E...'
5457 01:28:03 SELECT co.Name AS Country, AVG(City.Population) AS Average_City_Population FROM city ci JOIN country co ON ci.CountryCode = co.Code GROUP BY co.Name
5458 01:28:03 SELECT co.Name AS Country, ci.Name AS Capital_City, ci.Population AS Capital_Population FROM country co JOIN city ci ON co.Capital = ci.ID ORDER BY ci.Population DESC;
```

17. **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,
 Population,
 SurfaceArea,
 ROUND(Population / SurfaceArea, 2) AS Population_Density
FROM country
WHERE SurfaceArea > 0 -- Avoid division by zero
ORDER BY Population_Density ASC;
```

The screenshot displays the SQL Server Enterprise Manager interface. The left pane shows the 'Server Enterprise' tree with 'Databases' expanded. The 'world' database is selected. The 'Query Editor' window shows the following SQL query:

```
SELECT
 Name AS Country,
 Population,
 SurfaceArea,
 ROUND(Population / SurfaceArea, 2) AS Population_Density
FROM country
WHERE SurfaceArea > 0 -- Avoid division by zero
ORDER BY Population_Density ASC;
```

The 'Results' pane shows the following data:

| Country                                      | Population | SurfaceArea | Population_Density |
|----------------------------------------------|------------|-------------|--------------------|
| Antarctica                                   | 0          | 13120000.00 | 0.00               |
| French Southern territories                  | 0          | 7760.00     | 0.00               |
| Bouvet Island                                | 0          | 50.00       | 0.00               |
| Heard Island and McDonald Islands            | 0          | 350.00      | 0.00               |
| British Indian Ocean Territory               | 0          | 78.00       | 0.00               |
| South Georgia and the South Sandwich Islands | 0          | 3903.00     | 0.00               |
| United States Minor Outlying Islands         | 0          | 16.00       | 0.00               |
| Greenland                                    | 56000      | 2166090.00  | 0.03               |
| Swabland and Jan Mayen                       | 3200       | 42402.00    | 0.05               |
| Falkland Islands                             | 2000       | 12173.00    | 0.16               |
| Pitcairn                                     | 50         | 49.00       | 1.02               |
| Western Sahara                               | 293000     | 266000.00   | 1.10               |
| Mongolia                                     | 2462000    | 1564000.00  | 1.70               |
| French Guiana                                | 181000     | 80000.00    | 2.01               |
| Namibia                                      | 1726000    | 824292.00   | 2.09               |
| Australia                                    | 18886000   | 7741220.00  | 2.44               |
| Suriname                                     | 417000     | 163265.00   | 2.55               |

The 'Output' pane shows the execution plan for the query, indicating that the query was executed successfully and returned 239 rows.

18. **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
 Name AS Country,
 ROUND(GNP / Population, 2) AS GDP_Per_Capita
FROM country
WHERE Population > 0 AND GNP IS NOT NULL
ORDER BY GDP_Per_Capita DESC
LIMIT 10;
```

The screenshot displays the SQL Enterprise Manager interface. The central pane shows the execution of a SQL query. The result grid displays the following data:

| Country        | GDP_Per_Capita |
|----------------|----------------|
| Bermuda        | 0.04           |
| Luxembourg     | 0.04           |
| Switzerland    | 0.04           |
| Brunei         | 0.04           |
| Austria        | 0.03           |
| Liechtenstein  | 0.03           |
| Germany        | 0.03           |
| Denmark        | 0.03           |
| Japan          | 0.03           |
| Cayman Islands | 0.03           |

The bottom pane shows the execution log with the following entries:

| #    | Time     | Action | Message                                                                                                                                 | Duration / Fetch      |
|------|----------|--------|-----------------------------------------------------------------------------------------------------------------------------------------|-----------------------|
| 5484 | 01:30:05 | SELECT | co Name AS Country, co Name AS Capital FROM country co JOIN city co ON co Capital = co ID WHERE co Name = 'Spain' LIMIT 0, 1000         | 1 row(s) returned     |
| 5485 | 01:30:05 | SELECT | co Name AS City, co Name AS Country, co Population FROM city co JOIN country co ON co CountryCode = co Code WHERE co Continent = 'E'... | 841 row(s) returned   |
| 5486 | 01:30:05 | SELECT | co Name AS Country, AVG(co Population) AS Average_Population FROM city co JOIN country co ON co CountryCode = co Code GROUP...          | 232 row(s) returned   |
| 5487 | 01:30:05 | SELECT | co Name AS Country, co Name AS Capital, co Population AS Capital_Population FROM country co JOIN city co ON co Capital = co ID...       | 239 row(s) returned   |
| 5488 | 01:30:05 | SELECT | Name AS Country, Population, SurfaceArea, ROUND(Population / SurfaceArea, 2) AS Population_Density FROM country WHERE...                | 0.000 sec / 0.000 sec |
| 5489 | 01:30:05 | SELECT | Name AS Country, ROUND(GNP / Population, 2) AS GDP_Per_Capita FROM country WHERE Population > 0 AND GNP IS NOT NULL...                  | 10 row(s) returned    |

19. **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 Name, CountryCode, Population

FROM city

ORDER BY Population DESC

LIMIT 30, 10;

The screenshot displays the SQL Server Enterprise Manager interface. The central pane shows a query window with the following SQL code:

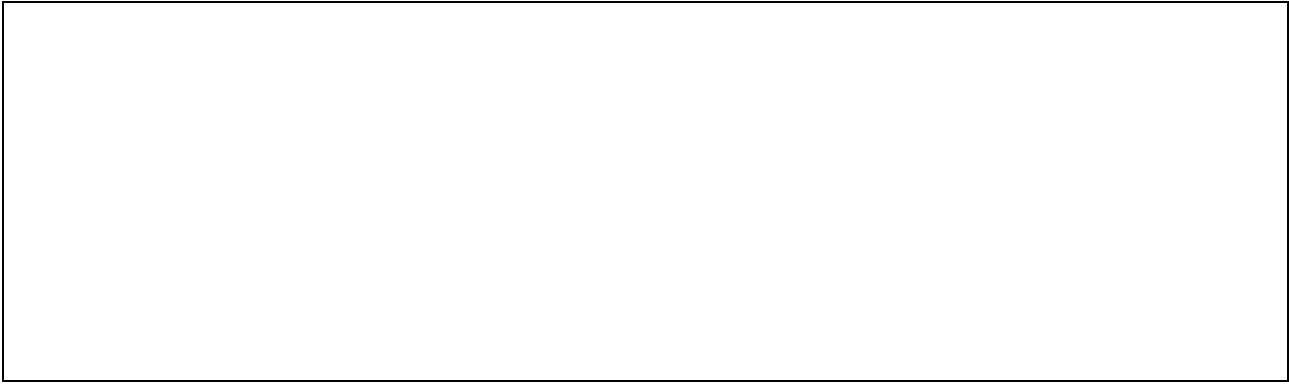
```
99
100 * SELECT
101 Name AS Country,
102 ROUND(GDP / Population, 2) AS GDP_Per_Capita
103 FROM country
104 WHERE Population > 0 AND GDP IS NOT NULL
105 ORDER BY GDP_Per_Capita DESC
106 LIMIT 10;
107
108 * SELECT Name, CountryCode, Population
109 FROM city
110 ORDER BY Population DESC
111 LIMIT 30, 10;
112
```

The Results pane below the query shows a table with three columns: Name, CountryCode, and Population. The data is as follows:

| Name               | CountryCode | Population |
|--------------------|-------------|------------|
| Shenyang           | CHN         | 4265200    |
| Kanton [Guangzhou] | CHN         | 4256300    |
| Singapore          | SGP         | 4017733    |
| Ho Chi Minh City   | VNM         | 3980000    |
| Chennai (Madras)   | IND         | 3841296    |
| Pusan              | KOR         | 3694622    |
| Los Angeles        | USA         | 3694620    |
| Dhaka              | BGD         | 3612850    |
| Berlin             | DEU         | 3386667    |
| Rangoon (Yangon)   | MMR         | 3361700    |

The bottom pane shows the execution plan for the query, indicating that the query returned 10 rows in 0.000 seconds.





## 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:



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.**

