USING SQL TO QUERY 'WORLD DATA BASE'

Charlie Thomas

05/05/2024

SQL Assignment

Special Thanks to Mr. Yusuf Satilmis

Table of Content

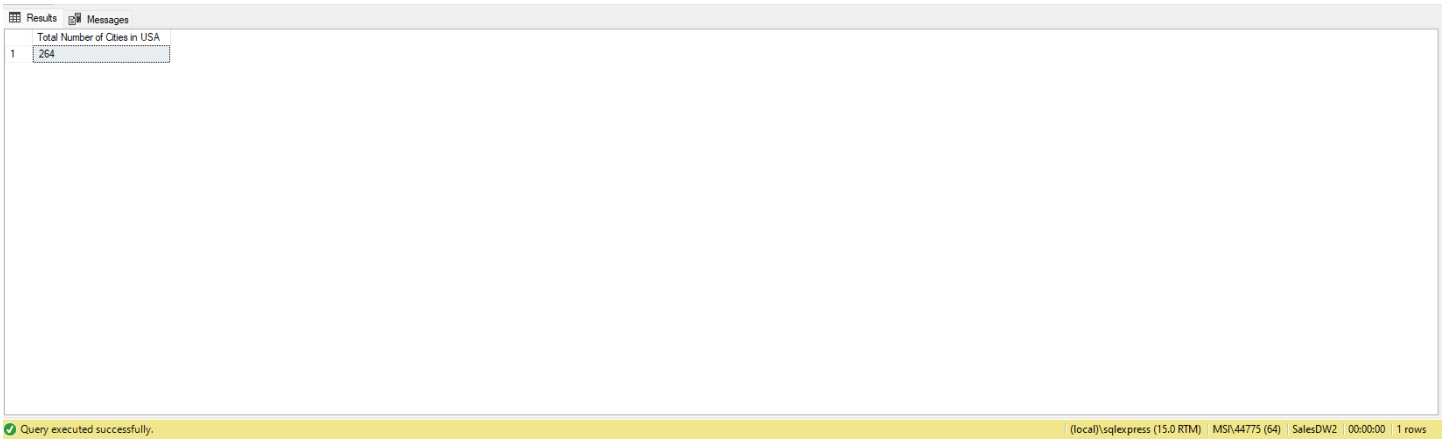
PART A: ASSIGNMENT QUESTIONS

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.

Query:

```
SELECT
	COUNT( DISTINCT name) as [Total number of Cities in USA]
FROM
	dbo.city
WHERE
	CountryCode = 'USA';
```

Result Snippet:



Explanation:

This SQL query is used to count the total number of distinct city names from the **dbo.city** table where the **CountryCode** is 'USA'. Let's break down the query and explain each part:

1. **SELECT COUNT(DISTINCT name) as [Total number of City]**: This part of the query selects the count of distinct city names from the **name** column of the **dbo.city** table. The **COUNT()** function is used to count the number of rows, and **DISTINCT** ensures that each city name is counted only once. The alias **[Total number of City]** is given to the result column for better readability.

2. **FROM dbo.city**: This specifies the table from which the data is being retrieved, in this case, the **dbo.city** table.

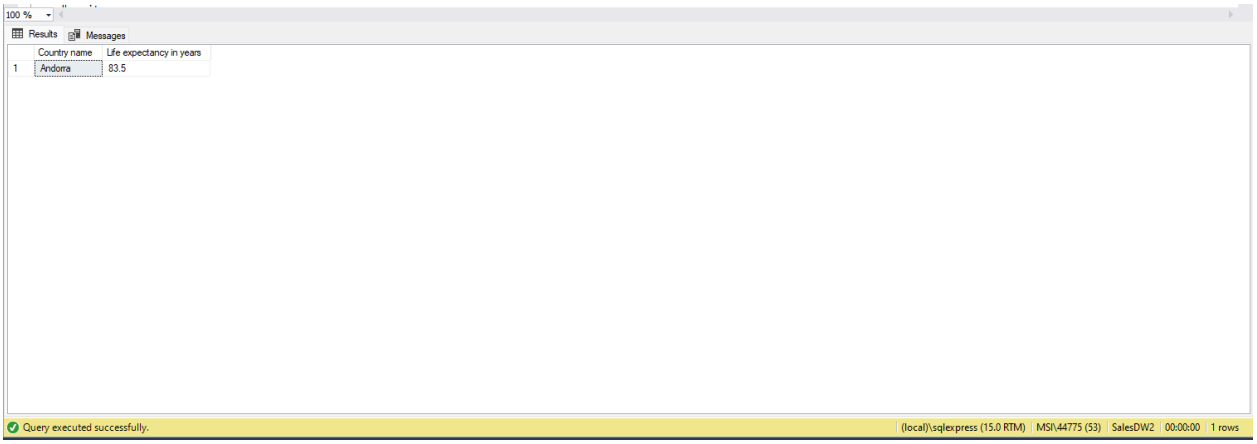
3. **WHERE CountryCode = 'USA'**: This part of the query filters the rows to include only those where the **CountryCode** column equals 'USA'. This restricts the count to only the cities belonging to the United States.

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 prioritizing healthcare resources and interventions.

Query:

```
SELECT
	TOP 1 name AS [Country name], Lifeexpectancy AS [Life expectancy in years]
FROM
	dbo.country
ORDER BY
	lifeexpectancy DESC;
```

Result Snippet:



Explanation:

This SQL query retrieves the name and life expectancy of the country with the highest life expectancy from the **dbo.country** table. Let's break down the query and explain each part:

1. **SELECT TOP 1 name AS [Country name], Lifeexpectancy AS [Life expectancy in years]**: This part of the query selects the top 1 record from the result set. It retrieves the **name** column from the **dbo.country** table, aliased as **[Country name]**, and the **Lifeexpectancy** column, aliased as **[Life expectancy in years]**. This selects the name and life expectancy of the country with the highest life expectancy.

2. **FROM dbo.country**: This specifies the table from which the data is being retrieved, in this case, the **dbo.country** table.

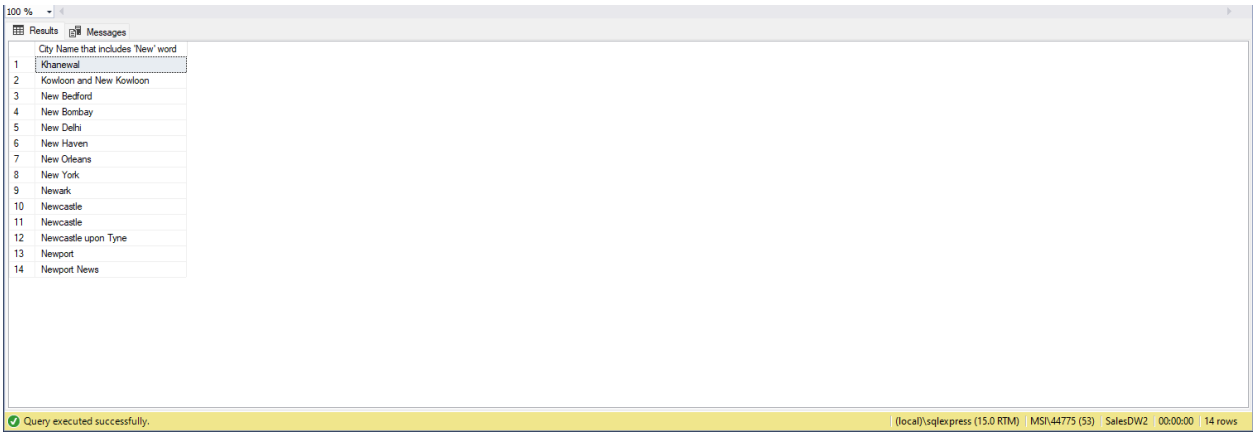
3. **ORDER BY lifeexpectancy DESC**: This part of the query orders the result set by the **lifeexpectancy** column in descending order. This means that the country with the highest life expectancy will be at the top of the result set.

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.

Query:

```
SELECT
	name AS [City Name that includes 'New' word]
FROM
	dbo.city
WHERE
	name like '%new%'
ORDER BY
	name ASC;
```

Result Snippet:



	City Name that includes 'New' word
1	Khanewal
2	Kowloon and New Kowloon
3	New Bedford
4	New Bombay
5	New Delhi
6	New Haven
7	New Orleans
8	New York
9	Newark
10	Newcastle
11	Newcastle
12	Newcastle upon Tyne
13	Newport
14	Newport News

Explanation:

This SQL query retrieves the names of cities from the **dbo.city** table where the city name includes the word 'New'.

1. **SELECT name AS [City Name that includes 'New' word]**: This part of the query selects the **name** column from the **dbo.city** table. The alias **[City Name that includes 'New' word]** is given to the result column for better readability.

2. **FROM dbo.city**: This specifies the table from which the data is being retrieved, in this case, the **dbo.city** table.

3. **WHERE name like '%new%'**: This part of the query filters the rows to include only those where the **name** column contains the substring 'new' (case insensitive) anywhere in the city name. The **%** symbols are wildcards, which means that 'new' can occur at the beginning, middle, or end of the city name.

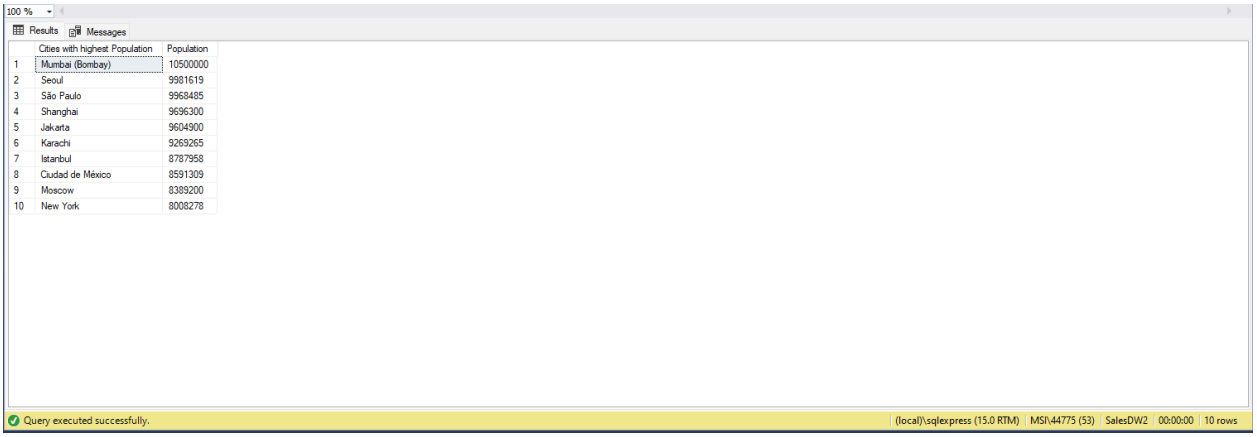
4. **ORDER BY name ASC**: This part of the query orders the result set by the **name** column in ascending order. This ensures that the city names are displayed alphabetically.

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.

Query:

```
SELECT
	TOP 10 name AS [Cities with highest Population],Population
FROM
	dbo.city
ORDER BY
	population DESC;
```

Result Snippet:



	Cities with highest Population	Population
1	Mumbai (Bombay)	10500000
2	Seoul	9981619
3	São Paulo	9968485
4	Shanghai	9696300
5	Jakarta	9604900
6	Karachi	9269265
7	Istanbul	8787958
8	Ciudad de México	8591309
9	Moscow	8389200
10	New York	8008278

Explanation:

This SQL query retrieves the names and populations of the top 10 cities with the highest populations from the **dbo.city** table.

1. **SELECT TOP 10 name AS [Cities with highest Population], Population**: This part of the query selects the top 10 records from the result set. It retrieves the **name** column from the **dbo.city** table, aliased as **[Cities with highest Population]**, and the **Population** column. This selects the names and populations of the top 10 cities with the highest populations.

2. **FROM dbo.city**: This specifies the table from which the data is being retrieved, in this case, the **dbo.city** table.

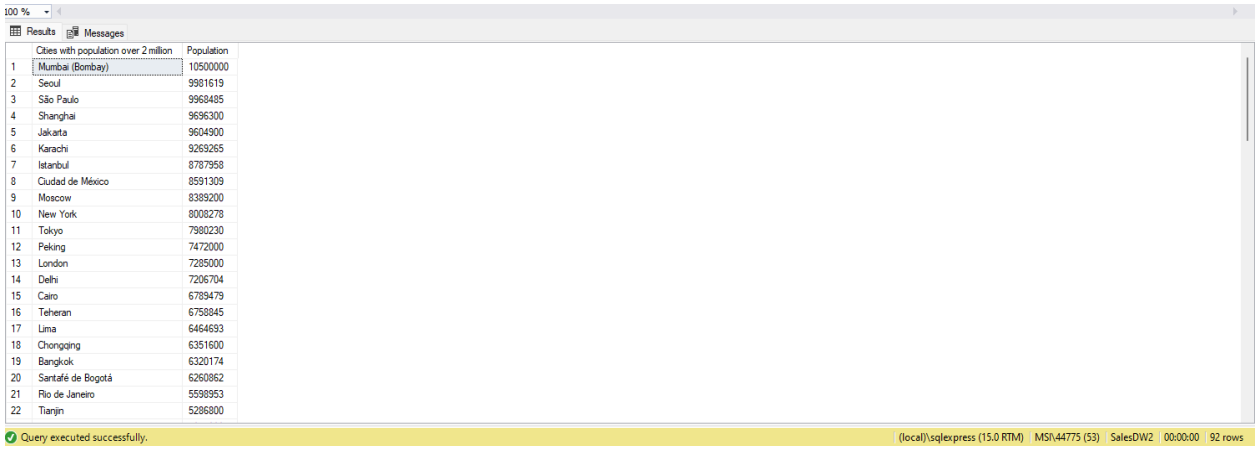
3. **ORDER BY population DESC**: This part of the query orders the result set by the **Population** column in descending order. This means that the cities with the highest populations will appear at the top of the result set.

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.

Query:

```
SELECT
	name AS [Cities with population over 2 million],Population
FROM
	dbo.city
WHERE
	population > 2000000
ORDER BY
	population DESC;
```

Code Snippet:



Results | Messages

	Cities with population over 2 million	Population
1	Mumbai (Bombay)	10500000
2	Seoul	9981619
3	São Paulo	9968485
4	Shanghai	9696300
5	Jakarta	9604900
6	Karachi	9269265
7	Istanbul	8787958
8	Ciudad de México	8591309
9	Moscow	8389200
10	New York	8008278
11	Tokyo	7980230
12	Peking	7472000
13	London	7285000
14	Delhi	7206704
15	Cairo	6789479
16	Teheran	6758845
17	Lima	6464693
18	Chongqing	6351600
19	Bangkok	6320174
20	Santafé de Bogotá	6260862
21	Rio de Janeiro	5598953
22	Tianjin	5286800

Query executed successfully. (local)\sqlexpress (15.0 RTM) | MSI\44775 (53) | SalesDW2 | 00:00:00 | 92 rows

Explanation:

This SQL query retrieves the names and populations of cities from the **dbo.city** table where the population is over 2 million.

1. **SELECT name AS [Cities with population over 2 million], Population**: This part of the query selects the **name** column from the **dbo.city** table, aliased as **[Cities with population over 2 million]**, and the **Population** column. This selects the names and populations of cities where the population is over 2 million.

2. **FROM dbo.city**: This specifies the table from which the data is being retrieved, in this case, the **dbo.city** table.

3. **WHERE population > 2000000**: This part of the query filters the rows to include only those where the **Population** column is greater than 2 million. This restricts the results to cities with populations over 2 million.

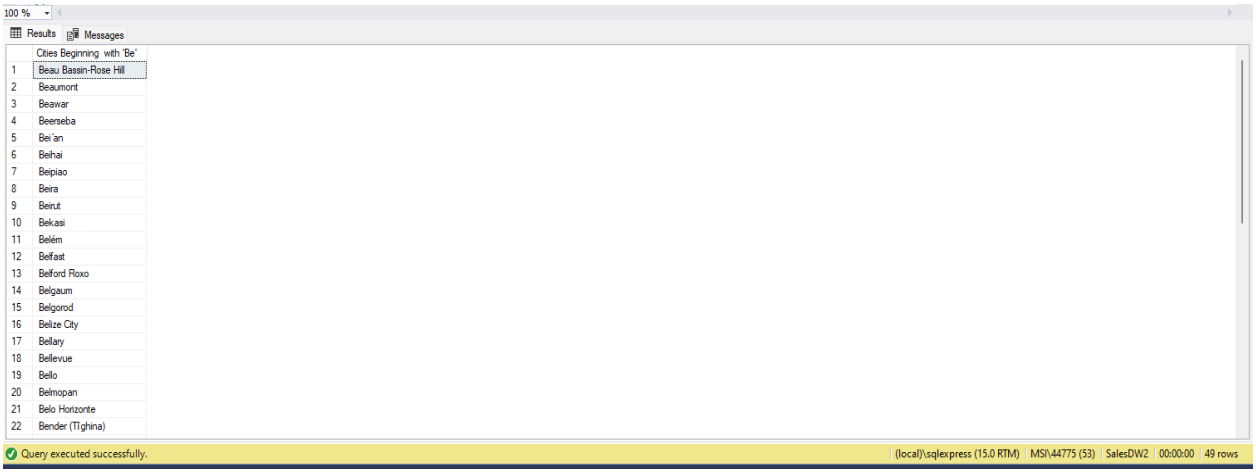
4. **ORDER BY population DESC**: This part of the query orders the result set by the **Population** column in descending order. This means that cities with the highest populations will appear at the top of the result set.

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.

Query:

```
SELECT
	Distinct name AS [Cities Beginning  with 'Be']
FROM
	dbo.city
WHERE
	name LIKE 'Be%'
ORDER BY
	name;
```

Result Snippet:



	Cities Beginning with 'Be'
1	Beau Bassin-Rose Hill
2	Beaumont
3	Beawar
4	Beerseba
5	Bei'an
6	Beihai
7	Beipiao
8	Beira
9	Beirut
10	Bekasi
11	Belém
12	Belfast
13	Belford Roxo
14	Belgaum
15	Belgorod
16	Belize City
17	Bellary
18	Bellevue
19	Bello
20	Belmopan
21	Belo Horizonte
22	Bender (Tighina)

Explanation:

This SQL query retrieves the distinct names of cities from the **dbo.city** table where the city name begins with 'Be'.

1. **SELECT DISTINCT name AS [Cities Beginning with 'Be']**: This part of the query selects the distinct **name** column from the **dbo.city** table, aliased as **[Cities Beginning with 'Be']**. The **DISTINCT** keyword ensures that only unique city names are returned.

2. **FROM dbo.city**: This specifies the table from which the data is being retrieved, in this case, the **dbo.city** table.

3. **WHERE name LIKE 'Be%'**: This part of the query filters the rows to include only those where the **name** column starts with **'Be'**. The **LIKE** operator with the pattern **'Be%'** matches city names that start with 'Be' followed by any number of characters.

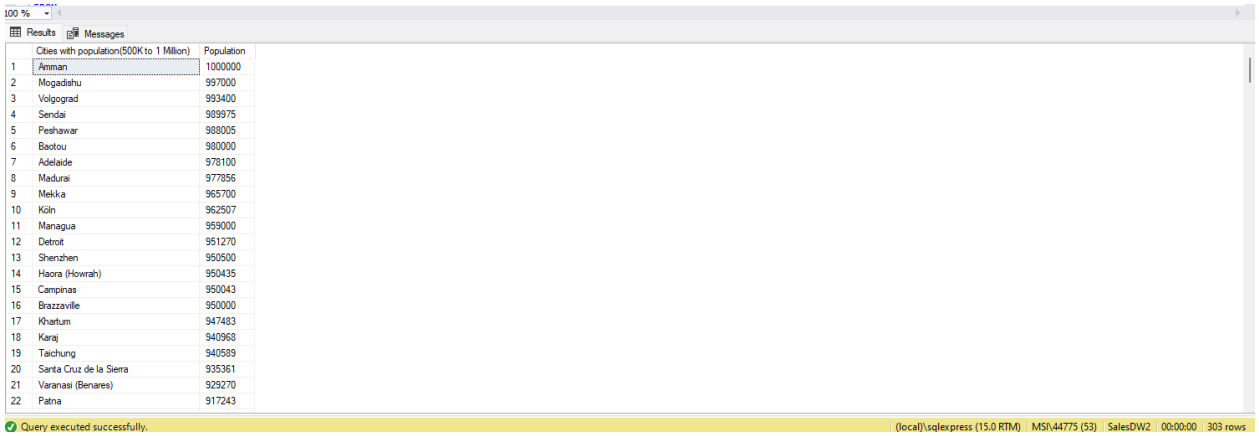
4. **ORDER BY name**: This part of the query orders the result set by the **name** column in ascending order. This ensures that the city names are displayed alphabetically.

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.

Query:

```
SELECT
	Distinct name AS [Cities with population(500K to 1 Million)],Population
FROM
	dbo.city
WHERE
	population BETWEEN 500000 AND 1000000
ORDER BY
	population DESC;
```

Result Snippet:



	Cities with population(500K to 1 Million)	Population
1	Amman	1000000
2	Mogadishu	997000
3	Volgograd	993400
4	Sendai	989975
5	Peshawar	988005
6	Baotou	980000
7	Adelaide	978100
8	Madurai	977856
9	Mekka	965700
10	Köln	962507
11	Managua	959000
12	Detroit	951270
13	Shenzhen	950500
14	Haora (Howrah)	950435
15	Campinas	950043
16	Brazzaville	950000
17	Khartum	947483
18	Karaj	940968
19	Taichung	940589
20	Santa Cruz de la Sierra	935361
21	Varanasi (Benares)	929270
22	Patna	917243

Explanation:

This SQL query retrieves the distinct names and populations of cities from the **dbo.city** table where the population falls within the range of 500,000 to 1 million. Let's break down the query and explain each part:

1. **SELECT DISTINCT name AS [Cities with population(500K to 1 Million)], Population**: This part of the query selects the distinct **name** column from the **dbo.city** table, aliased as **[Cities with population(500K to 1 Million)]**, and the **Population** column. The **DISTINCT** keyword ensures that only unique combinations of city names and populations are returned.

2. **FROM dbo.city**: This specifies the table from which the data is being retrieved, in this case, the **dbo.city** table.

3. **WHERE population BETWEEN 500000 AND 1000000**: This part of the query filters the rows to include only those where the **Population** column falls within the range of 500,000 to 1 million. The **BETWEEN** operator is used to specify the range inclusively.

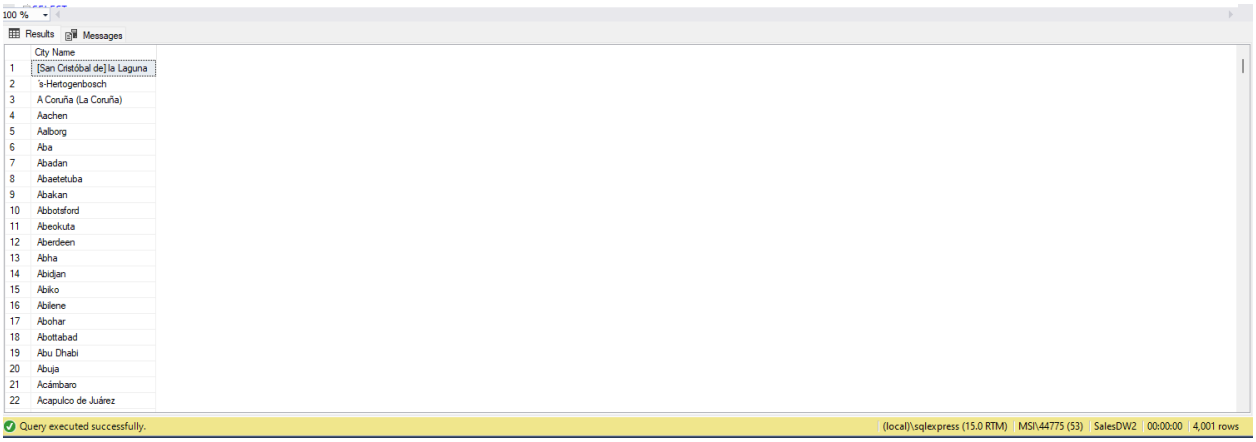
4. **ORDER BY population DESC**: This part of the query orders the result set by the **Population** column in descending order. This ensures that the cities with the highest populations within the specified range appear at the top of the result set.

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.

Query:

```
SELECT
	Distinct name AS [City Name]
FROM
	dbo.city
ORDER BY
	name ASC;
```

Result Snippet:



	City Name
1	[San Cristóbal de] la Laguna
2	's-Hertogenbosch
3	A Coruña (La Coruña)
4	Aachen
5	Aalborg
6	Aba
7	Abadan
8	Abaetetuba
9	Abakan
10	Abbotsford
11	Abeokuta
12	Aberdeen
13	Abha
14	Abidjan
15	Abiko
16	Abilene
17	Abohar
18	Abottabad
19	Abu Dhabi
20	Abuja
21	Acámbaro
22	Acapulco de Juárez

Explanation:

This SQL query retrieves the distinct names of cities from the **dbo.city** table and orders them alphabetically in ascending order. Let's break down the query and explain each part:

1. **SELECT DISTINCT name AS [City Name]**: This part of the query selects the distinct **name** column from the **dbo.city** table and aliases it as **[City Name]**. The **DISTINCT** keyword ensures that only unique city names are returned.

2. **FROM dbo.city**: This specifies the table from which the data is being retrieved, in this case, the **dbo.city** table.

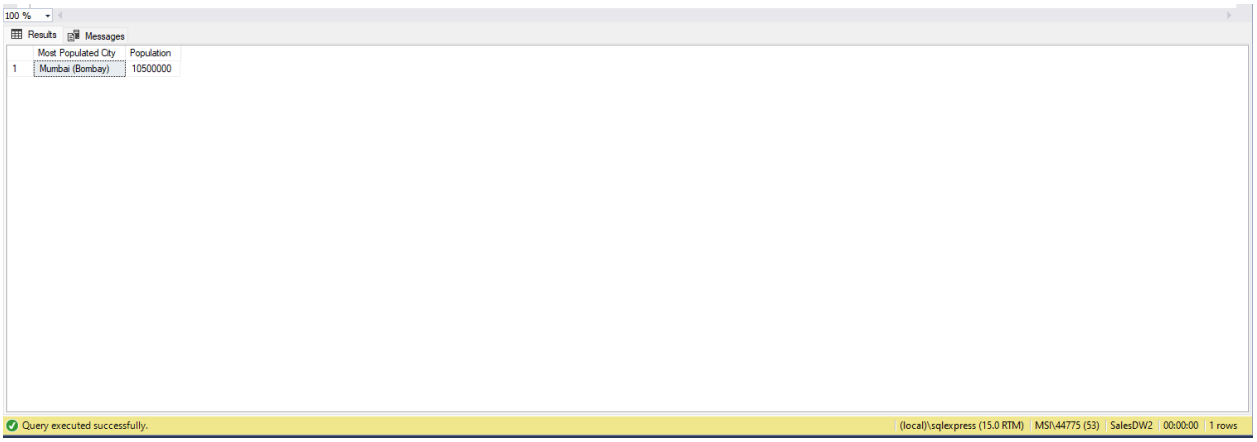
3. **ORDER BY name ASC**: This part of the query orders the result set by the **name** column in ascending order. This ensures that the city names are displayed alphabetically.

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.

Query:

```
SELECT
	Top 1 name AS [Most Populated City],Population
FROM
	dbo.city
ORDER BY
	population DESC;
```

Result Snippet:



Explanation:

This SQL query retrieves the name and population of the most populated city from the **dbo.city** table. Let's break down the query and explain each part:

1. **SELECT TOP 1 name AS [Most Populated City], Population**: This part of the query selects the top 1 record from the result set. It retrieves the **name** column from the **dbo.city** table, aliased as **[Most Populated City]**, and the **Population** column. This selects the name and population of the most populated city.

2. **FROM dbo.city**: This specifies the table from which the data is being retrieved, in this case, the **dbo.city** table.

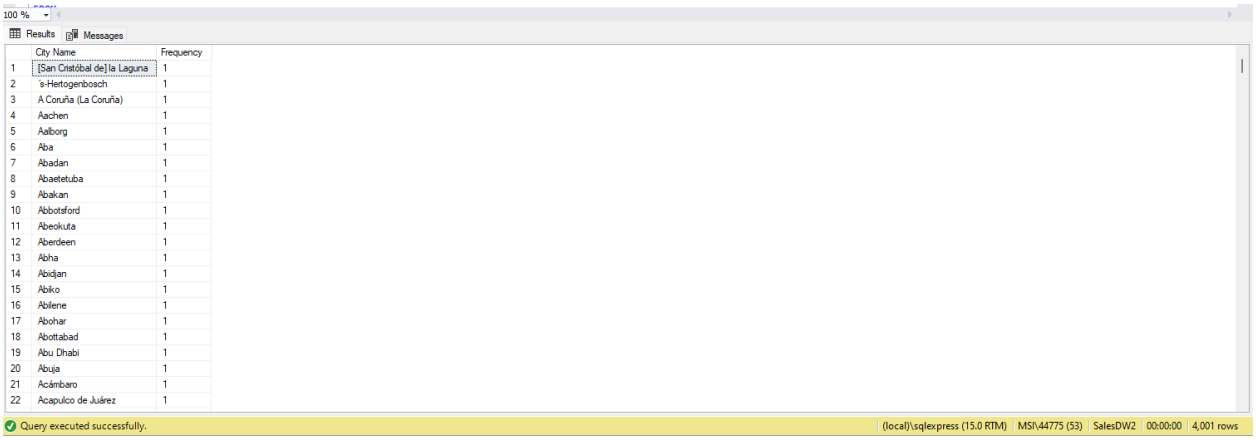
3. **ORDER BY population DESC**: This part of the query orders the result set by the **Population** column in descending order. This means that the city with the highest population will appear at the top of the result set.

10. 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's l.

Query:

```
SELECT
        name AS [City Name], Count(Name) as [Frequency]
FROM
        city
GROUP BY
        name
ORDER BY
        name;
```

Result Snippet:



	City Name	Frequency
1	[San Cristóbal de]la Laguna	1
2	's-Hertogenbosch	1
3	A Coruña (La Coruña)	1
4	Aachen	1
5	Aalborg	1
6	Aba	1
7	Abadan	1
8	Abaetetuba	1
9	Abakan	1
10	Abbotsford	1
11	Abeokuta	1
12	Aberdeen	1
13	Abha	1
14	Abidjan	1
15	Abiko	1
16	Abilene	1
17	Abohar	1
18	Abottabad	1
19	Abu Dhabi	1
20	Abuja	1
21	Acámbaro	1
22	Acapulco de Juárez	1

Explanation:

This SQL query retrieves the names of cities along with their frequencies from the **city** table. Let's break down the query and explain each part:

1. **SELECT name AS [City Name], COUNT(Name) as [Frequency]**: This part of the query selects the **name** column from the **city** table and aliases it as **[City Name]**. Additionally, it uses the **COUNT()** function to count the occurrences of each city name. The result of this count is aliased as **[Frequency]**.

2. **FROM city**: This specifies the table from which the data is being retrieved, in this case, the **city** table.

3. **GROUP BY name**: This part of the query groups the rows by the **name** column. This means that rows with the same city name will be grouped together.

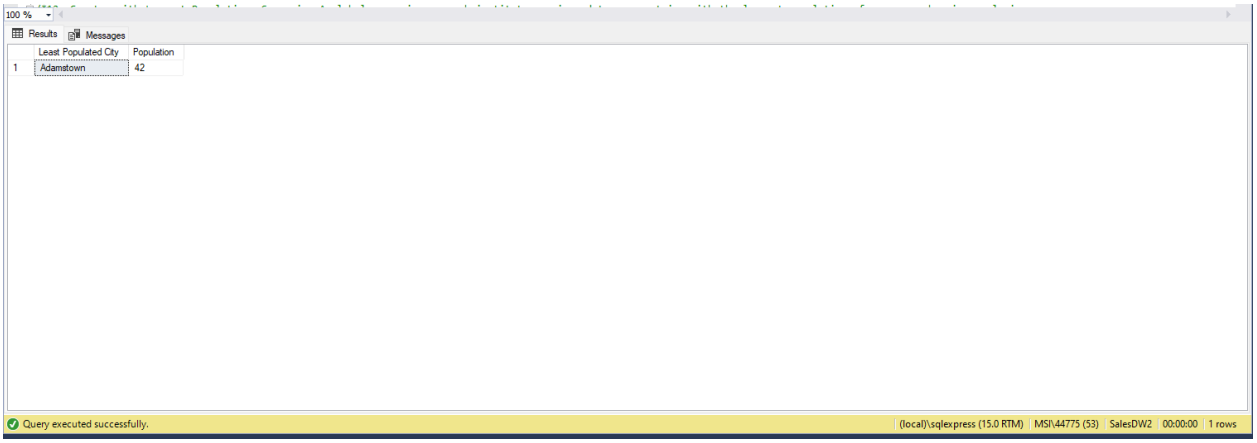
4. **ORDER BY name**: This part of the query orders the result set by the **name** column in ascending order. This ensures that the city names are displayed alphabetically.

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.

Query:

```
SELECT
	TOP 1 name AS [Least Populated City],Population
FROM
	dbo.city
ORDER BY
	population ASC;
```

Result Snippet:



Explanation:

This SQL query retrieves the name and population of the least populated city from the **dbo.city** table. Let's break down the query and explain each part:

1. **SELECT TOP 1 name AS [Least Populated City], Population**: This part of the query selects the top 1 record from the result set. It retrieves the **name** column from the **dbo.city** table, aliased as **[Least Populated City]**, and the **Population** column. This selects the name and population of the least populated city.

2. **FROM dbo.city**: This specifies the table from which the data is being retrieved, in this case, the **dbo.city** table.

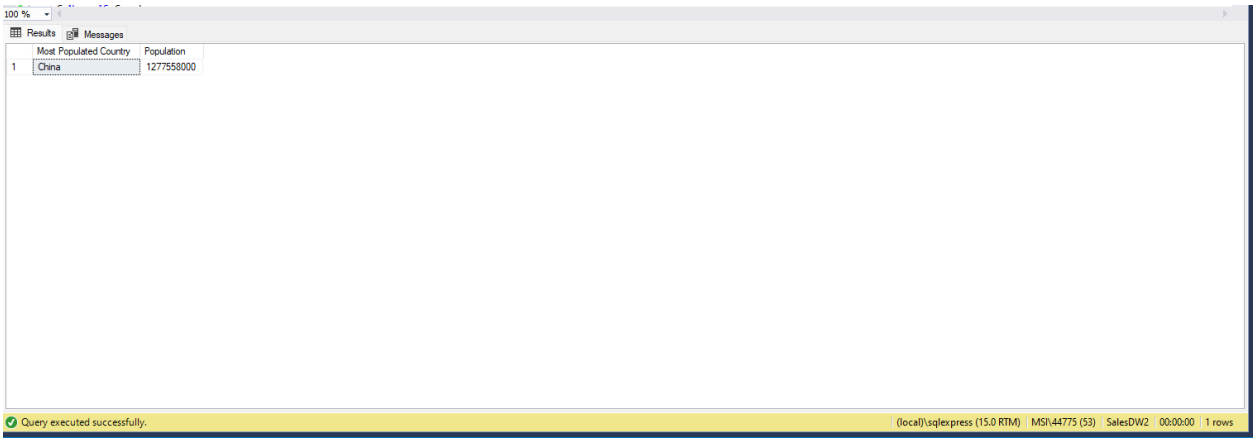
3. **ORDER BY population ASC**: This part of the query orders the result set by the **Population** column in ascending order. This means that the city with the lowest population will appear at the top of the result set.

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.

Query:

```
SELECT
	TOP 1 name AS [Most Populated Country], Population
FROM
	dbo.country
ORDER BY
	population DESC;
```

Result Snippet:



Explanation:

This SQL query retrieves the name and population of the most populated country from the **dbo.country** table. Let's break down the query:

1. **SELECT TOP 1 name AS [Most Populated Country], Population**: This part of the query selects the top 1 record from the result set. It retrieves the **name** column from the **dbo.country** table, aliased as **[Most Populated Country]**, and the **Population** column. This selects the name and population of the most populated country.

2. **FROM dbo.country**: This specifies the table from which the data is being retrieved, in this case, the **dbo.country** table.

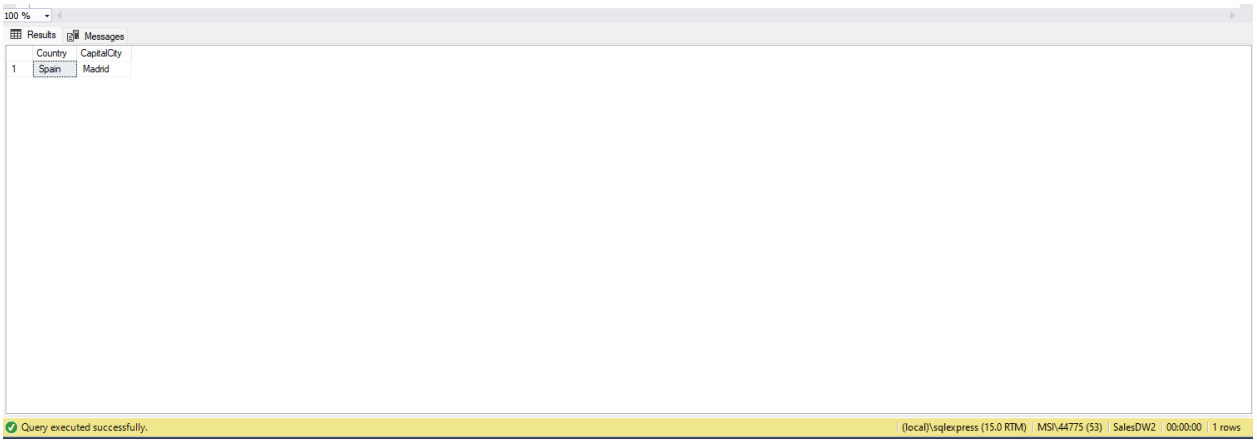
3. **ORDER BY population DESC**: This part of the query orders the result set by the **Population** column in descending order. This means that the country with the highest population will appear at the top of the result set.

13. Capital of Spain: Scenario: A travel agency is organizing 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 travelers with essential destination information.

Query:

```
SELECT
        C.Name AS Country,
        CI.Name AS CapitalCity
FROM
      dbo.country AS C
JOIN
     dbo.city AS CI ON CI.ID = C.Capital
WHERE
        C.countrycode='ESP';
```

Result Snippet:



Explanation:

This SQL query retrieves the name of a country and the name of its capital city from the **dbo.country** and **dbo.city** tables, respectively, for the country with the country code 'ESP'. Let's break down the query:

1. **SELECT C.Name AS Country, CI.Name AS CapitalCity**: This part of the query selects the name of the country from the **dbo.country** table, aliased as **C**, and the name of the capital city from the **dbo.city** table, aliased as **CI**. The aliases are used to differentiate between the two tables when selecting columns.

2. **FROM dbo.country AS C**: This specifies the **dbo.country** table and aliases it as **C**.

3. **JOIN dbo.city AS CI ON CI.ID = C.Capital**: This part of the query performs an inner join between the **dbo.country** table and the **dbo.city** table based on the condition that the **ID** column in the **dbo.city** table matches the **Capital** column in the **dbo.country** table. This join links each country to its capital city.

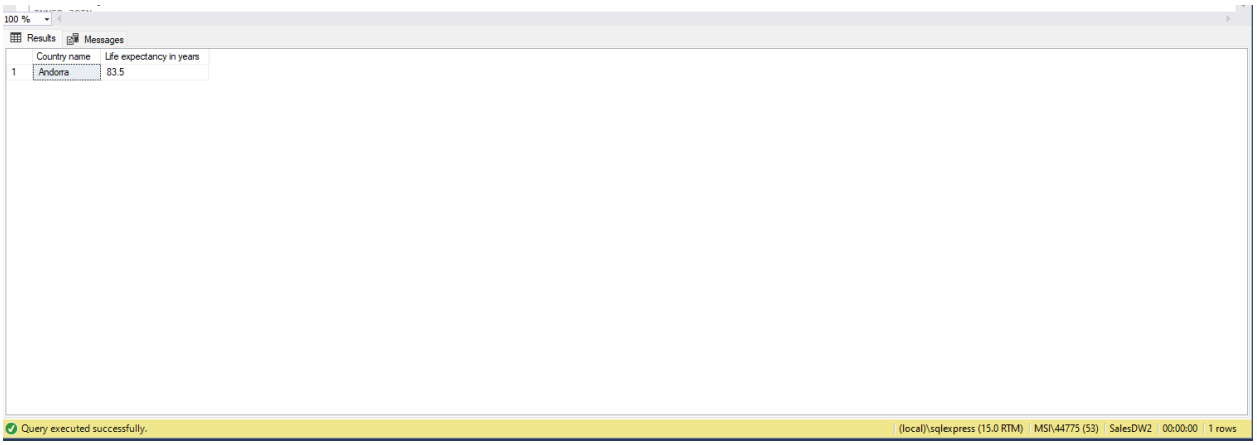
4. **WHERE C.countrycode='ESP'**: This part of the query filters the rows to include only those where the **countrycode** column in the **dbo.country** table equals 'ESP'. This restricts the results to the country with the country code 'ESP'.

14. Country with Highest Life Expectancy: Scenario: A healthcare foundation is conducting research on global health indicators. You're tasked with identifying the country with the highest life expectancy from the database to inform their efforts in improving healthcare systems and policies.

Query:

```
SELECT
        TOP 1 name AS [Country name],
         Lifeexpectancy AS [Life expectancy in years]
FROM
        dbo.country
ORDER BY
         lifeexpectancy DESC;
```

Result Snippet:



Explanation:

This SQL query retrieves the name and life expectancy of the country with the highest life expectancy from the **dbo.country** table. Let's break down the query:

1. **SELECT TOP 1 name AS [Country name], Lifeexpectancy AS [Life expectancy in years]**: This part of the query selects the top 1 record from the result set. It retrieves the **name** column from the **dbo.country** table, aliased as **[Country name]**, and the **Lifeexpectancy** column, aliased as **[Life expectancy in years]**. This selects the name and life expectancy of the country with the highest life expectancy.

2. **FROM dbo.country**: This specifies the table from which the data is being retrieved, in this case, the **dbo.country** table.

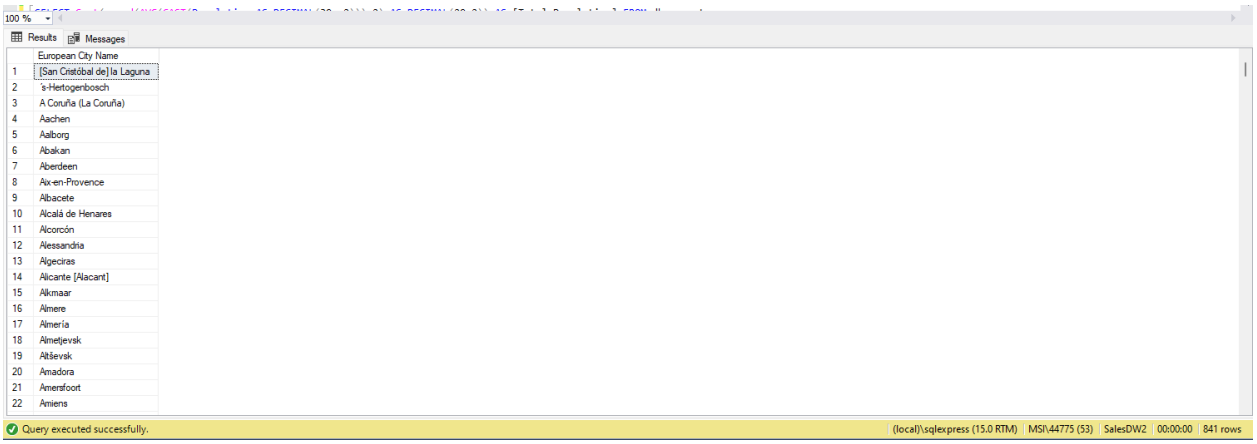
3. **ORDER BY lifeexpectancy DESC**: This part of the query orders the result set by the **Lifeexpectancy** column in descending order. This means that the country with the highest life expectancy will appear at the top of the result set.

15. 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.

Query:

```
SELECT
	C.name AS [European City Name]
FROM
	dbo.city AS C
INNER JOIN
	 dbo.country AS Co
ON
	C.CountryCode = Co.CountryCode
WHERE
	CO.continent = 'Europe'
Order by
	C.name;
```

Result Snippet:



	European City Name
1	[San Cristóbal de] la Laguna
2	s-Hertogenbosch
3	A Coruña (La Coruña)
4	Aachen
5	Aalborg
6	Abakan
7	Aberdeen
8	Aix-en-Provence
9	Albacete
10	Alcalá de Henares
11	Alcorcón
12	Alessandria
13	Algeciras
14	Alicante [Alacant]
15	Alkmaar
16	Almere
17	Almería
18	Almetjevsk
19	Altăevsk
20	Amadora
21	Amersfoort
22	Amiens

Explanation:

This SQL query retrieves the names of European cities from the **dbo.city** table and their corresponding countries from the **dbo.country** table. Let's break down the query:

1. **SELECT C.name AS [European City Name]**: This part of the query selects the name of cities from the **dbo.city** table and aliases it as **[European City Name]**. The alias **C** is used for the **dbo.city** table.

2. **FROM dbo.city AS C**: This specifies the **dbo.city** table and aliases it as **C**.

3. **INNER JOIN dbo.country AS Co ON C.CountryCode = Co.CountryCode**: This part of the query performs an inner join between the **dbo.city** table (**C**) and the **dbo.country** table (**Co**) based on the condition that the **CountryCode** column in both tables matches. This join links each city to its corresponding country.

4. **WHERE CO.continent = 'Europe'**: This part of the query filters the rows to include only those where the **continent** column in the **dbo.country** table equals 'Europe'. This restricts the results to cities located in Europe.

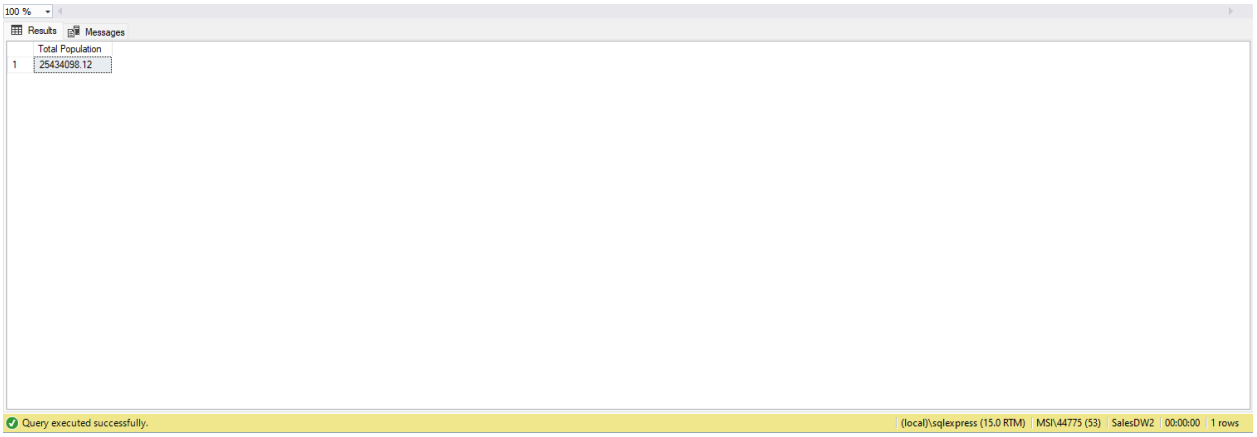
5. **ORDER BY C.name**: This part of the query orders the result set by the **name** column in the **dbo.city** table in ascending order. This ensures that the city names are displayed alphabetically.

16. 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.

Query:

```
SELECT
    Cast(round(AVG(CAST(Population AS DECIMAL(30, 2))),2) AS DECIMAL(20,2))
    AS [Total Population] FROM dbo.country;
```

Result Snippet:



Explanation:

This SQL query calculates the average population of all countries in the **dbo.country** table. Let's break down the query:

1. **SELECT CAST(ROUND(AVG(CAST(Population AS DECIMAL(30, 2))),2) AS DECIMAL(20,2)) AS [Total Population]**: This part of the query calculates the average population of countries.
    - **CAST(Population AS DECIMAL(30, 2))** converts the **Population** column from its original data type to a **DECIMAL** data type with precision 30 and scale 2.
    - **AVG()** calculates the average of the population values.
    - **ROUND(..., 2)** rounds the average population to two decimal places.
    - **CAST(... AS DECIMAL(20, 2))** converts the rounded average back to a **DECIMAL** data type with precision 20 and scale 2.
    - The result is aliased as **[Total Population]**.

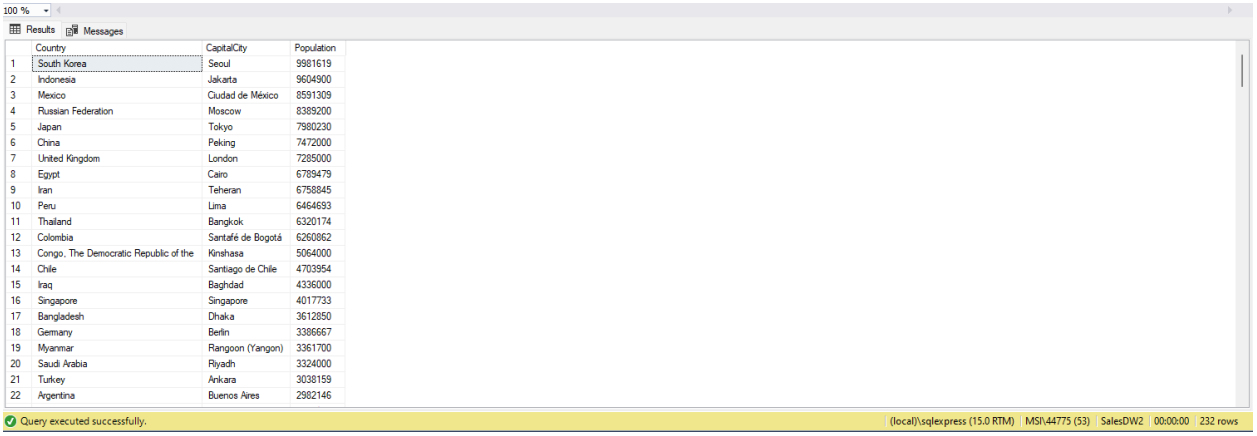
2. **FROM dbo.country**: This specifies the table from which the data is being retrieved, in this case, the **dbo.country** table.

17. 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.

Query:

```
SELECT
        C.Name AS Country,
        CI.Name AS CapitalCity,
        CI.Population AS Population
FROM
         dbo.country AS C
JOIN
        dbo.city AS CI ON CI.ID = C.Capital
ORDER BY
        CI.Population DESC;
```

Result Snippet:



	Country	CapitalCity	Population
1	South Korea	Seoul	9981619
2	Indonesia	Jakarta	9604900
3	Mexico	Ciudad de México	8591309
4	Russian Federation	Moscow	8389200
5	Japan	Tokyo	7980230
6	China	Peking	7472000
7	United Kingdom	London	7285000
8	Egypt	Cairo	6789479
9	Iran	Teheran	6758845
10	Peru	Lima	6464693
11	Thailand	Bangkok	6320174
12	Colombia	Santafé de Bogotá	6260862
13	Congo, The Democratic Republic of the	Kinshasa	5064000
14	Chile	Santiago de Chile	4703954
15	Iraq	Baghdad	4336000
16	Singapore	Singapore	4017733
17	Bangladesh	Dhaka	3612850
18	Germany	Berlin	3386667
19	Myanmar	Rangoon (Yangon)	3361700
20	Saudi Arabia	Riyadh	3324000
21	Turkey	Ankara	3038159
22	Argentina	Buenos Aires	2982146

Explanation:

This SQL query retrieves the name of a country, the name of its capital city, and the population of the capital city from the **dbo.country** and **dbo.city** tables, respectively. Let's break down the query:

1. **SELECT C.Name AS Country, CI.Name AS CapitalCity, CI.Population AS Population**: This part of the query selects the name of the country from the **dbo.country** table, aliased as **C**, the name of the capital city from the **dbo.city** table, aliased as **CI**, and the population of the capital city from the **dbo.city** table.

2. **FROM dbo.country AS C**: This specifies the **dbo.country** table and aliases it as **C**.

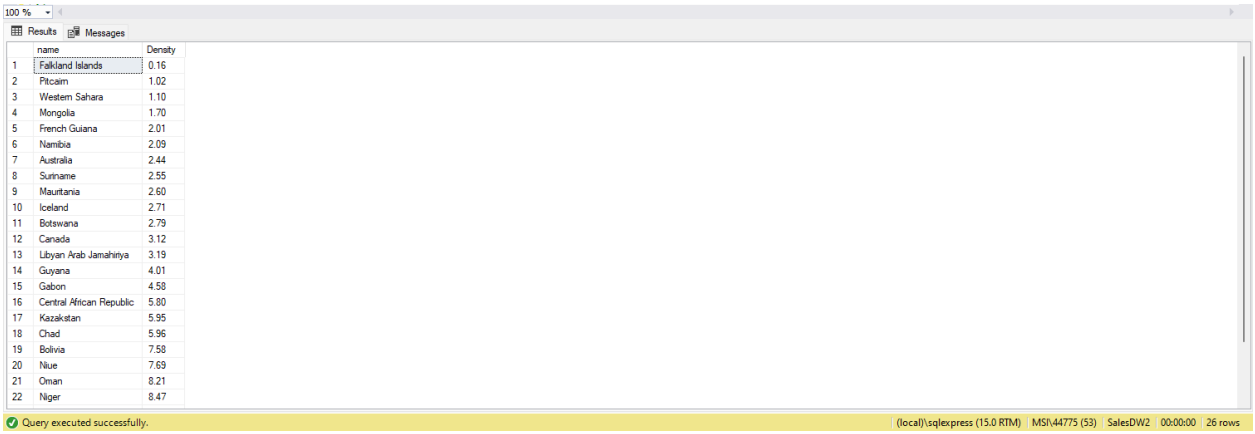
3. **JOIN dbo.city AS CI ON CI.ID = C.Capital**: This part of the query performs an inner join between the **dbo.country** table (**C**) and the **dbo.city** table (**CI**) based on the condition that the **ID** column in the **dbo.city** table matches the **Capital** column in the **dbo.country** table. This join links each country to its capital city.
4. **ORDER BY CI.Population DESC**: This part of the query orders the result set by the **Population** column in the **dbo.city** table in descending order. This ensures that the capital city with the highest population will appear at the top of the result set.

18. 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.

Query:

```
SELECT
        name, (Population/SurfaceArea) AS Density from dbo.country
WHERE
        (Population/SurfaceArea) between 0.1 and 10
Order by
        (Population/SurfaceArea);
```

Result Snippet:



	name	Density
1	Falkland Islands	0.16
2	Pitcairn	1.02
3	Western Sahara	1.10
4	Mongolia	1.70
5	French Guiana	2.01
6	Namibia	2.09
7	Australia	2.44
8	Suriname	2.55
9	Mauritania	2.60
10	Iceland	2.71
11	Botswana	2.79
12	Canada	3.12
13	Libyan Arab Jamahiriya	3.19
14	Guyana	4.01
15	Gabon	4.58
16	Central African Republic	5.80
17	Kazakstan	5.95
18	Chad	5.96
19	Bolivia	7.58
20	Niue	7.69
21	Oman	8.21
22	Niger	8.47

Explanation:

This SQL query retrieves the name of each country and calculates its population density by dividing the population by the surface area. It then filters the result to include only countries with a population density between 0.1 and 10 people per square unit. Finally, it orders the result by population density. Let's break it down:

1. **SELECT name, (Population/SurfaceArea) AS Density FROM dbo.country**: This part selects the name of each country and calculates its population density by dividing the population by the surface area. The result is aliased as **Density**.

2. **WHERE (Population/SurfaceArea) BETWEEN 0.1 AND 10**: This part filters the result to include only rows where the population density falls between 0.1 and 10 people per square unit.

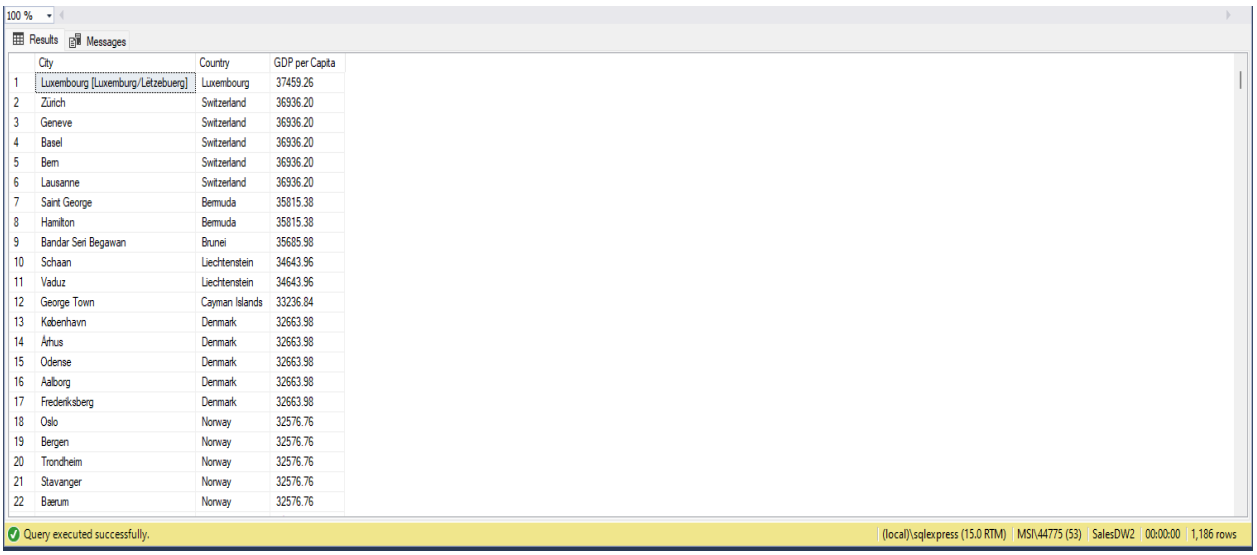
3. **ORDER BY (Population/SurfaceArea)**: This part orders the filtered result by population density in ascending order.

19. Cities with High GDP per Capita: Scenario: An economic consulting firm is analyzing 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.

Query:

```
SELECT
        C.Name AS Country,
        CI.Name AS City,
        CAST(round((((C.GNP*1000000)*(CAST ((CI.population) AS decimal(10,2))/
        C.population)))/CI.Population,2)AS decimal(10,2)) as [GDP per Capita]
FROM
        dbo.country AS C
JOIN
         dbo.city AS CI ON C.countrycode = CI.Countrycode
WHERE
        C.GNP>1 and CAST(round((((C.GNP*1000000)*(CAST ((CI.population) AS decimal(10,2))/
        C.population)))/CI.Population,2)AS decimal(10,2))
        >(SELECT
                AVG(CAST(round((((C.GNP*1000000)*(CAST ((CI.population) AS decimal(10,2))/
                C.population)))/CI.Population,2)AS decimal(10,2)))
          FROM
                dbo.country AS C
          JOIN
                dbo.city AS CI ON C.countrycode = CI.Countrycode)
ORDER BY
        [GDP per Capita] DESC;
```

Result Snippet:



	City	Country	GDP per Capita
1	Luxembourg [Luxemburg/Lëtzebuerg]	Luxembourg	37459.26
2	Zürich	Switzerland	36936.20
3	Geneve	Switzerland	36936.20
4	Basel	Switzerland	36936.20
5	Bern	Switzerland	36936.20
6	Lausanne	Switzerland	36936.20
7	Saint George	Bermuda	35815.38
8	Hamilton	Bermuda	35815.38
9	Bandar Seri Begawan	Brunei	35685.98
10	Schaan	Liechtenstein	34643.96
11	Vaduz	Liechtenstein	34643.96
12	George Town	Cayman Islands	33236.84
13	København	Denmark	32663.98
14	Århus	Denmark	32663.98
15	Odense	Denmark	32663.98
16	Aalborg	Denmark	32663.98
17	Frederiksberg	Denmark	32663.98
18	Oslo	Norway	32576.76
19	Bergen	Norway	32576.76
20	Trondheim	Norway	32576.76
21	Stavanger	Norway	32576.76
22	Bærum	Norway	32576.76

Explanation:

This SQL query calculates the GDP per capita for each city in each country, filters the result to include only cities where the GDP per capita is greater than the average GDP per capita across all cities, and orders the result by GDP per capita in descending order. Let's break it down:

1. **SELECT C.Name AS Country, CI.Name AS City, ... AS [GDP per Capita]**: This part selects the name of the country, the name of the city, and calculates the GDP per capita for each city in each country. The calculation involves the country's GNP (Gross National Product), city population, and country population.

2. **FROM dbo.country AS C JOIN dbo.city AS CI ON C.countrycode = CI.Countrycode**: This part performs an inner join between the **dbo.country** table (**C**) and the **dbo.city** table (**CI**) based on the **countrycode** column. This join links each country to its cities.

3. **WHERE C.GNP > 1 AND ... > (SELECT AVG(...))**: This part filters the result to include only rows where the country's GNP is greater than 1 and the calculated GDP per capita is greater than the average GDP per capita across all cities. This is done using a subquery to calculate the average GDP per capita.

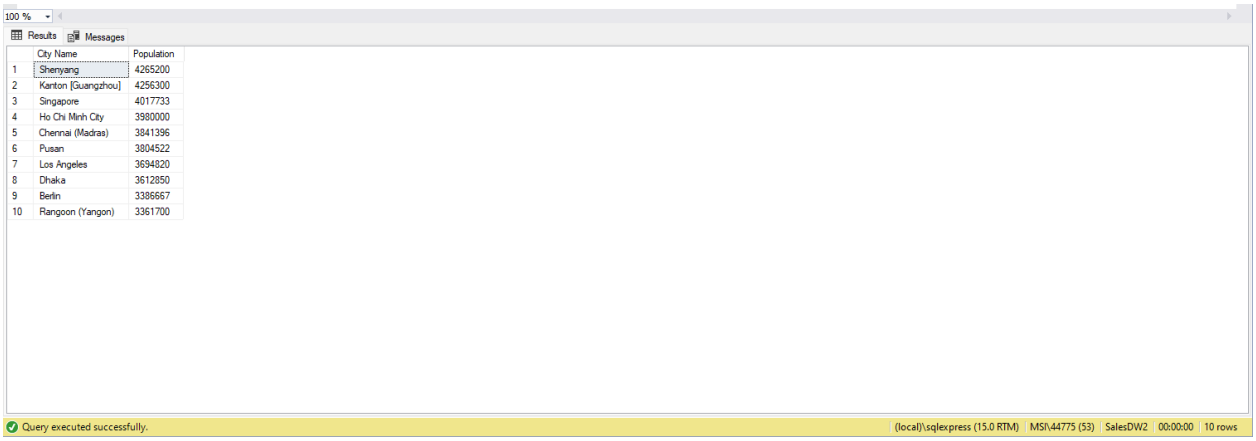
4. **ORDER BY [GDP per Capita] DESC**: This part orders the filtered result by GDP per capita in descending order.

20. 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.

Query:

```
SELECT
	Name AS [City Name],
	 Population
FROM
	dbo.city
ORDER BY
	Population DESC
OFFSET 30 ROWS
FETCH NEXT 10 ROWS ONLY;
```

Result Snippet:



	City Name	Population
1	Shenyang	4265200
2	Kanton [Guangzhou]	4256300
3	Singapore	4017733
4	Ho Chi Minh City	3980000
5	Chennai (Madras)	3841396
6	Pusan	3804522
7	Los Angeles	3694820
8	Dhaka	3612850
9	Berlin	3386667
10	Rangoon (Yangon)	3361700

Explanation:

This SQL query retrieves the names and populations of cities from the **dbo.city** table, orders them by population in descending order, skips the first 30 rows, and then fetches the next 10 rows only. Let's break it down:

1. **SELECT Name AS [City Name], Population**: This part selects the **Name** column from the **dbo.city** table and aliases it as **[City Name]**. It also selects the **Population** column.

2. **FROM dbo.city**: This specifies the table from which the data is being retrieved, in this case, the **dbo.city** table.

3. **ORDER BY Population DESC**: This part orders the result set by the **Population** column in descending order. This ensures that the cities with the highest populations will appear first.

4. **OFFSET 30 ROWS**: This part skips the first 30 rows from the ordered result set. In other words, it starts retrieving rows from the 31st row onwards.

5. **FETCH NEXT 10 ROWS ONLY**: This part fetches the next 10 rows from the result set after skipping the first 30 rows. It ensures that only 10 rows are returned in the final result set.

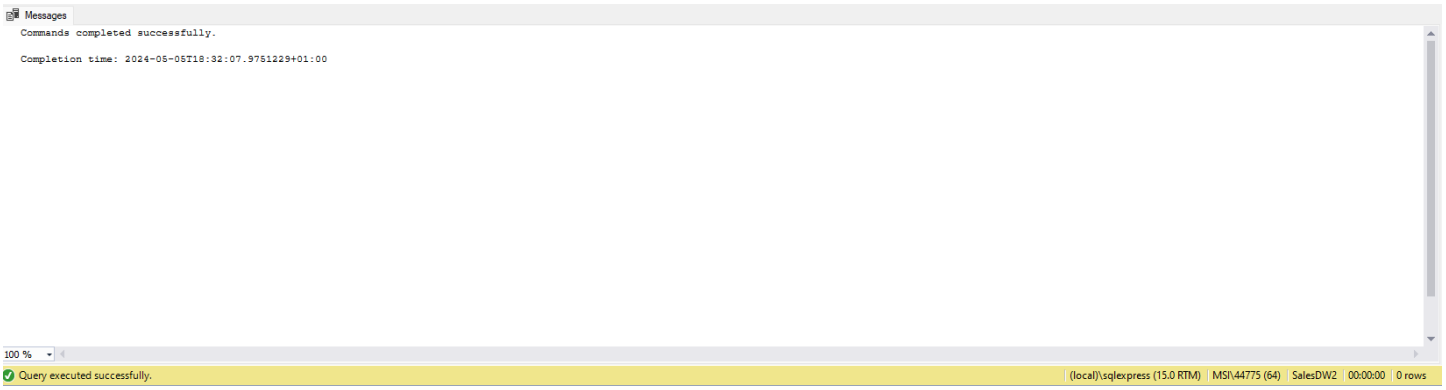
PART B : DATABASE INSPECTION/UPDATION

1. **Specifying the Database Context.**

Query:

```
USE SalesDW2;
```

Result Snippet:



Explanation:

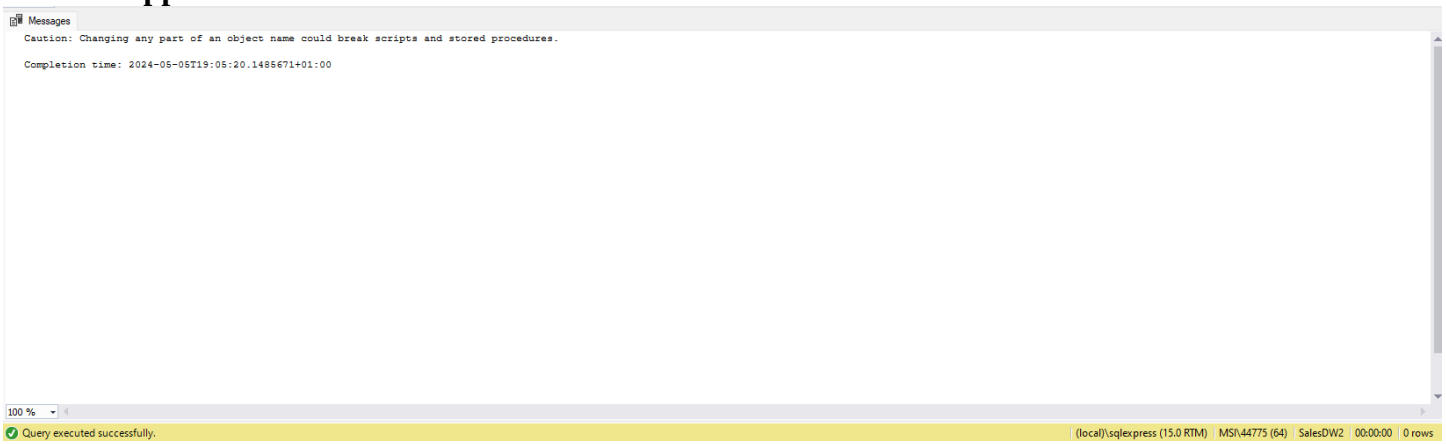
When you execute "USE SalesDW2;", it instructs the SQL Server to set the "SalesDW2" database as the current database for subsequent queries in the same session.

2. **Stored procedure to rename the column 'code' to 'CountryCODE'.**

Query:

```
EXEC sp_rename 'dbo.country.code', 'CountryCode', 'COLUMN';
```

Result Snippet:



Explanation:

This SQL statement executes the stored procedure **sp_rename** to rename a column in the **dbo.country** table from **code** to **CountryCode**.

Let's break down the components of the statement:

1. **EXEC sp_rename**: This is the syntax for executing a stored procedure named **sp_rename**, which is used to rename database objects.

2. **'dbo.country.code'**: This is the current name of the column to be renamed. It specifies the column **code** in the **dbo.country** table.

3. **'CountryCode'**: This is the new name that the column will be renamed to.

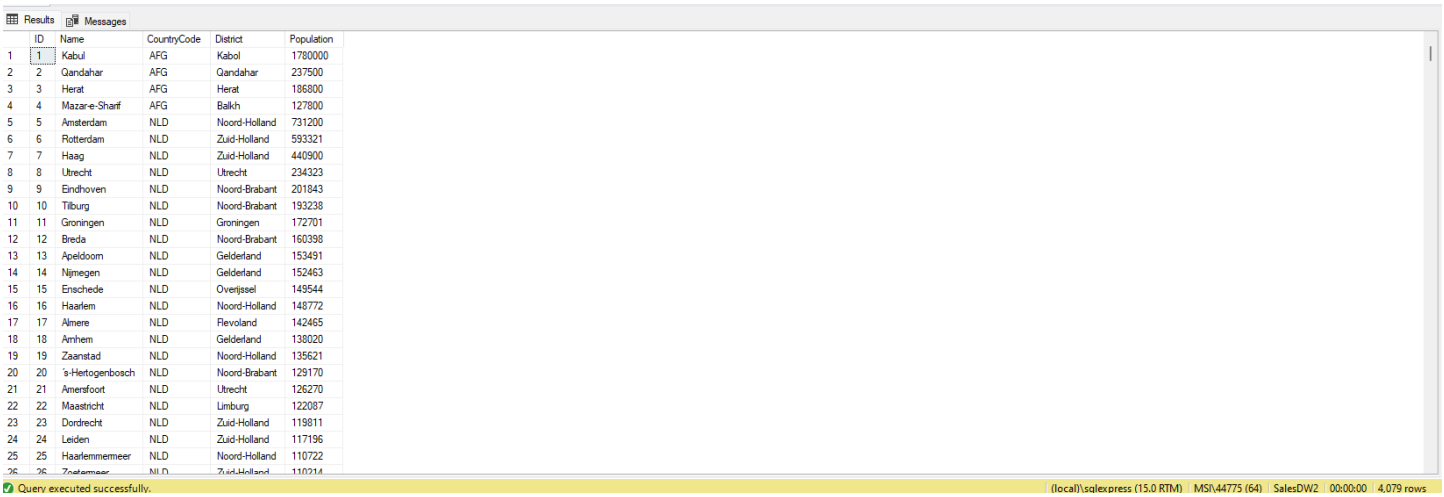
4. **'COLUMN'**: This specifies that the object being renamed is a column.

3. Inspecting the table dbo.city.

Query:

```
SELECT * FROM dbo.city;
```

Result Snippet:



Results | Messages

	ID	Name	CountryCode	District	Population
1	1	Kabul	AFG	Kabol	1780000
2	2	Qandahar	AFG	Qandahar	237500
3	3	Herat	AFG	Herat	186800
4	4	Mazar-e-Sharif	AFG	Balkh	127800
5	5	Amsterdam	NLD	Noord-Holland	731200
6	6	Rotterdam	NLD	Zuid-Holland	593321
7	7	Haag	NLD	Zuid-Holland	440900
8	8	Utrecht	NLD	Utrecht	234323
9	9	Eindhoven	NLD	Noord-Brabant	201843
10	10	Tilburg	NLD	Noord-Brabant	193238
11	11	Groningen	NLD	Groningen	172701
12	12	Breda	NLD	Noord-Brabant	160398
13	13	Apeldoorn	NLD	Gelderland	153491
14	14	Nijmegen	NLD	Gelderland	152463
15	15	Enschede	NLD	Overijssel	149544
16	16	Haarlem	NLD	Noord-Holland	148772
17	17	Almere	NLD	Flevoland	142465
18	18	Arnhem	NLD	Gelderland	138020
19	19	Zaanstad	NLD	Noord-Holland	135621
20	20	's-Hertogenbosch	NLD	Noord-Brabant	129170
21	21	Amersfoort	NLD	Utrecht	126270
22	22	Maastricht	NLD	Limburg	122087
23	23	Dordrecht	NLD	Zuid-Holland	119811
24	24	Leiden	NLD	Zuid-Holland	117196
25	25	Haarlemmermeer	NLD	Noord-Holland	110722
26	26	Zoetermeer	NLD	Zuid-Holland	110214

Query executed successfully. (local)\sqlexpress (15.0 RTM) | MSI\44775 (64) | SalesDW2 | 00:00:00 | 4,079 rows

Explanation:

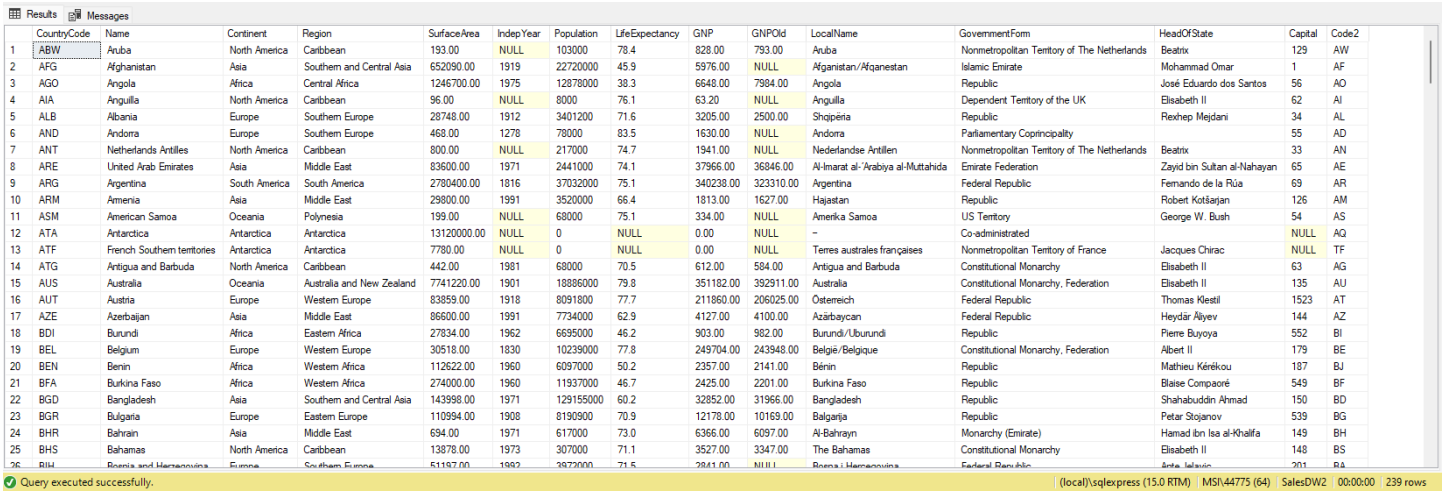
This query selects all columns and all rows from the **dbo.city** table.

4. Inspecting the table dbo.country.

Query:

```
SELECT * FROM dbo.country;
```

Result Snippet:



Results Messages

	CountryCode	Name	Continent	Region	SurfaceArea	IndepYear	Population	LifeExpectancy	GNP	GNPOld	LocalName	GovernmentForm	HeadOfState	Capital	Code2
1	ABW	Aruba	North America	Caribbean	193.00	NULL	103000	78.4	828.00	793.00	Aruba	Nonmetropolitan Territory of The Netherlands	Beatrix	129	AW
2	AFG	Afghanistan	Asia	Southern and Central Asia	652090.00	1919	22720000	45.9	5976.00	NULL	Afganistan/Afqanestan	Islamic Emirate	Mohammad Omar	1	AF
3	AGO	Angola	Africa	Central Africa	1246700.00	1975	12878000	38.3	6648.00	7984.00	Angola	Republic	José Eduardo dos Santos	56	AO
4	AIA	Anguilla	North America	Caribbean	96.00	NULL	8000	76.1	63.20	NULL	Anguilla	Dependent Territory of the UK	Elisabeth II	62	AI
5	ALB	Albania	Europe	Southern Europe	28748.00	1912	3401200	71.6	3205.00	2500.00	Shqipëria	Republic	Rexhep Mejdani	34	AL
6	AND	Andorra	Europe	Southern Europe	468.00	1278	78000	83.5	1630.00	NULL	Andorra	Parliamentary Coprincipality		55	AD
7	ANT	Netherlands Antilles	North America	Caribbean	800.00	NULL	217000	74.7	1941.00	NULL	Nederlandse Antillen	Nonmetropolitan Territory of The Netherlands	Beatrix	33	AN
8	ARE	United Arab Emirates	Asia	Middle East	83600.00	1971	2441000	74.1	37966.00	36846.00	Al-Imarat al-'Arabiya al-Muttahida	Emirate Federation	Zayid bin Sultan al-Nahayan	65	AE
9	ARG	Argentina	South America	South America	2780400.00	1816	37032000	75.1	340238.00	323310.00	Argentina	Federal Republic	Fernando de la Rúa	69	AR
10	ARM	Armenia	Asia	Middle East	29800.00	1991	3520000	66.4	1813.00	1627.00	Hajastan	Republic	Robert Kotšarjan	126	AM
11	ASM	American Samoa	Oceania	Polynesia	199.00	NULL	68000	75.1	334.00	NULL	Amerika Samoa	US Territory	George W. Bush	54	AS
12	ATA	Antarctica	Antarctica	Antarctica	13120000.00	NULL	0	NULL	0.00	NULL	–	Co-administrated		NULL	AQ
13	ATF	French Southern territories	Antarctica	Antarctica	7780.00	NULL	0	NULL	0.00	NULL	Terres australes françaises	Nonmetropolitan Territory of France	Jacques Chirac	NULL	TF
14	ATG	Antigua and Barbuda	North America	Caribbean	442.00	1981	68000	70.5	612.00	584.00	Antigua and Barbuda	Constitutional Monarchy	Elisabeth II	63	AG
15	AUS	Australia	Oceania	Australia and New Zealand	7741220.00	1901	18886000	79.8	351182.00	392911.00	Australia	Constitutional Monarchy, Federation	Elisabeth II	135	AU
16	AUT	Austria	Europe	Western Europe	83859.00	1918	8091800	77.7	211860.00	206025.00	Österreich	Federal Republic	Thomas Klestil	1523	AT
17	AZE	Azerbaijan	Asia	Middle East	86600.00	1991	7734000	62.9	4127.00	4100.00	Azärbaycan	Federal Republic	Heydär Äliyev	144	AZ
18	BDI	Burundi	Africa	Eastern Africa	27834.00	1962	6695000	46.2	903.00	982.00	Burundi/Uburundi	Republic	Pierre Buyoya	552	BI
19	BEL	Belgium	Europe	Western Europe	30518.00	1830	10239000	77.8	249704.00	243948.00	België/Belgique	Constitutional Monarchy, Federation	Albert II	179	BE
20	BEN	Benin	Africa	Western Africa	112622.00	1960	6097000	50.2	2357.00	2141.00	Bénin	Republic	Mathieu Kérékou	187	BJ
21	BFA	Burkina Faso	Africa	Western Africa	274000.00	1960	11937000	46.7	2425.00	2201.00	Burkina Faso	Republic	Blaise Compaoré	549	BF
22	BGD	Bangladesh	Asia	Southern and Central Asia	143998.00	1971	129155000	60.2	32852.00	31966.00	Bangladesh	Republic	Shahabuddin Ahmad	150	BD
23	BGR	Bulgaria	Europe	Eastern Europe	110994.00	1908	8190900	70.9	12178.00	10169.00	Balgarija	Republic	Petar Stojanov	539	BG
24	BHR	Bahrain	Asia	Middle East	694.00	1971	617000	73.0	6366.00	6097.00	Al-Bahrayn	Monarchy (Emirate)	Hamad ibn Isa al-Khalifa	149	BH
25	BHS	Bahamas	North America	Caribbean	13878.00	1973	307000	71.1	3527.00	3347.00	The Bahamas	Constitutional Monarchy	Elisabeth II	148	BS
26	BIH	Bosnia and Hercegovina	Europe	Southern Europe	51197.00	1992	3972000	71.5	2841.00	NULL	Bosna i Hercegovina	Federal Republic	Arta Jelavic	201	BA

Query executed successfully. (local)\sqlexpress (15.0 RTM) MSI\44775 (64) SalesDW2 00:00:00 239 rows

Explanation:

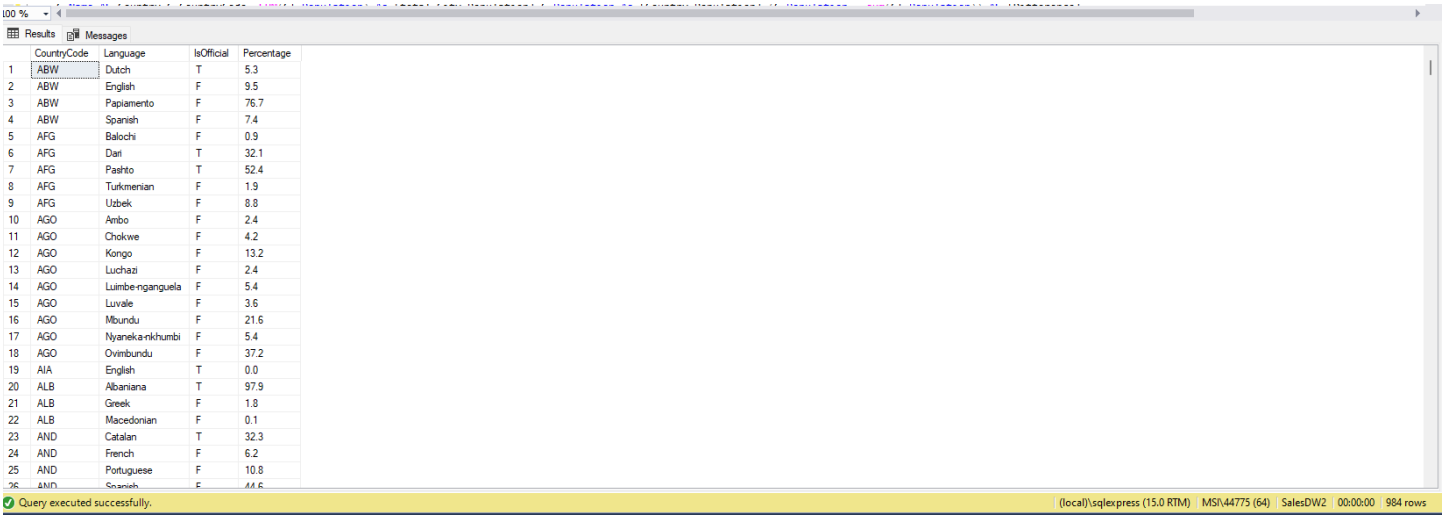
This query selects all columns and all rows from the **dbo.country** table.

5. Inspecting the table dbo.countryLanguage

Query:

```
SELECT * FROM dbo.countryLanguage;
```

Result Snippet:



100 %

Results Messages

	CountryCode	Language	IsOfficial	Percentage
1	ABW	Dutch	T	5.3
2	ABW	English	F	9.5
3	ABW	Papiamento	F	76.7
4	ABW	Spanish	F	7.4
5	AFG	Balochi	F	0.9
6	AFG	Dari	T	32.1
7	AFG	Pashto	T	52.4
8	AFG	Turkmenian	F	1.9
9	AFG	Uzbek	F	8.8
10	AGO	Ambo	F	2.4
11	AGO	Chokwe	F	4.2
12	AGO	Kongo	F	13.2
13	AGO	Luchazi	F	2.4
14	AGO	Luimbe-nganguela	F	5.4
15	AGO	Luvale	F	3.6
16	AGO	Mbundu	F	21.6
17	AGO	Nyaneka-nkhumbi	F	5.4
18	AGO	Ovimbundu	F	37.2
19	AIA	English	T	0.0
20	ALB	Albaniana	T	97.9
21	ALB	Greek	F	1.8
22	ALB	Macedonian	F	0.1
23	AND	Catalan	T	32.3
24	AND	French	F	6.2
25	AND	Portuguese	F	10.8
26	AND	Spanish	F	44.6

Query executed successfully. (local)\sqlexpress (15.0 RTM) MSI\44775 (64) SalesDW2 00:00:00 984 rows

Explanation:

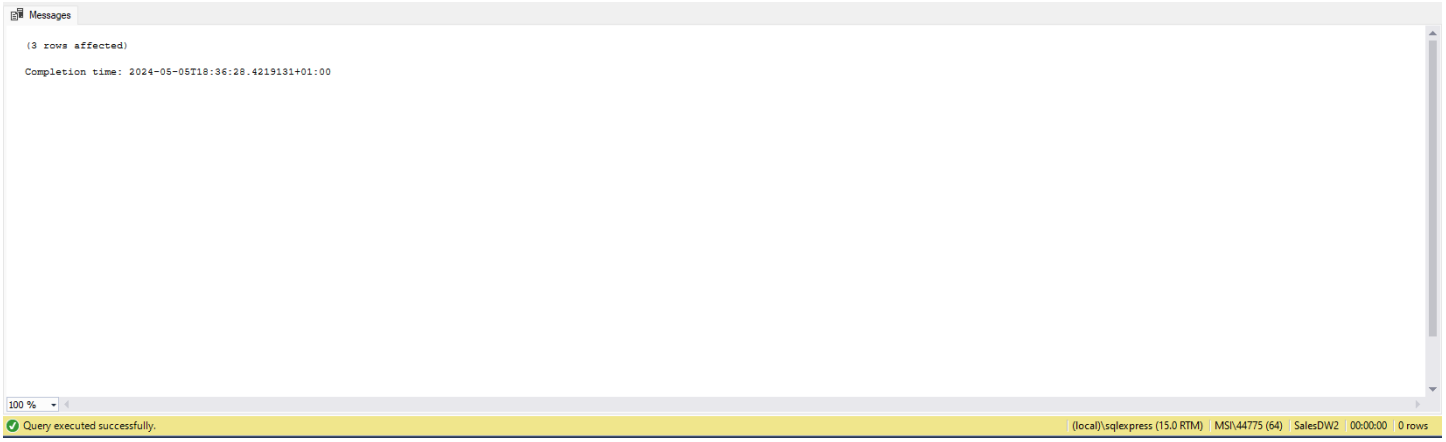
This query selects all columns and all rows from the **dbo.country** table.

6. Updating the Column 'Population' in Country table where the Country population is less than City Population.

Query:

```
UPDATE
	dbo.country
SET
	Population = CI.Population
FROM
	Country as C
JOIN
	City As CI
 ON
	C.CountryCode = CI.CountryCode
WHERE
	 CI.CountryCode in(
				SELECT
					C.CountryCode
				FROM
					dbo.country AS C
				JOIN
					dbo.city AS CI
				ON
				    C.countrycode = CI.Countrycode
				WHERE
					C.name = C.name
				GROUP BY
					C.Name, C.Countrycode, c.Population
				HAVING
					(C.Population - sum(CI.Population)) <= 0);
```

Result Snippet:



Explanation:

This SQL query updates the **Population** column in the **dbo.country** table with the population of the corresponding city from the **dbo.city** table for countries where the city population is greater than the country population. Let's break it down:

1. **UPDATE dbo.country SET Population = CI.Population**: This part of the query specifies that the **Population** column in the **dbo.country** table will be updated with the population value from the **CI** alias, which represents the **dbo.city** table.

2. **FROM Country AS C JOIN City AS CI ON C.CountryCode = CI.CountryCode**: This part of the query joins the **dbo.country** table (**Country**) with the **dbo.city** table (**City**) based on the **CountryCode** column, linking each country to its corresponding cities.

3. **WHERE CI.CountryCode IN (...)**: This part of the query filters the rows to be updated based on a subquery. The subquery selects country codes where the condition **(C.Population - SUM(CI.Population)) <= 0** holds true, meaning that the city population is greater than the country population.

4. **SELECT C.CountryCode ... HAVING (C.Population - SUM(CI.Population)) <= 0**: This subquery selects country codes from the joined tables where the city population is greater than the country population. It groups the results by country name, country code, and country population and applies the condition in the HAVING clause to filter the results.