|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | | | |
|  | | USING SQL TO QUERY ‘WORLD DATA BASE’ | | | | |  | |
|  |  | | | | | | |  |
|  | | | |  |  | | | |
|  | | | | Charlie Thomas |  | | | |
|  | | | | **05/05/2024** **SQL Assignment** **Special Thanks to Mr. Yusuf Satilmis** |  | | | |
|  | | |  | | |  | | |

**Table of Content**

|  |  |  |
| --- | --- | --- |
| **Index** | **Particular** | **Page No.** |
| **1** | **PART A : ASSIGNMENT QUESTIONS** | **3** |
| **2** | Question 1. Count Cities in USA | **4** |
| **3** | Question 2. Country with Highest Life Expectancy | **5** |
| **4** | Question 3. "New Year Promotion: Featuring Cities with 'New’ | **6** |
| **5** | Question 4. Top 10most populous cities in the world | **7** |
| **6** | Question 5. Cities with Population Larger than 2,000,000 | **8** |
| **7** | Question 6. Cities Beginning with 'Be' Prefix | **9** |
| **8** | Question 7. Cities with Population Between 500,000-1,000,000 | **10** |
| **9** | Question 8. Display Cities Sorted by Name in Ascending Order | **11** |
| **10** | Question 9. Most Populated City | **12** |
| **11** | Question 10. City Name Frequency Analysis | **13** |
| **12** | Question 11. City with the Lowest Population | **14** |
| **13** | Question 12. Country with Largest Population | **15** |
| **14** | Question 13. Capital of Spain | **16** |
| **15** | Question 14. Country with Highest Life Expectancy | **17** |
| **16** | Question 15. List of Cities in Europe | **18** |
| **17** | Question 16. Average Population by Country | **20** |
| **18** | Question 17. Capital Cities Population Comparison | **21** |
| **19** | Question 18. Countries with Low Population Density(Benchmark:10) | **22** |
| **20** | Question 19. Cities with Above Average GDP per Capita | **23** |
| **21** | Question 20. Display cities ranked between 31st and 40th by population | **25** |
|  |  |  |
| **22** | **PART B: DATABASE INSPECTION AND UPDATION** | **26** |
| **23** | Specifying the Database Context. | **27** |
| **24** | Stored procedure to rename the column 'code' to 'Country CODE'. | **27** |
| **25** | Inspecting the table dbo.city | **28** |
| **26** | Inspecting the table dbo.country. | **29** |
| **27** | Inspecting the table dbo.countryLanguage | **29** |
| **28** | Updating the Column 'Population' in Country table where the Country population is less than City Population. | **30** |
|  |  |  |
|  |  |  |

**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:**

A white rectangular object with yellow border

Description automatically generated

**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:**

A white screen with yellow and white text

Description automatically generated with medium confidence

**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:**

A white screen with yellow and grey lines

Description automatically generated with medium confidence

**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:**

A white screen with a yellow border

Description automatically generated

**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:**

A screen shot of a computer

Description automatically generated

**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:**

A white screen with a yellow border

Description automatically generated

**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:**

A white screen with a yellow border

Description automatically generated

**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:**

A white screen with a yellow border

Description automatically generated

**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:**

A white screen with a yellow border

Description automatically generated

**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:**

A white screen with yellow border

Description automatically generated

**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:**

A white screen with a yellow border

Description automatically generated with medium confidence

**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:**

A white screen with a yellow border

Description automatically generated

**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:**

A white screen with a yellow border

Description automatically generated

**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:**

A white screen with yellow and white text

Description automatically generated with medium confidence

**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:**

A screenshot of a computer

Description automatically generated

**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:**

A white rectangular object with yellow border

Description automatically generated

**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:**

A white rectangular object with yellow lines

Description automatically generated with medium confidence

**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:**

A screenshot of a computer

Description automatically generated

**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:**

A white screen with yellow and black text

Description automatically generated with medium confidence

**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:**

A white screen with yellow border

Description automatically generated

**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:**

A white rectangular object with yellow edges

Description automatically generated

**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.

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

**Query:**

EXEC sp\_rename 'dbo.country.code', 'CountryCode', 'COLUMN';

**Result Snippet:**

**A screenshot of a computer

Description automatically generated**

**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.
5. **Inspecting the table dbo.city.**

**Query:**

SELECT \* FROM dbo.city;

**Result Snippet:**

**A white screen with a yellow border

Description automatically generated**

**Explanation:**

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

1. **Inspecting the table dbo.country.**

**Query:**

SELECT \* FROM dbo.country;

**Result Snippet:**

**A screenshot of a computer

Description automatically generated**

**Explanation:**

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

1. **Inspecting the table dbo.countryLanguage**

**Query:**

SELECT \* FROM dbo.countryLanguage;

**Result Snippet:**

**A white screen with a yellow border

Description automatically generated**

**Explanation:**

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

1. **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:**

A white rectangular object with a yellow border

Description automatically generated

**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.