1. Background information

The **World database** contains 3 tables that contain

- **Countries:** Details like population, continent, government form, and their capital city.
- **Cities:** Information about major cities, including their population and the country they belong to.
- **Languages:** Which languages are spoken in which countries, their official status, and the percentage of speakers.

The world database can be **downloaded** from https://dev.mysql.com/doc/index-other.html

2. Relationship Between the Tables

Understanding how tables are linked is crucial for writing correct joins. The World database has three main tables with the following relationships:

2.1. Country. Code and City.countrycode

- **Relationship Type:** One-to-Many (1:M)
- **Description:** One country can have many city records. Each city belongs to exactly one country.
- **How they connect:**
  - country.Code (Primary Key in country table - e.g., 'USA', 'IND')
  - city.CountryCode (Foreign Key in city table, referencing country.Code)
  -

2.2. Country. capital and city.ID

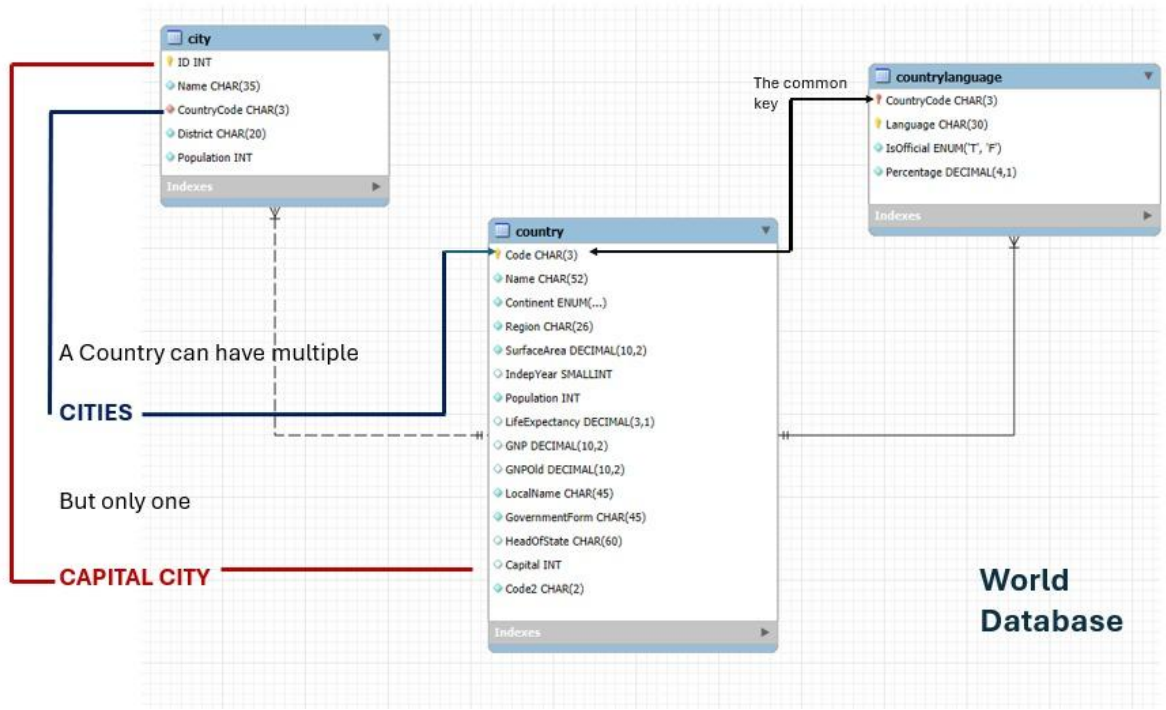
- **(Capital City Relationship - A Specific Link)**
- **Relationship Type:** One-to-One (1:1) from country to city since a city can be a capital for at most one country.
- **Description:** Each country has one designated capital city.
- **How they connect:**
  - country.Capital (Foreign Key in country table, referencing city.ID)
  - city.ID (Primary Key in city table - unique ID for each city)

2.3. Country and Countrylanguage

- **Languages Spoken - Many-to-Many Resolution**
- **Relationship Type:** Many-to-Many (M:M), resolved by the countrylanguage table.
- **Description:** A country can have many languages, and a language can be spoken in many countries. The countrylanguage table acts as a **junction table** to link them.
- **How they connect:**

- country.Code (Primary Key in country table)
- countrylanguage.CountryCode (Foreign Key in countrylanguage table, referencing country.Code)

3. An E-R diagram with the relationship



4. Writing joins on the world database

4.1. INNER JOIN

Returns only the rows that have **matching values in both tables** based on the join condition. If a record in one table doesn't have a match in the other, it's excluded.

Question: List the countries along with their capital, in Europe, where people generally live more than 80 years.

Corresponding SQL:

```sql
SELECT
    c.Name AS Country_Name,
    ci.Name AS City_Name,
    c.LifeExpectancy AS Life_Expectancy
FROM
    country c
INNER JOIN
    city ci ON c.Capital = ci.ID
WHERE
    c.Continent = 'Europe'
    AND c.LifeExpectancy > 80;
```

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

Country_Name	City_Name	Life_Expectancy
Andorra	Andorra la Vella	83.5
San Marino	San Marino	81.1

Assignment:

Create two tables and country and City, with the contents shown below. Inner join the 2 tables

- Inner Join on Capital ID
- Inner Join on Countrycode

Note and explain the o/p in each case.

CITY			
ID	**Name**	**CountryCode**	**population**
123	New Delhi	IND	413339
A12	Chennai	IND	643667
B13	Bengaluru	IND	530489
C14	Hyderabad	IND	884093
245	London	UK	773683
BT5	Birmingham	UK	263691
C98	Leeds	UK	802093
567	Canberra	AUS	722159
G45	Melbourna	AUS	475787
L76	Sydney	AUS	404484

COUNTRY			
Code	**Name**	**Continent**	**Capital**
IND	India	Asia	123
UK	United Kir	Europe	245
AUS	Australia	Asia Pacific	567

4.2. LEFT JOIN (or LEFT OUTER JOIN)

Returns **all rows from the left table**, and the matching rows from the right table. If there's no match in the right table for a row in the left table, NULL values will be returned for the columns from the right table. Think: "Keep everything from the left, add matches from the right if they exist."

Question: **(a)** Give me a list of all countries and the names of their capital city, **(b)** how many countries do not have a capital.

Solution (a)

```
24    -- List all countries along with their capital cities (if available)
25    SELECT
26        c.Name AS Country_Name,
27        ci.Name AS Capital_City
28    FROM
29        country AS c
30    LEFT JOIN
31        city AS ci
32        ON c.Capital = ci.ID;
33
```

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

Country_Name	Capital_City
Aruba	Oranjestad
Afghanistan	Kabul
Angola	Luanda
Anguilla	The Valley
Albania	Tirana
Andorra	Andorra la Vella
Netherlands Antilles	Willemstad
United Arab Emirates	Abu Dhabi
Argentina	Buenos Aires
Armenia	Yerevan

Solution (b)

```
24    -- List all countries along with their capital cities (if available)
25    SELECT
26        c.Name AS Country_Name,
27        ci.Name AS Capital_City
28    FROM
29        country AS c
30    LEFT JOIN
31        city AS ci
32        ON c.Capital = ci.ID
33    WHERE
34        ci.Name IS NULL;
35
36
```

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

Country_Name	Capital_City
Antarctica	NULL
French Southern territories	NULL
Bouvet Island	NULL
Heard Island and McDonald Islands	NULL
British Indian Ocean Territory	NULL
South Georgia and the South Sandwich Islands	NULL
United States Minor Outlying Islands	NULL

4.3. RIGHT JOIN (or RIGHT OUTER JOIN)

Returns **all rows from the right table**, and the matching rows from the left table. If there's no match in the left table for a row in the right table, NULL values will be returned for the columns from the left table.

Question: Give me a list of all cities, and the corresponding country's name and life expectancy where it is a capital city

```
2 •  SELECT
3        ci.Name AS CityName,
4        c.Name AS CountryName,
5        c.LifeExpectancy
6    FROM
7        country AS c
8    RIGHT JOIN
9        city AS ci ON c.Capital = ci.ID;
10
```

Result Grid | Filter Rows: | Export: | Wrap Cell C

CityName	CountryName	LifeExpectancy
Kabul	Afghanistan	45.9
Qandahar	NULL	NULL
Herat	NULL	NULL
Mazar-e-Sharif	NULL	NULL
Amsterdam	Netherlands	78.3
Rotterdam	NULL	NULL
Haag	NULL	NULL
Utrecht	NULL	NULL
Eindhoven	NULL	NULL
Tilburg	NULL	NULL

These values do not match on the Capital ID column; and the JOIN returns a NULL value for the right table that do not have a match in the LEFT table

4.4. SELF JOIN

A regular JOIN where a table is joined with **itself**. This is used to compare or combine rows *within the same table*. You **must** use aliases to distinguish between the two instances of the table.

Question: In Netherlands (country code = 'NLD'), show me the cities that share the same district.

```sql
SELECT
    c1.Name AS City_One,
    c2.Name AS City_Two,
    c1.District AS 'District'
FROM
    city c1
INNER JOIN
    city c2 ON c1.District = c2.District
    WHERE c1.CountryCode ='NLD'
    AND c1.ID > c2.ID
    ORDER BY 3;
```

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

City_One	City_Two	District
Haag	Rotterdam	Zuid-Holland
Dordrecht	Rotterdam	Zuid-Holland
Dordrecht	Haag	Zuid-Holland
Leiden	Rotterdam	Zuid-Holland
Leiden	Haag	Zuid-Holland
Leiden	Dordrecht	Zuid-Holland
Zoeterm...	Rotterdam	Zuid-Holland
Zoeterm...	Haag	Zuid-Holland
Zoeterm...	Dordrecht	Zuid-Holland
Zoeterm...	Leiden	Zuid-Holland
Delft	Rotterdam	Zuid-Holland

5. CROSS JOIN (Cartesian Product)

A CROSS JOIN produces the Cartesian product of the two tables involved. In simple terms, it means every row from the first table is combined with every single row from the second table.

Explanation:

- Imagine you have Table A with N rows and Table B with M rows.
- A CROSS JOIN between Table A and Table B will result in a new table with N * M (N multiplied by M) rows.
- You **do not need a common column** in a CROSS JOIN. You can perfectly CROSS JOIN two totally unrelated tables.
- There is no ON clause for a CROSS JOIN because there's no matching condition; it just combines all possibilities.

Example: Imagine you have two tables

Table 1	Table 2
Red	Small
Blue	Medium
Green	Large

```
SELECT c.color_name, s.size_label
FROM Colors
CROSS JOIN Sizes ;
```

This query doesn't need a common **color_id** in Sizes or a **size_id** in Colors. It will simply generate a cross join of 3*3 = 9 rows.

color_name	size_label
Red	Small
Red	Medium
Red	Large
Blue	Small
Blue	Medium
Blue	Large
Green	Small
Green	Medium
Green	Large

6. Set Operations

6.1. UNION

Combines the result sets of two or more SELECT statements. **It removes duplicate rows** from the result.

The SELECT statements must have the same number of columns, and corresponding columns must have compatible data types.

Explanation:

- The first SELECT Name FROM country gets all country names.
- The second SELECT Name FROM city gets all city names.
- UNION combines these two lists and then removes any names that appear in both (e.g., if a country and a city happened to have the exact same name, it would only appear once).

Result: A single, unique list of names from both tables.

6.2. UNION ALL

Combines the result sets of two or more SELECT statements. **It includes all rows, even if they are duplicates.**

The SELECT statement on either side must have compatible number of columns and data types.