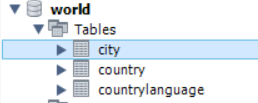
**Daily Homeworks**

**(26/05/2021)**

**I> Thực thi các câu lệnh cơ bản**

***\*Chạy database, ta thu được kết quả là các bảng sau:***

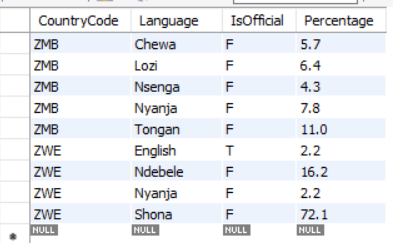


***\*Thêm vào bảng city 1 giá trị:***

* Thực thi câu lệnh sau:



Với ZWE là khóa ngoại là CountryCode = ZWE từ bảng CountryLanguage:



* Kết quả:



Sử dụng câu lệnh sau để hiển thị giá trị vừa thêm vào:



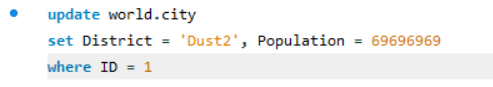


***\* Sửa 1 giá trị trong bảng city:***

* Giá trị ban đầu:



* Thực thi câu lệnh:



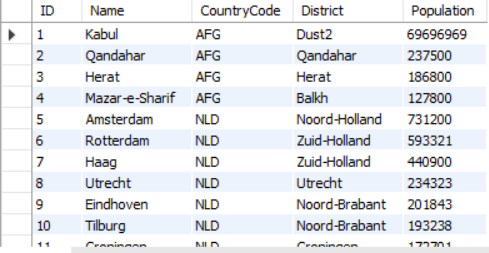
* Kết quả:





***\* Xóa 1 giá trị trong bảng city***

* Bảng ban đầu:



* Tắt Safe mode để tránh lỗi 1175

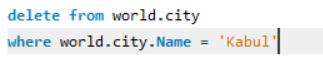


Sử dụng câu lệnh sau:



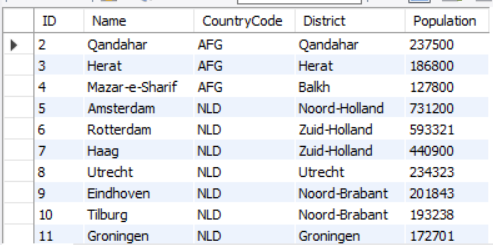


* Thực thi câu lệnh:



* Kết quả:





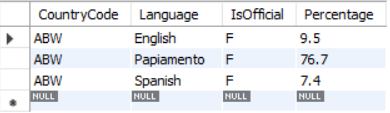
***\* Nhóm câu lệnh OPERATOR:***

- Sử dụng **AND**:

* Thực thi câu lệnh



* Kết quả

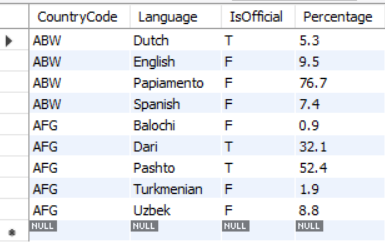


- Sử dụng **OR**

* Thực thi câu lệnh



* Kết quả

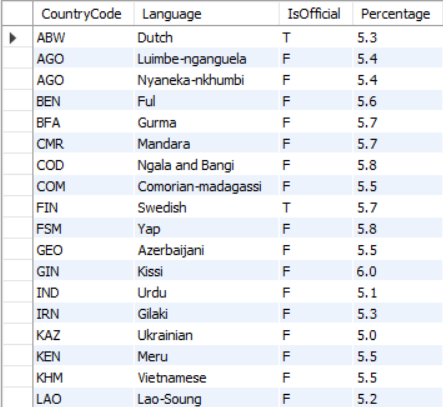


- Sử dụng **BETWEEN**

* Thực thi câu lệnh



* Kết quả

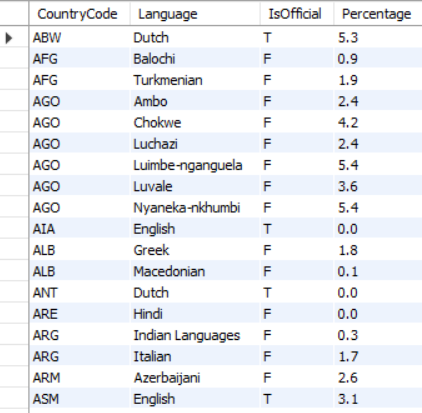


- Sử dụng **NOT BETWEEN**

* Thực thi câu lệnh



* Kết quả

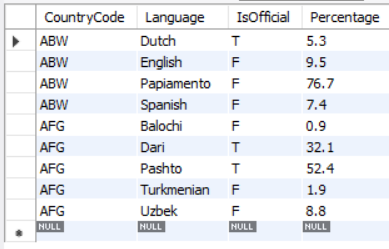


- Sử dụng **IN**:

* Thực thi câu lệnh



* Kết quả



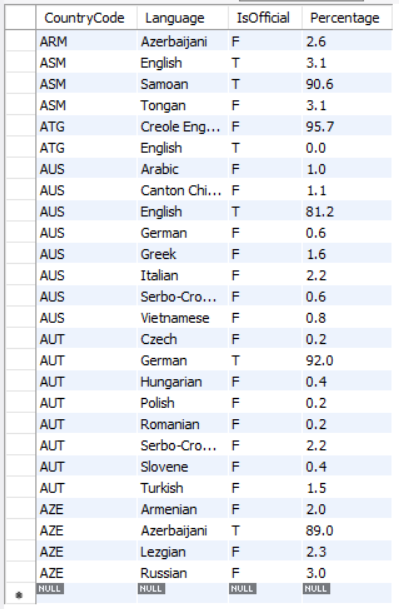
- Sử dụng **LIKE**

+ **LIKE "S%":**

* Thực thi câu lệnh



* Kết quả

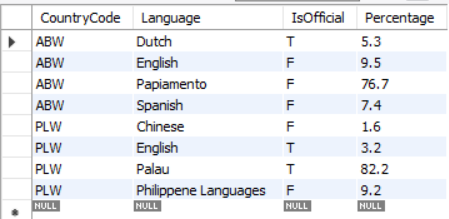


+ **LIKE "%S":**

* Thực thi câu lệnh



* Kết quả

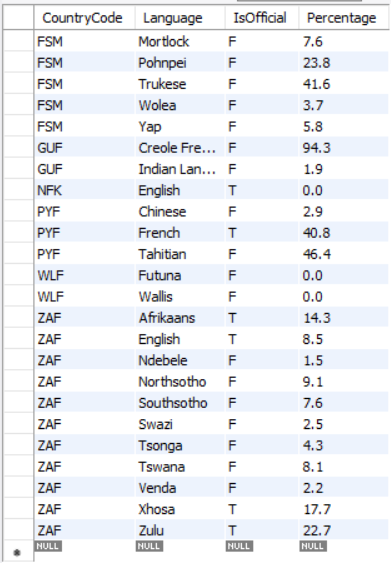


+ **LIKE "%S%":**

* Thực thi câu lệnh



* Kết quả



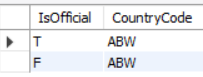
***\*Nhóm câu lệnh GROUP BY***

***-*** Normal **GROUP BY**

* Thực thi câu lệnh

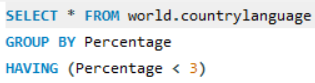


* Kết quả

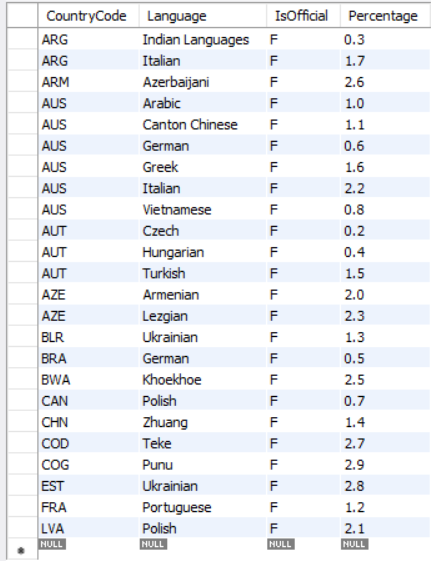


***-*** **GROUP BY** kết hợp điều kiện **(HAVING)**

* **Thực thi câu lệnh**

****

* **Kết quả**

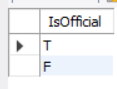
****

***\* Sử dụng DISTINCT***

* Thực thi câu lệnh



* Kết quả



**II> - Tìm hiểu câu lệnh Sub query , Union, Join, SORT**

***1, Câu lệnh Sub query***

- Sub querylà truy vấn trong một truy vấn SQL khác và được nhúng trong mệnh đề WHERE.

- Một sub query được sử dụng để trả về dữ liệu sẽ được sử dụng trong truy vấn chính như một điều kiện để hạn chế hơn nữa dữ liệu được truy xuất.

- Sub query có thể được sử dụng với câu lệnh SELECT, INSERT, UPDATE, và DELETE cùng với các toán tử như: =, <,>,> =, <=, IN, BETWEEN, v.v ...

Có một vài quy tắc mà Sub query phải tuân theo:

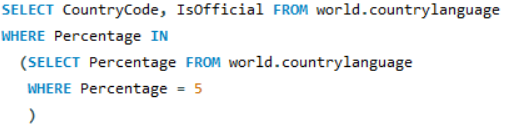
* Sub query phải được đặt trong dấu ngoặc đơn.
* Một sub query có thể chỉ có một cột trong mệnh đề SELECT, trừ khi nhiều cột trong truy vấn chính cho sub query để so sánh các cột đã chọn của nó.
* Không thể sử dụng lệnh ORDER BY trong sub query, mặc dù truy vấn chính có thể sử dụng ORDER BY. Lệnh GROUP BY có thể được sử dụng để thực hiện chức năng giống như ORDER BY trong một sub query.
* Sub query trả về nhiều hơn một hàng chỉ có thể được sử dụng với toán tử nhiều giá trị như toán tử IN.
* Danh sách SELECT không được bao gồm bất kỳ tham chiếu nào đến các giá trị đánh giá BLOB, ARRAY, CLOB hoặc NCLOB.
* Một sub query không thể được chứa trực tiếp một chức năng set.
* Toán tử BETWEEN không thể được sử dụng với một sub query. Tuy nhiên, toán tử BETWEEN có thể được sử dụng trong sub query.

- Cấu trúc của câu lệnh sử dụng Sub query

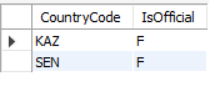
|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | |  | SELECT column\_name  FROM   table1  WHERE  column\_name OPERATOR     (SELECT column\_name     FROM table1     [WHERE]) | |

- Thực hành

* Thực thi câu lệnh



* Kết quả



***2, Toán tử UNION***

- Toán tử UNION cho phép chúng ta nối kết quả của hai hoặc nhiều câu truy vấn lại với nhau để trở thành một danh sách kết quả duy nhất.

- Khi sử dụng toán tử UNION chúng ta cần phải tuân thủ những nguyên tắc sau đây:

* Số lượng colums trong tất cả các lệnh SELECT phải bằng nhau
* Mỗi column tương ứng vị trí phải có cùng kiểu dữ liệu và độ dài

- Theo mặc định thì UNION sẽ loại bỏ các kết quả trùng lặp của các câu SELECT nên nó tạo cho chúng ta hai lựa chọn sau:

* Nếu chọn UNION DISTINCT thì nó sẽ loại bỏ kết quả trùng.
* Nếu chọn UNION ALL thì nó giữ lại kết quả trùng.
* Nếu bạn không chọn gì thì mặc định nó sẽ lấy UNION DISTINCT

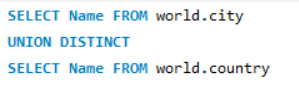
- Cấu trúc của câu lệnh sử dụng toán tử UNION

|  |
| --- |
| SELECT column1, column2 FROM table 1  UNION [DISTINCT | ALL]  SELECT column1, column2 FROM table 2 |

- Thực hành

+ Sử dụng UNION DISTINCT

* Thực thi câu lệnh

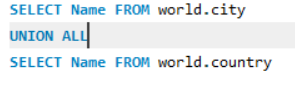


* Kết quả



+ Sử dụng UNION ALL

* Thực thi câu lệnh



* Kết quả



***3, Mệnh đề JOIN***

- Mệnh đề **JOIN** trong SQL được sử dụng để kết hợp các bản ghi từ hai hay nhiều bảng trong cơ sở dữ liệu. JOIN là một phương tiện để kết hợp các trường từ hai bảng bằng cách sử dụng các giá trị chung cho mỗi bảng.

- Nó được thực hiện trong mệnh đề WHERE. Một số toán tử có thể được sử dụng để kết nối các bảng, chẳng hạn như =, < ,>, <>, <=,> =, ! =, BETWEEN, LIKE, và NOT. Tất cả chúng có thể được sử dụng để kết nối các bảng.

- Các kiểu Join trong SQL

**a, INNER JOIN**

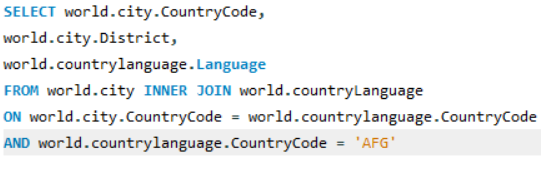
- Lệnh **Inner Join trong SQL**tạo một bảng kết quả mới bằng cách kết hợp các giá trị cột của hai bảng (table1 và table2) dựa trên điều kiện nối. Truy vấn so sánh từng hàng của table1 với từng hàng của table2 để tìm tất cả các cặp hàng thỏa mãn điều kiện nối. Khi điều kiện nối được thỏa mãn, các giá trị cột cho mỗi cặp hàng A và B phù hợp sẽ được kết hợp thành một hàng kết quả.

- Cấu trúc của mệnh đề:

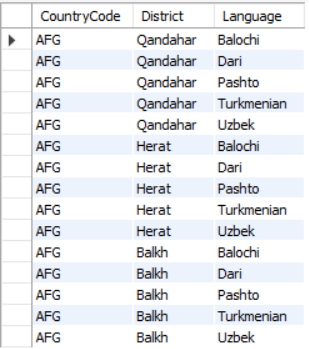
|  |
| --- |
| SELECT table1.column1, table2.column2...  FROM table1  INNER JOIN table2  ON table1.common\_field = table2.common\_field; |

- Thực hành

* Thực thi câu lệnh



* Kết quả



**b, LEFT JOIN**

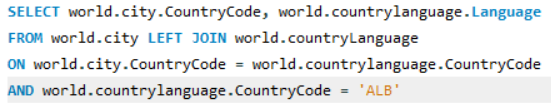
- Lệnh **Left Join trong SQL** trả về tất cả các hàng từ bảng bên trái, ngay cả khi không có kết quả khớp nào trong bảng bên phải. Điều này có nghĩa là nếu mệnh đề ON không khớp với các bản ghi của bảng bên phải, lệnh Left Join vẫn sẽ trả về một hàng trong kết quả, nhưng với NULL cho mỗi cột của bảng bên phải.

- Cấu trúc của mệnh đề:

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | |  | SELECT table1.column1, table2.column2...  FROM table1  LEFT JOIN table2  ON table1.common\_field = table2.common\_field; | |

- Thực hành

* Thực thi câu lệnh



* Kết quả



**c, RIGHT JOIN**

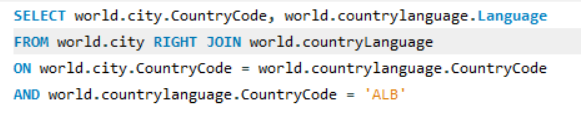
- Ngược lại so với LEFT JOIN, lệnh **Right Join** trả về tất cả các hàng từ bảng bên phải, ngay cả khi không có kết quả khớp nào trong bảng bên trái. Điều này có nghĩa là nếu mệnh đề ON không khớp với các bản ghi của bảng bên trái. Lệnh Left Join vẫn sẽ trả về một hàng trong kết quả, nhưng với NULL cho mỗi cột của bảng bên trái.

- Cấu trúc của mệnh đề:

|  |
| --- |
| SELECT table1.column1, table2.column2...  FROM table1  RIGHT JOIN table2  ON table1.common\_field = table2.common\_field;; |

- Thực hành

* Thực thi câu lệnh



* Kết quả



**d, FULL JOIN**

- Lệnh **Full Join** là sự kết hợp các kết quả của cả hai phép nối Left Join và Right Join.

- Kết quả join sẽ chứa tất cả các bản ghi từ cả hai bảng và điền vào NULL cho các kết quả bị thiếu ở hai bên.

- Cấu trúc của mệnh đề:

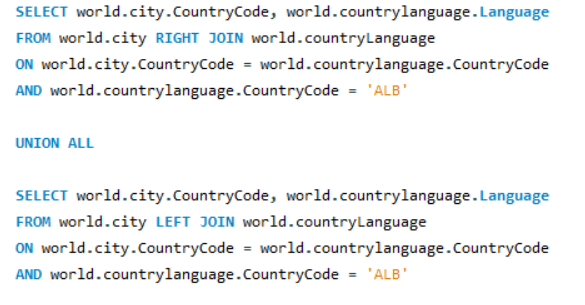
|  |
| --- |
| SELECT table1.column1, table2.column2...  FROM table1  FULL JOIN table2  ON table1.common\_field = table2.common\_field; |

- Thực hành

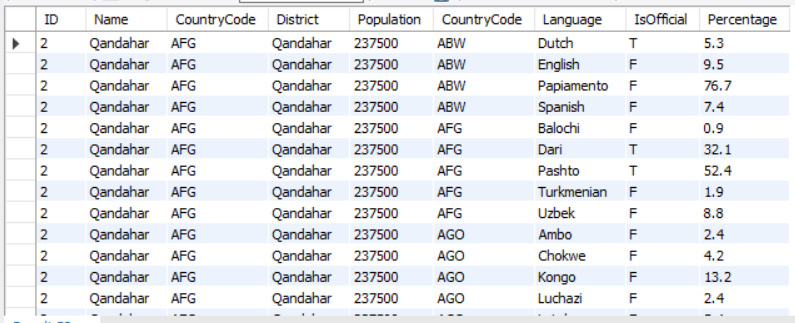
* Thực thi câu lệnh



Nếu MySQL của máy không hỗ trợ FULL JOIN chúng ta sẽ sử dụng UNION ALL để kết hợp 2 lệnh LEFT UNION và RIGHT UNION



* Kết quả



Sử dụng UNION ALL cho LEFT JOIN và RIGHT JOIN



**e, SELF JOIN**

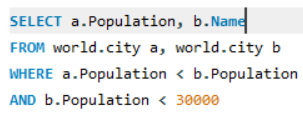
- Lệnh Self Join trong SQL được sử dụng để nối một bảng với chính nó.

- Cấu trúc mệnh đề:

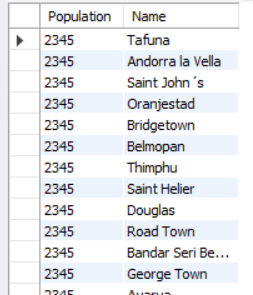
|  |
| --- |
| SELECT a.column\_name, b.column\_name...  FROM table1 a, table1 b  WHERE a.common\_field = b.common\_field; |

- Thực hành

* Thực thi câu lệnh



* Kết quả



***4, SORT- ORDER BY***

- Mệnh đề ORDER BY trong SQL được sử dụng để sắp xếp dữ liệu theo thứ tự tăng dần hoặc giảm dần, dựa trên một hoặc nhiều cột.

- Một số cơ sở dữ liệu sắp xếp các kết quả truy vấn theo thứ tự tăng dần theo mặc định, tuy nhiên, ta có thể sử dụng:

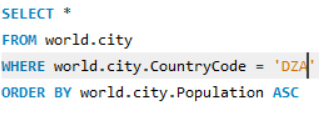
* Lệnh ASC được sử dụng để sắp xếp tăng dần
* Lệnh DESC được sử dụng để sắp xếp giảm dần.

- Cấu trúc mệnh đề

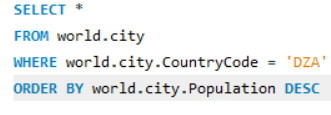
|  |
| --- |
| SELECT column-list  FROM table\_name  [WHERE condition]  [ORDER BY column1, column2, .. columnN] [ASC | DESC]; |

- Thực hành:

* Thực thi câu lệnh:

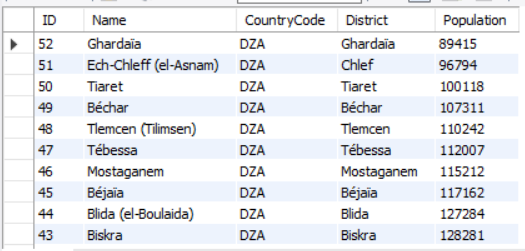


Hoặc



* Kết quả:

Sử dụng ASC (tăng dần)



Sử dụng DESC (giảm dần)

