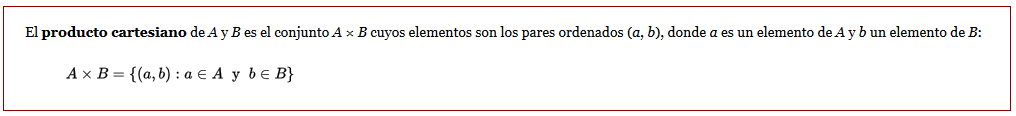
**Producto Cartesiano**



- En matemáticas, el **producto cartesiano** es una operación entre dos conjuntos.

- El resultado de esta operación son todos los [pares ordenados](https://es.wikipedia.org/wiki/Par_ordenado) que pueden formarse.

Es decir, el primer elemento del par ordenado pertenece al conjunto 1 y el segundo elemento pertenece al conjunto 2.

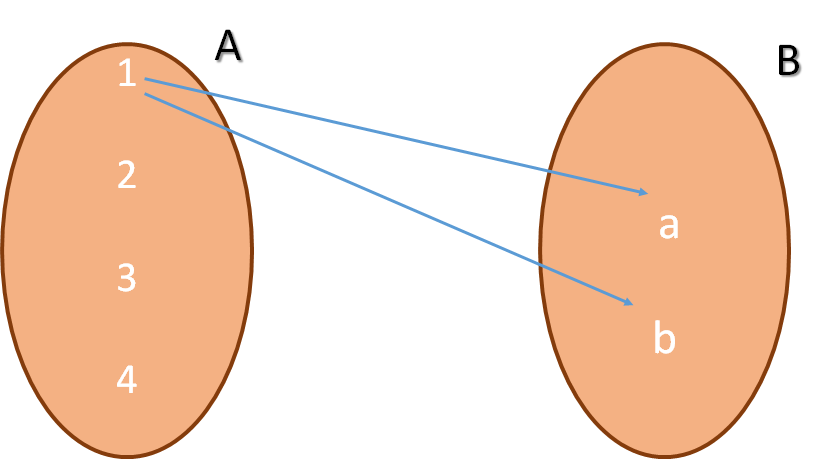
- Los pares ordenados también se denominan tuplas.

- Se llama así en honor a  [René Descartes](https://es.wikipedia.org/wiki/Ren%C3%A9_Descartes), cuya formulación de la [geometría analítica](https://es.wikipedia.org/wiki/Geometr%C3%ADa_anal%C3%ADtica) dio origen a este concepto.

- Ejemplo



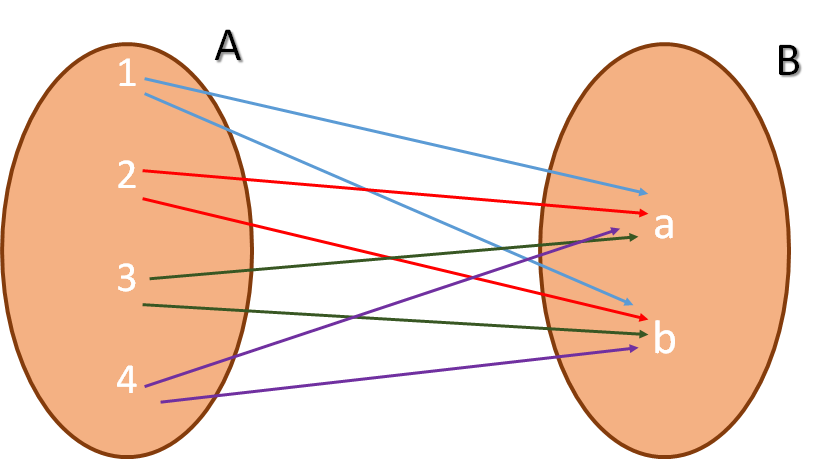




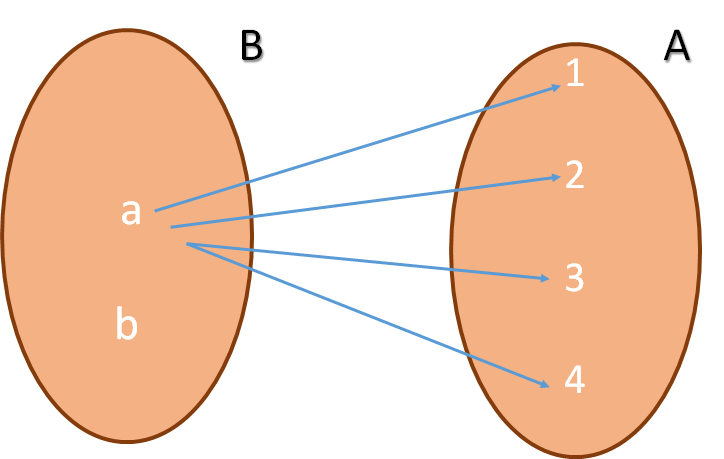
El producto cartesiano sería cada elemento de A unido con cada elemento de B

Producto cartesiano de los conjuntos anteriores:

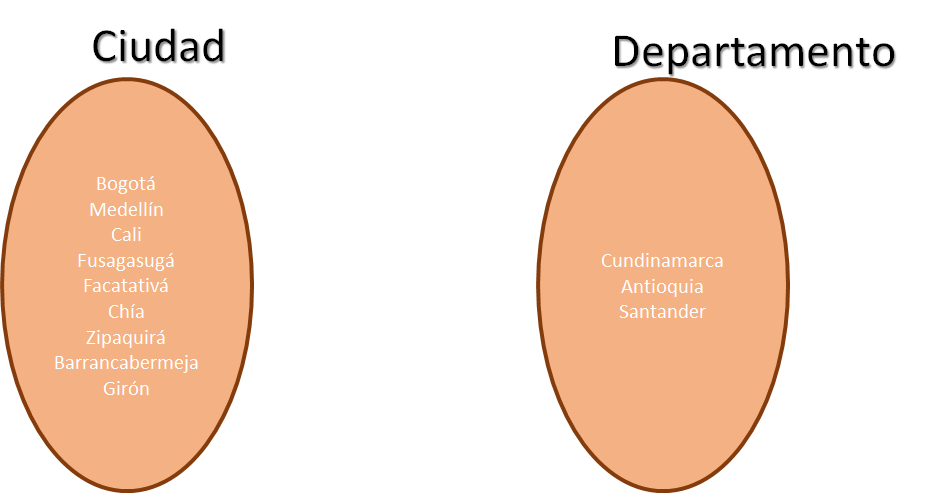


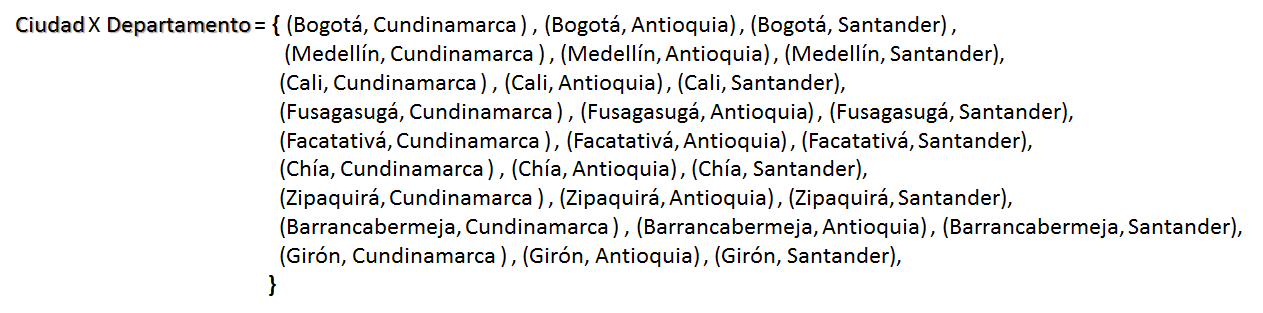


Tenga en cuenta que :





****

****

**Obviamente, el producto cartesiano no se ajusta para saber a cuál departamento pertenece una ciudad, entonces ¿Cómo relacionar correctamente?**

****

Ejemplo:

1. Crear la base de datos con las tablas *Ciudad* y *Departamento*:

[*Ejemplo\_Producto cartesiano*](https://drive.google.com/drive/folders/14vfMYMwhd02dSsGoF0h6isXXFCNR6iWT?usp=drive_link)

1. Alimentar la base de datos ([*Ejemplo Importar (CSV a MySQL)*](https://drive.google.com/drive/folders/1O4Tah8Q3gzHGAhiVY0BQC7imEyvWkn3h?usp=drive_link))

**INSERT INTO Departamento (nombreDepartamento)**

**VALUES**

**("Amazonas"),**

**("Antioquia"),**

**("Arauca"),**

**("Atlántico"),**

**("Bolívar"),**

**("Boyacá"),**

**("Caldas"),**

**("Caquetá"),**

**("Casanare");**

**INSERT INTO Ciudad (nombreCiudad,idDepartamento)**

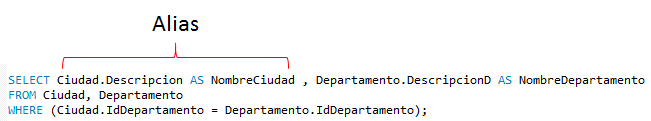
**VALUES**

**("Bogotá D.C",15),**

**("Medellín",2);**

**("Cali",31);**

1. Ahora vamos a imprimir los datos:



¿Qué resultado se obtuvo?

Efectivamente, acabamos de realizar una operación entre dos conjuntos, el *producto cartesiano.*

1. ¿Cómo obtener los registros que sean acordes a la realidad?

Para obtener los resultados esperados, vamos a realizar una operación de *intersección* de conjuntos.

Recordemos que la intersección de conjuntos es una operación que resulta en un nuevo conjunto formado por los elementos comunes a dos o más conjuntos originales.

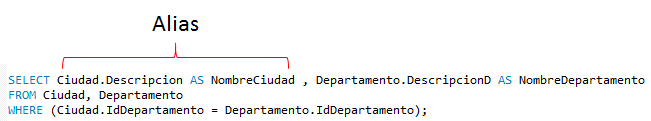
Tengamos en cuenta que, entre el conjunto llamado ***Ciudad*** y el conjunto ***Departamento***, el elemento común es el *IdDepartamento*.

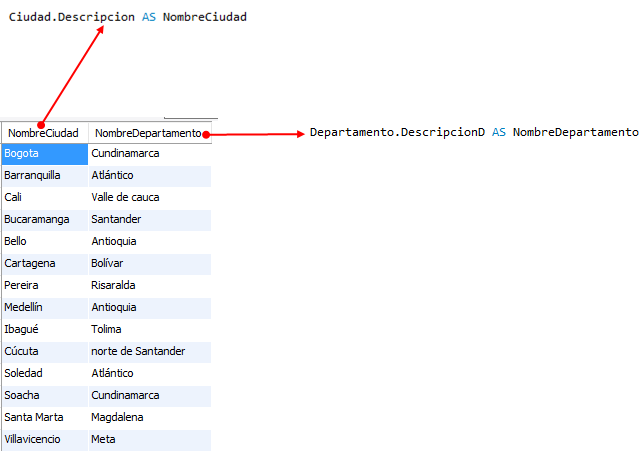
Siendo así, podemos filtrar de la siguiente manera:

**SELECT** Ciudad.Descripcion AS NombreCiudad , Departamento.DescripcionD AS NombreDepartamento

**FROM** Ciudad, Departamento

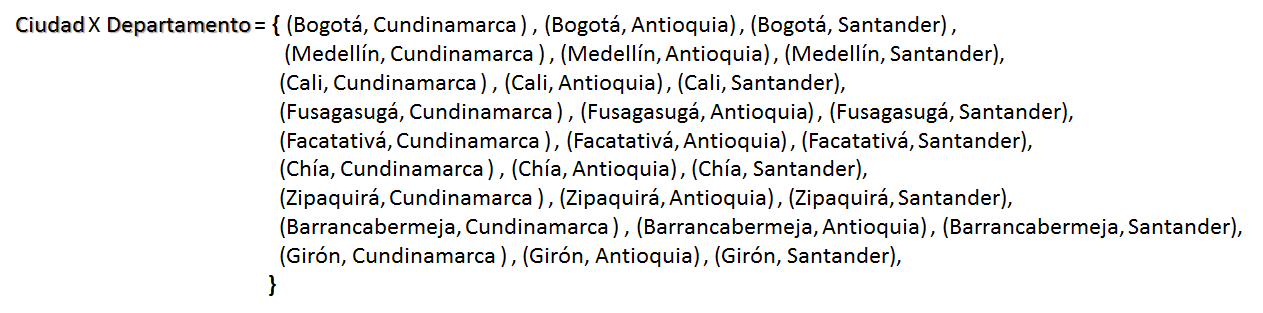
**WHERE** (Ciudad.IdDepartamento = Departamento.IdDepartamento);





**Retomemos:**

Como sabemos, **AxB** (producto cartesiano) nos genera el siguiente conjunto:

****

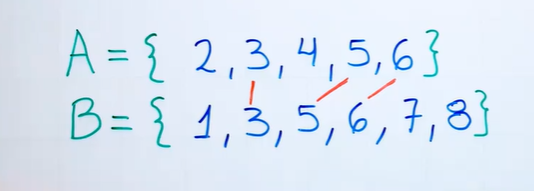
El producto cartesiano en sql sería:

**Select** Ciudad.nombreCiudad, Departamento.nombreDepartamento

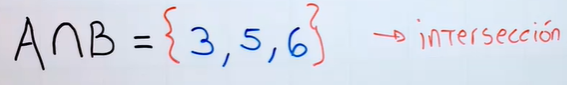
**From** Ciudad, Departamento;

Pero debemos notar que las tuplas resultantes no son las correspondencias de la realidad, para ello realizaremos la intersección de los conjuntos.

Recordemos: Hallar la intersección de dos conjuntos A y B



Solución:



¿Qué tienen en común la ***tabla Ciudad*** y la ***tabla Departamento***?



R/ Evidentemente, el campo llamado *idDepartamento*. Por tanto, en sql la intersección de las dos tablas sería:

| **where** ( Ciudad.*idDepartamento* = Departamento**.***idDepartamento* ) |
| --- |

Finalmente, la consulta (query) quedaría así:

**Select** Ciudad.nombreCiudad **as** “Ciudad”, Departamento**.**nombreDepartamento **as** “Nombre Dpto”

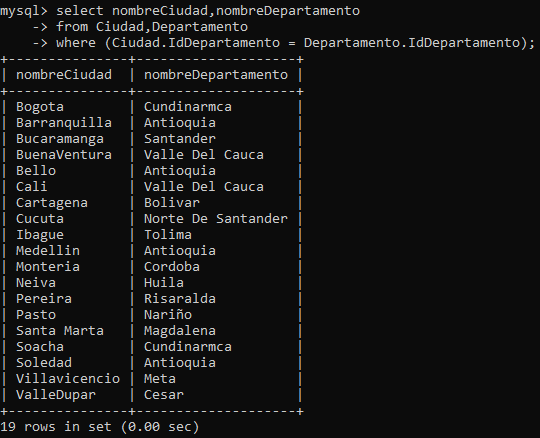
**From** Ciudad, Departamento

**where (**ciudad.idDepartamento = Departamento**.**idDepartamento**)**;

**SELECT** Ciudad.Descripcion AS NombreCiudad , Departamento.Descripcion AS NombreDepartamento

**FROM** Ciudad, Departamento

**WHERE** (Ciudad.IdDepartamento = Departamento.IdDepartamento);



*\* Verificar datos del archivo original*

Ahora veamos:

1. Unión de tablas
2. Composición de tablas.

Los **tipos de composición** de tablas son:

* El **producto cartesiano**
* El **INNER JOIN**
* El **LEFT / RIGHT JOIN**

**UNIÓN DE TABLAS:**

- Ver: <https://www.w3schools.com/SQL/sql_union.asp>

- El operador **UNION** se utiliza para combinar el conjunto de resultados de dos o más instrucciones **SELECT**.

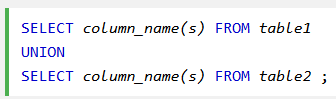
- Esta operación se utiliza cuando tenemos dos tablas con las mismas columnas y queremos obtener una nueva tabla con las filas de la Tabla 1 y las filas de la Tabla 2.

- La tabla resultante tiene las mismas columnas que la primera tabla (que son las mismas que las de la segunda tabla).

Pero sin incluir los valores repetidos. Si queremos incluir los valores duplicados, usaremos **UNION ALL**.

- Importante:

* Cada sentencia SELECT dentro de UNION debe tener el mismo número de columnas
* Las columnas también deben tener tipos de datos similares
* Las columnas de cada instrucción SELECT también deben estar en el mismo orden

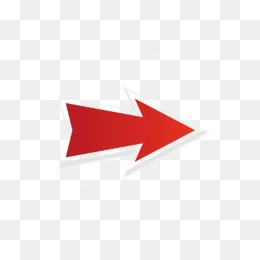


- Ejemplo:

Tenemos una tabla de **libros nuevos** y una tabla de **libros antiguos.**

| **CREATE DATABASE** EjemploUnion;  **USE** EjemploUnion;  **CREATE TABLE** LibroAntiguo  (  IdLibroAntiguo int(3)NOT NUll auto\_increment,  DescripcionLibroAntiguo varchar(50) not null,  PRIMARY KEY (IdLibroAntiguo )  );  **CREATE TABLE** LibroNuevo  (  IdLibroNuevo int(3)NOT NUll auto\_increment,  DescripcionLibroNuevo varchar(50) not null,  PRIMARY KEY (IdLibroNuevo)  );  **INSERT INTO** LibroAntiguo (DescripcionLibroAntiguo) **VALUES**  ('Crimen y castigo'),  ('100 años de Soledad'),  ('Cumbres borrascosas'),  ('Las almas muertas'),  ('La comedia humana'),  ('Pedro Páramo'),  ('Las metamorfosis'),  ('El tambor de hojalata'),  ('Lo que el viento se llevó');    **INSERT INTO** LibroNuevo (DescripcionLibroNuevo) **VALUES**  ('El monarca de las sombras'),  ('Lo que te diré cuando te vuelva a ver'),  ('El laberinto de los espíritus');    **SELECT** \* FROM LibroAntiguo;  **SELECT** \* FROM LibroNuevo;  -- Union:  **SELECT** \* FROM LibroAntiguo  **UNION**  **SELECT** \* FROM LibroNuevo**;** |
| --- |





**SELECT** \* FROM LibroAntiguo

**UNION**

**SELECT** \* FROM LibroNuevo;

En este caso las dos tablas tienen las mismas columnas,

lo único que varía son las filas.



* **Tablas Reales (Fìsico)**:

Almacenadas en la base de datos (Create table).

* **Tablas Lógicas:**

Resultados de una consulta (query).

El resultado de un query **es siempre** una tabla lógica.

Ahora insertemos un libro repetido:



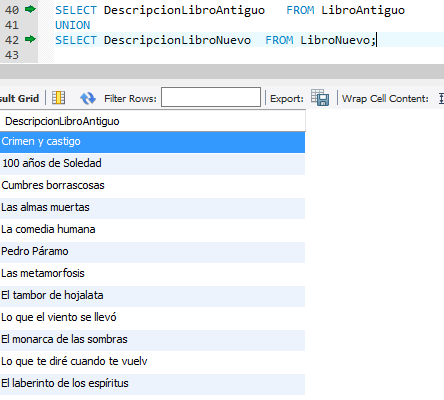
INSERT INTO LibroNuevo (DescripcionLibroNuevo) VALUES('El laberinto de los espíritus');

1. **UNION (No incluye repetidos)**

**SELECT** DescripcionLibroAntiguo FROM LibroAntiguo

**UNION**

**SELECT** DescripcionLibroNuevo FROM LibroNuevo;



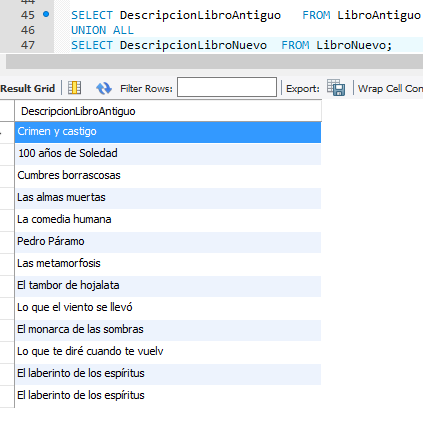
1. **UNION ALL (Sí incluye repetidos)**

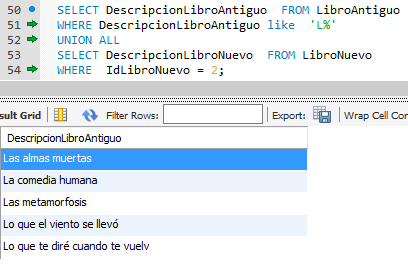
Nos permite utilizar la operación con más frecuencia ya que pocas veces tenemos en una base de datos tablas idénticas en cuanto a columnas.

**SELECT** DescripcionLibroAntiguo FROM LibroAntiguo

**UNION ALL**

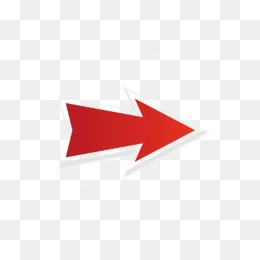
**SELECT** DescripcionLibroNuevo FROM LibroNuevo;



Otro ejemplo:

**SELECT** DescripcionLibroAntiguo

**FROM** LibroAntiguo

**WHERE** DescripcionLibroAntiguo like 'L%'

**UNION ALL**

**SELECT** DescripcionLibroNuevo

**FROM** LibroNuevo

**WHERE**  IdLibroNuevo = 2**;**

**Join**

- JOIN (unir, combinar)

- Es una sentencia de [SQL](https://es.wikipedia.org/wiki/SQL)

- Permite combinar [registros](https://es.wikipedia.org/wiki/Registro_%28base_de_datos%29) de una o más [tablas](https://es.wikipedia.org/wiki/Tabla_%28base_de_datos%29) en una [base de datos relacional](https://es.wikipedia.org/wiki/Base_de_datos_relacional).

- En el Lenguaje de Consultas Estructurado ([SQL](https://es.wikipedia.org/wiki/SQL)) hay 3 (tres) tipos de JOIN: ***interno***, ***externo*** y ***cruzado***.

- El estándar **ANSI del SQL** específica 5 (cinco) tipos de JOIN:

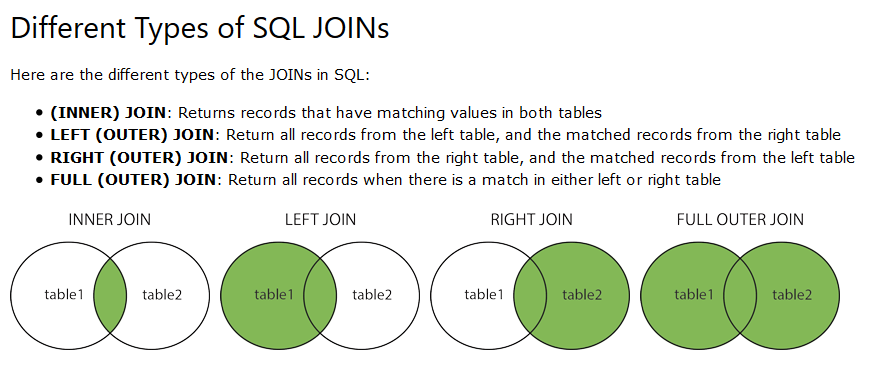
1. **INNER**
2. **LEFT OUTER**
3. **RIGHT OUTER**
4. **FULL OUTER**
5. **CROSS**

- Una tabla puede unirse a sí misma, produciendo una auto-combinación, **SELF-JOIN**.

- Matemáticamente, **JOIN** es composición relacional, la operación fundamental en el [álgebra relacional](https://es.wikipedia.org/wiki/%C3%81lgebra_relacional), y, generalizando, es una función de [composición](https://es.wikipedia.org/wiki/Operaci%C3%B3n_matem%C3%A1tica).

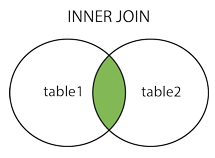
- Ver:

<https://www.w3schools.com/sql/sql_join.asp>



1. **INNER JOIN**

- Intersección de conjuntos

****

- Intersección de conjuntos

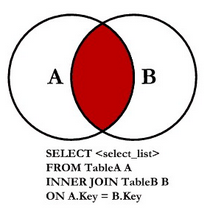




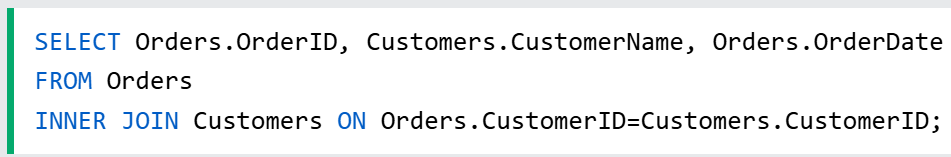


La intersección de dichos conjuntos estará formada por todos los elementos que estén a la vez en los dos conjuntos.

* Importante: 
* Ejemplo:



Ejemplo:

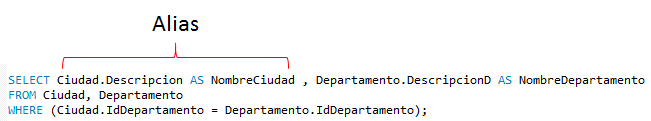


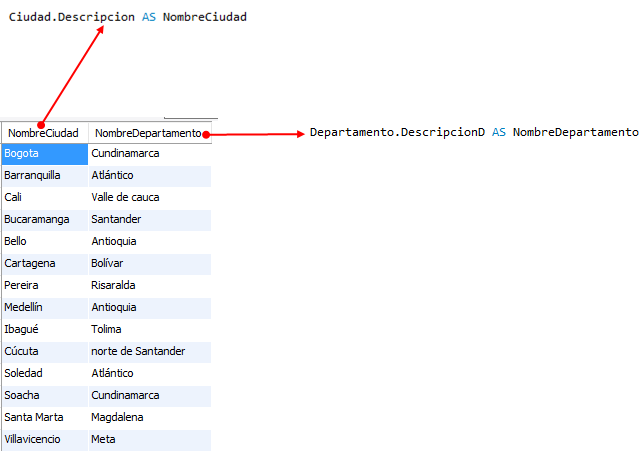
Por favor, convierta la siguiente instrucción en **INNER JOIN**:

**SELECT** Ciudad.Descripcion AS NombreCiudad , Departamento.DescripcionD AS NombreDepartamento

**FROM** Ciudad, Departamento

**WHERE** (Ciudad.IdDepartamento = Departamento.IdDepartamento);





Solución:

