Ejercicios de Teoría de Conjuntos

- Utilizando los siguientes conjuntos

```
All = {1,2,3,4,5,6,7,8,9,10}
Even = {2,4,6,8,10}
Odd = {1,3,5,7,9}
```

Realice las siguientes operaciones:

1. `Even U Odd`
2. `Even ∩ Odd`
3. `All - Odd`
4. `C(Even)`
5. `C(Odd-All)`

1. {1,2,3,4,5,6,7,8,9,10}
2. {}
3. {2,4,6,8,10}
4. {1,3,5,7,9}
5. {1,2,3,4,5,6,7,8,9,10}

Ejercicios de `JOINs`

1. Investigue y busque documentación sobre el `SELECT` que explique cómo utilizar las funcionalidades de `ORDER BY`, `LIMIT`, `GROUP BY` y tres tipos de `JOIN`, y cree una tabla (informativa, no de SQL) a partir de sus descubrimientos.

El SELECT no solo sirve para traer columnas, sino que depende mucho de cómo se "limpie" la información antes de mostrarla.

1. Organización y Restricción

ORDER BY: Es lo que usamos para que la lista no salga aleatoria. Si pones ASC va de menor a mayor y con DESC al revés. Se pueden poner varias columnas (por ejemplo, ordenar por apellido y luego por nombre).

LIMIT: Super útil para no colapsar la pantalla. Si la tabla tiene 10,000 filas pero solo quieres ver las primeras 10, le pones LIMIT 10 al final de todo.

GROUP BY: Este es un poco más complejo. Sirve para "agrupar" filas que repiten un valor. Por ejemplo, si tienes una lista de ventas, puedes agrupar por "Categoría" para saber cuánto se vendió en total por cada una usando un SUM.

2. Tipos de JOIN (Para cruzar tablas)

Para que esto funcione, las tablas tienen que tener una columna en común (como un ID).

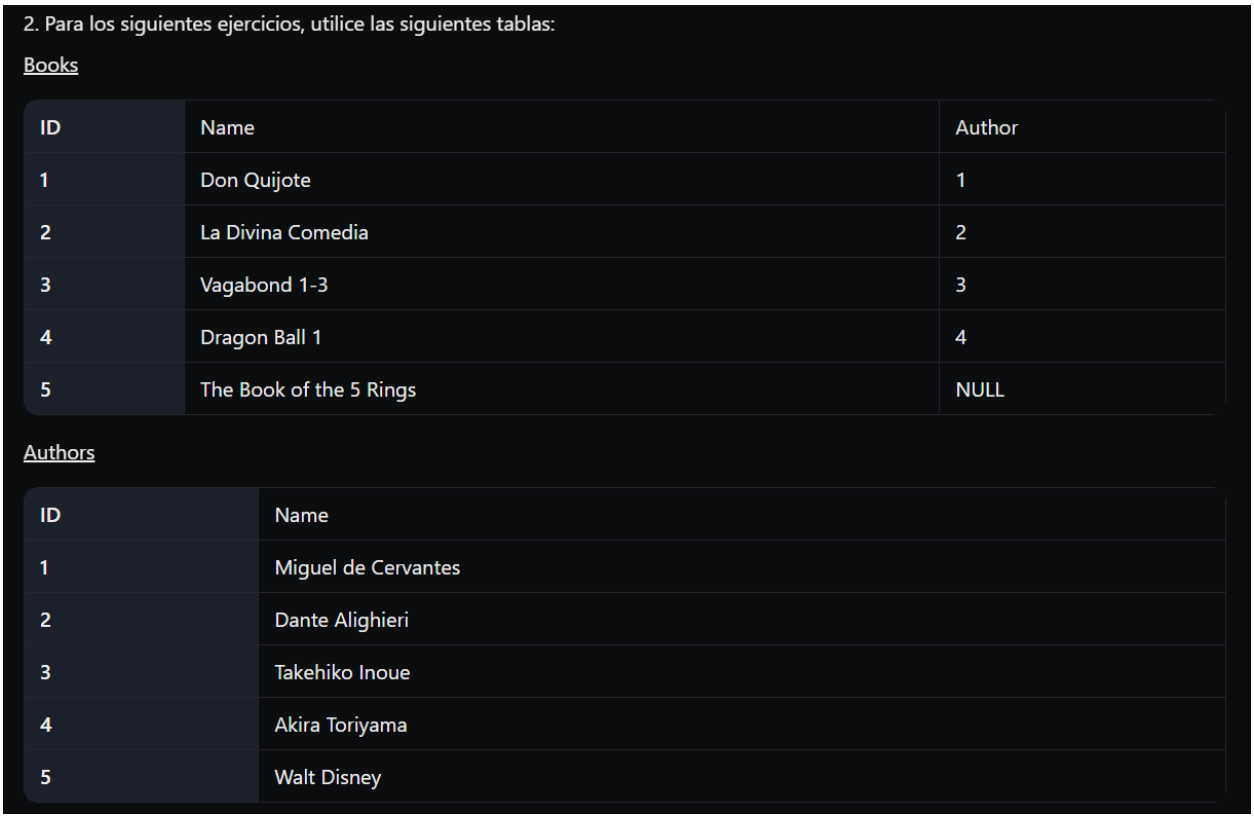
INNER JOIN: Es el más común. Solo te muestra los registros que tienen "pareja" en ambas tablas. Si un dato no coincide en la otra tabla, desaparece del resultado.

LEFT JOIN: Te da todo lo de la primera tabla (la de la izquierda), aunque no tenga coincidencia en la segunda. Lo que no encuentra lo rellena con NULL.

RIGHT JOIN: Es lo mismo que el anterior, pero al revés: manda la tabla que mencionas al final.

Comando	Para qué sirve	Nota importante
ORDER BY	Ordenar los resultados.	Por defecto es ascendente si no le pones nada.
LIMIT	Cortar el número de filas.	Siempre va al puro final del código.
GROUP BY	Agrupar por categorías.	Obliga a usar funciones como COUNT, SUM o AVG.
INNER JOIN	Cruce exacto de tablas.	Si no hay coincidencia en ambas, no sale nada.
LEFT JOIN	Priorizar tabla izquierda.	Ideal para ver qué registros están "huérfanos".
RIGHT JOIN	Priorizar tabla derecha.	Menos usado, pero útil si la tabla principal es la segunda.

Nota personal sobre el orden: Algo que encontré es que el SQL es muy estricto con el orden en que escribes las cosas. No puedes poner el LIMIT antes que el WHERE o el GROUP BY, porque te da error de sintaxis de inmediato.



2. Para los siguientes ejercicios, utilice las siguientes tablas:

Books

ID	Name	Author
1	Don Quijote	1
2	La Divina Comedia	2
3	Vagabond 1-3	3
4	Dragon Ball 1	4
5	The Book of the 5 Rings	NULL

Authors

ID	Name
1	Miguel de Cervantes
2	Dante Alighieri
3	Takehiko Inoue
4	Akira Toriyama
5	Walt Disney

Customers

ID	Name	Email
1	John Doe	j.doe@email.com
2	Jane Doe	jane@doe.com
3	Luke Skywalker	darth.son@email.com

Rents

ID	BookID	CustomerID	State
1	1	2	Returned
2	2	2	Returned
3	1	1	On time
4	3	1	On time
5	2	2	Overdue

- Cree las tablas en una base de datos SQL y realice las siguientes operaciones:
  - Debe entregar todos los queries realizados y capturas del resultado.

1. Obtenga todos los libros y sus autores
2. Obtenga todos los libros que no tienen autor
3. Obtenga todos los autores que no tienen libros
4. Obtenga todos los libros que han sido rentados en algún momento
5. Obtenga todos los libros que nunca han sido rentados
6. Obtenga todos los clientes que nunca han rentado un libro
7. Obtenga todos los libros que han sido rentados y están en estado “Overdue”

Books

SQL ▼ 1 / 1 1 - 5 of 5

Books_id	Name	Author
1	Don Quijote	1
2	La Divina Comedia	2
3	Vagabond 1-3	3
4	Dragon Ball 1	4
5	The Book of the 5 Rings	NULL

Authors

SQL ▼ 1 / 1 1 - 5

Author_id	Name
1	Miguel de Cervantes
2	Dante Alighieri
3	Takehiko Inoue
4	Akira Toriyama
5	Walt Disney

Customers

SQL ▼ 1 / 1 1 - 3 of 3

Customer_id	Name	Email
1	John Doe	j.doe@email.com
2	Jane Doe	jane@doe.com
3	Luke Skywalker	darth.son@email.com

Rents

SQL ▼ 1 / 1 1 - 5 of 5

Rent_id	Books_id	Customer_id	State
1	1	2	Returned
2	2	2	Returned
3	1	1	On time
4	3	1	On time
5	2	2	Overdue

1. Obtenga todos los libros y sus autores

```sql
SELECT
Books.name AS book_name,
Authors.name AS author_name
FROM Books
LEFT JOIN Authors
ON Books.author = Authors.author_id
WHERE Authors.author_id IS NULL;
```

SQL ▼ 1 / 1 1 - 5 of 5

Name	Name
Don Quijote	Miguel de Cervantes
La Divina Comedia	Dante Alighieri
Vagabond 1-3	Takehiko Inoue
Dragon Ball 1	Akira Toriyama
The Book of the 5 Rings	Walt Disney

(omitamos lo del nombre, sé que hay que diferenciarlos, pero para efectos prácticos el resultado funciona)

2. Obtenga todos los libros que no tienen autor

```sql
SELECT
    Books.name AS book_name,
    Authors.name AS author_name
FROM Books
LEFT JOIN Authors
    ON Books.author = Authors.author_id
WHERE Authors.author_id IS NULL;
```

SQL ▼ 1 / 1 1 - 1 of 1

book_name	author_name
The Book of the 5 Rings	NULL

3. Obtenga todos los autores que no tienen libros

```
SELECT
Authors.name AS author_name,
Books.name AS books_name
FROM Authors
LEFT JOIN Books
ON Authors.author_id = Books.author
WHERE Books.books_id IS NULL;
```

SQL ▼ 1 / 1

author_name	books_name
Walt Disney	NULL

4. Obtenga todos los libros que han sido rentados en algún momento

```
SELECT DISTINCT
Books.Books_id,
Books.Name
FROM Books
INNER JOIN Rents
ON Books.Books_id = Rents.Books_id;
```

SQL ▼ < 1 / 1 > 1

Books_id	Name
1	Don Quijote
2	La Divina Comedia
3	Vagabond 1-3

5. Obtenga todos los libros que nunca han sido rentados

```
SELECT
Books.Books_id,
Books.Name
FROM Books
LEFT JOIN Rents
ON Books.Books_id = Rents.Books_id
WHERE Rents.Books_id IS NULL;
```

SQL ▼ < 1 / 1 > 1 - 2 of

Books_id	Name
4	Dragon Ball 1
5	The Book of the 5 Rings

6. Obtenga todos los clientes que nunca han rentado un libro

```
SELECT
Customers.Customer_id,
Customers.Name,
Customers.Email
FROM Customers
LEFT JOIN Rents
ON Customers.Customer_id = Rents.Customer_id
WHERE Rents.Customer_id IS NULL;
```

SQL ▼ 1 / 1 1 - 1 of 1

Customer_id	Name	Email
3	Luke Skywalker	darth.son@email.com

7. Obtenga todos los libros que han sido rentados y están en estado “Overdue”

```
SELECT DISTINCT
Books.Books_id,
Books.Name
FROM Books
INNER JOIN Rents
ON Books.Books_id = Rents.Books_id
WHERE Rents.State = 'Overdue';
```

SQL ▼ 1 / 1 1

Books_id	Name
2	La Divina Comedia