



Clase 04. SQL

SUBLENGUAJES SQL

***RECUERDA PONER A GRABAR LA
CLASE***





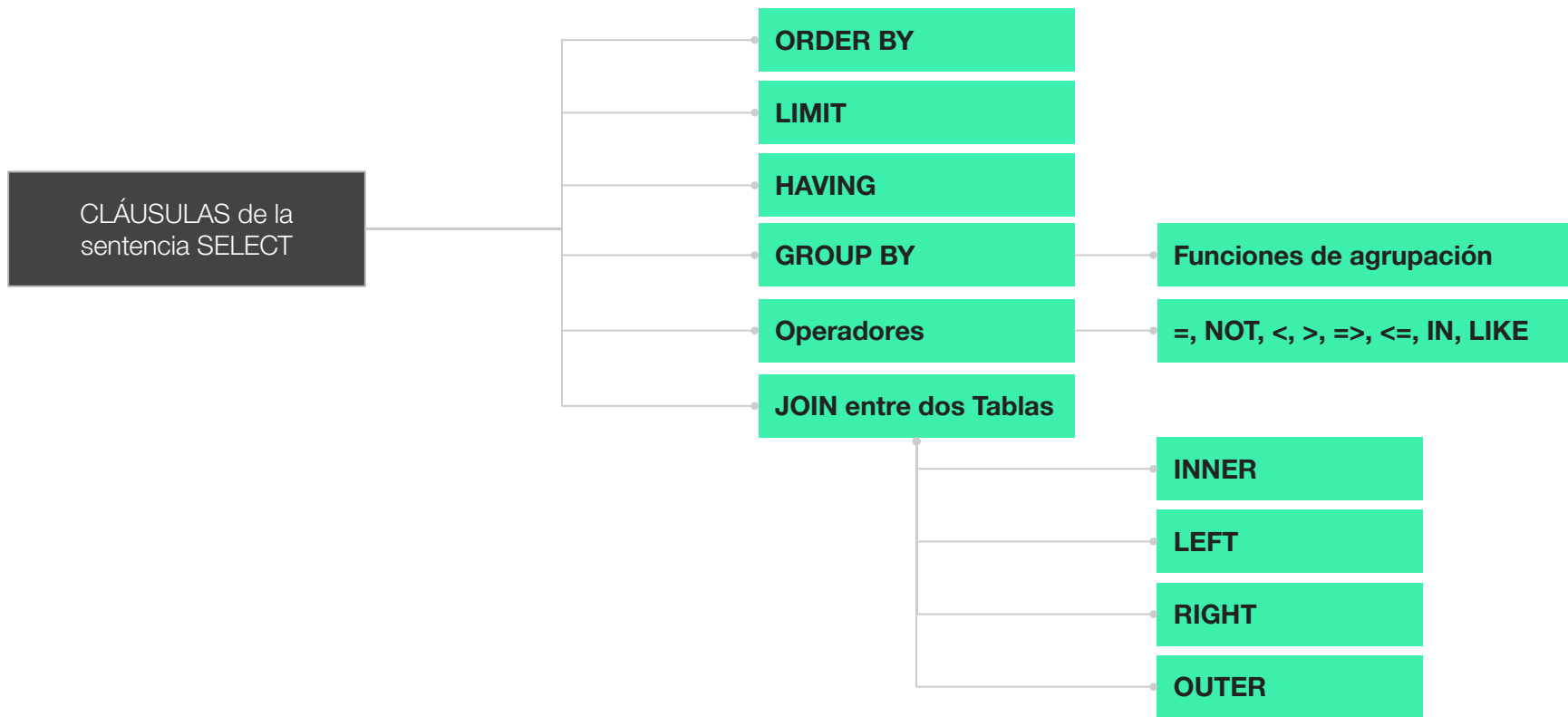
OBJETIVOS DE LA CLASE

- Reconocer e implementar las sentencias complementarias de SQL
- Identificar las funciones en SQL
- Identificar los diferentes tipos de intersección en tablas SQL

MAPA DE CONCEPTOS

MAPA DE CONCEPTOS CLASE 3

¡Para
recordar!



Comenzamos en el poder, y a su vez la complejidad que el lenguaje SQL tiene para el procesamiento, así como la obtención de información de una base de datos.

Veamos ahora de qué se tratan los sublenguajes SQL, y cuál es el objetivo de integrarlos en el uso cotidiano de éste.



SUBLENGUAJES SQL

PERO PRIMERO...

CODER HOUSE

¡VAMOS A PRACTICAR UN POCO!

OPERADORES DE COMPARACIÓN

=	<i>igual a</i>	IS [NOT] NULL	<i>no es nulo</i>	BETWEEN	<i>entre</i>
<	<i>menor a</i>	NOT	<i>NOT lógico</i>	[NOT] BETWEEN	<i>no esta entre</i>
>	<i>mayor a</i>	LIKE	<i>es como</i>	IN	<i>en (lista)</i>
<=	<i>menor o igual a</i>	[NOT] LIKE	<i>no es como</i>	[NOT] IN	<i>no esta en (lista)</i>
=>	<i>mayor o igual a</i>	IS [NOT] TRUE	<i>no es verdadero</i>	IS [NOT] FALSE	<i>no es falso</i>
!= ó <>	<i>distinto de</i>	AND	<i>AND lógico</i>	OR	<i>OR lógico</i>

Pre-calentemos con algunos ejemplos, combinando operadores de comparación mencionados la clase anterior, para ponernos en línea con el resto de los temas que siguen.

PRÁCTICAS CON OPERADORES



Veamos **cómo combinar diferentes operadores** sobre la tabla **GAME** y **COMMENTARY**, para obtener diferentes resultados posibles.

game	
id_game	INT
name	VARCHAR(100)
description	VARCHAR(300)
id_level	INT
id_class	INT
Indexes	

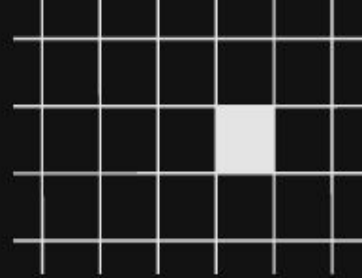
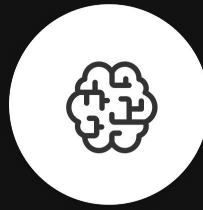
commentary	
id_commentary	INT
id_game	INT
id_system_user	INT
comment_date	DATE
commentary	VARCHAR(200)
Indexes	

Resolver las siguientes consultas:

Actividades
colaborativas

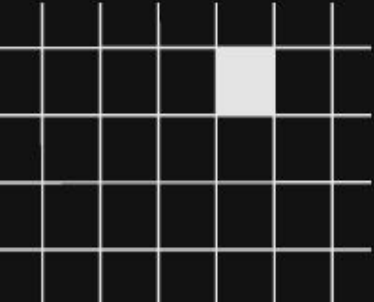


1. Todos los comentarios sobre juegos desde 2019 en adelante.
2. Todos los comentarios sobre juegos anteriores a 2011.
3. Los usuarios y texto de aquellos comentarios sobre juegos cuyo código de juego (id_game) sea 73
4. Los usuarios y texto de aquellos comentarios sobre juegos cuyo id de juego no sea 73.
5. Aquellos juegos de nivel 1.
6. Aquellos juegos sean de nivel 14 o superior.
7. Aquellos juegos de nombre 'Riders Republic' o 'The Dark Pictures: House Of Ashes'.
8. Aquellos juegos cuyo nombre empiece con 'Gran'.
9. Aquellos juegos cuyo nombre contenga 'field'.



DILEMAS DEL MUNDO SQL

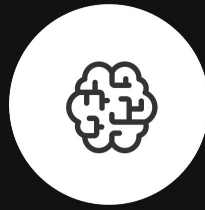
Si pensamos en un campo que debe almacenar el DNI o Cédula de identidad de las personas, ¿qué tipo de dato debería ser el de este campo?



¿NUMÉRICO O STRING?



CONTESTA EN EL CHAT



DILEMAS DEL MUNDO SQL

*Y si pensamos en un **campo contable**, que **debe almacenar el año de ejercicio** de los asientos registrados, ¿qué tipo de dato deberíamos darle a este campo?*

¿NUMÉRICO O STRING?
 CONTESTA EN EL CHAT

ORDENAMIENTO DE DATOS

ORDER BY

CODER HOUSE

LA SENTENCIA ORDER BY



Así como podemos consultar, seleccionar qué mostrar y cuáles datos filtrar, SQL nos brinda también la posibilidad de **ordenar** la información. Para el ordenamiento de la misma, debemos recurrir a la sentencia **ORDER BY**.



SELECT (campos)

FROM (tabla)

WHERE (condición)

ORDER BY

(columna)

Podemos **especificar una o más columnas**, separando las mismas por una coma, **para que el resultado SQL muestre los datos ordenados** de acuerdo a nuestro criterio o necesidad.

SELECT

...

ORDER BY

(columna)

ASC



Existen dos tipos de ordenamiento
(*de uso opcional*):

- **ASC**
- **DESC**

Si no especificamos ninguno, SQL
ordena de forma **ASC**.

PRÁCTICAS CON ORDENAMIENTO

Ejemplo
en vivo



Seleccionemos todos los registros de la tabla **GAME**, ordenados por **name** de forma ascendente (**ASC**).

```
1 • SELECT * FROM game
2   ORDER BY name ASC;
```

	id_game	name	description	id_level	id_class
▶	97	Actraiser Renaissance	viverra eget congue	3	231
	5	Age of Empires IV		2	50
	46	Alan Wake Remastered	dui luctus rutrum nulla tellus	15	137
	30	Assassins Creed Valhalla	vivamus	9	207
	67	Aven Colony	sollicitudin ut	14	238
	74	Back 4 Blood	pulvinar	9	36
	48	Back 4 Blood	ornare imperdiet sapien urna pretium	4	198
	59	Battlefield 2042	pede libero	12	31

game	
💡	id_game INT
💎	name VARCHAR(100)
💎	description VARCHAR(300)
💎	id_level INT
💎	id_class INT
Indexes ▶	

CODER HOUSE

PRÁCTICAS CON ORDENAMIENTO

Ejemplo
en vivo



Ahora seleccionemos todos los registros de la tabla **GAME**, ordenados por **id_level** de forma descendente (**DESC**) y *apliquemos el filtro donde el nombre (**name**) contenga el texto “of”*

```
1 • SELECT * FROM game
2 WHERE name like '%of%'
3 ORDER BY name ASC;
```

	id_game	name	description	id_level	id_class
▶	5	Age of Empires IV		2	50
	2	Call of Duty: Vanguard	morbi non	6	153
	25	Call of Duty: Vanguard	in felis eu sapien	3	118
	33	Call of Duty: Vanguard	proin eu	5	141
	98	Call of Duty: Vanguard	odio donec vitae nisi	13	14
	90	Ghost of Tsushima	purus	2	285
	53	God of War	sit amet diam in	5	121
	54	Kena: Bridge of Spirits	pellentesque at	1	101

game	
🔑	id_game INT
◆	name VARCHAR(100)
◆	description VARCHAR(300)
◆	id_level INT
◆	id_class INT
Indexes ▶	

CODER HOUSE

LIMIT

CODER HOUSE

LA SENTENCIA LIMIT

Ejemplo
en vivo



La utilizamos al final de toda la sentencia **SELECT**, para restringir el número de filas en el resultado de la consulta.

LIMIT espera uno o dos parámetros:

1. desde qué registro comenzar a mostrar.
2. el total de registros próximos a mostrar.

SELECT *

FROM game

ORDER BY id_class

LIMIT 5;

	id_video_game	name	description	id_class	id_level
▶	55	Kimetsu no Yaiba - The Hinokami Chr...		1	10
	57	Elden Ring	etiam pretium iaculis	5	10
	49	FIFA 22: Ultimate Team	habitasse	14	13
	98	Call of Duty: Vanguard	odio donec vitae nisi	14	13
	8	Blue Reflection: Second Light	libero rutrum ac lobortis	18	2
	93	Blast Brigade	in sapien iaculis congue vivamus	19	8
	23	The Dark Pictures: House Of Ashes		20	8
	65	Youtubers Life 2	lacus morbi quis	22	8
	87	First Class Trouble	ipsum primis in faucibus orci	25	1
	68	Saints & Sinners	placerat ante nulla justo	29	15
	59	Battlefield 2042	pede libero	31	12
	32	Metroid Dread		34	11
	74	Back 4 Blood	pulvinar	36	9

Si especificamos un solo parámetro numérico, nos devolverá sólo los primeros **N** registros del resultado de la consulta.

SELECT *

FROM game

ORDER BY id_class

LIMIT 3, 5;

	id_video_game	name	description	id_class	id_level
▶	55	Kimetsu no Yaiba - The Hinokami Chr...		1	10
	57	Elden Ring	etiam pretium iaculis	5	10
	49	FIFA 22: Ultimate Team	habitasse	14	13
	98	Call of Duty: Vanguard	odio donec vitae nisi	14	13
	8	Blue Reflection: Second Light	libero rutrum ac lobortis	18	2
	93	Blast Brigade	in sapien iaculis congue vivamus	19	8
	23	The Dark Pictures: House Of Ashes		20	8
	65	Youtubers Life 2	lacus morbi quis	22	8
	87	First Class Trouble	ipsum primis in faucibus orci	25	1
	68	Saints & Sinners	placerat ante nulla justo	29	15
	59	Battlefield 2042	pede libero	31	12
	32	Metroid Dread		34	11
	74	Back 4 Blood	pulvinar	36	9

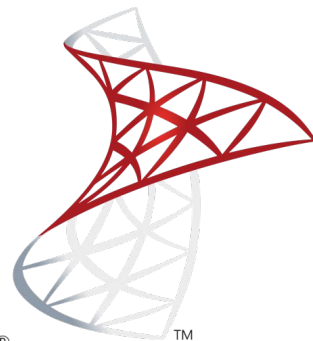
Si especificamos dos parámetros numéricos, se listarán los **N** registros del resultado de la consulta, partiendo de la posición indicada por el primer parámetro.

USO DE ~~LIMIT~~ TOP

Si alguna vez debes trabajar con **SQL Server**, ten presente que este motor de BD no incluye la sentencia **LIMIT**.

En su lugar, utiliza la **sentencia TOP** para devolverte los primeros **N registros** de la consulta solicitada. Ejemplo en código SQL Server:

```
SELECT TOP 10 *  
FROM game  
ORDER BY id_class DESC;
```



Microsoft®
SQL Server®



BREAK

¡5/10 MINUTOS Y VOLVEMOS!

ALIAS

CODER HOUSE

USO DE ALIAS

SQL Alias es una forma de **acotar el nombre** de una tabla o columna, **simplificando así su uso en sentencias SQL**

Se logra reducir las sentencias SQL cuando incluyen dos o más tablas y/o varios campos

Se debe usar la palabra reservada **AS**, seguida del alias que se desea dar a dicho campo o tabla



```
SELECT
```

```
    su.id_system_user AS id,  
    su.last_name AS l_n,  
    su.password AS pass
```

```
FROM system_user su
```

```
ORDER BY su.id_system_user;
```

	id	l_n	pass
▶	1	Besse	VA3hLGlp
	2	Fransewich	raX9Jch
	3	Barcroft	YLskgD
	4	Husher	k4I22qmvY
	5	Alliot	0DtH1nK1G
	6	Hannabus	U0H0CPUNLv6
	7	Fairlamb	gmsBTHSScu
	8	Maughan	AJOQWDFbrW
	9	Winkless	uTYSXVFT0Z
	10	Petracco	YNm8BEVusX0
	11	Burrough	Cy2RQTqwtYxU
	12	Randle	tZcDilfSEFD
	13	Eminson	5FPFX0yV9

La integración de **Alias** es ideal para acortar el nombre de campos y/o tablas.

FUNCIONES DE AGREGACIÓN

DEFINICIÓN

FUNCIONES DE AGREGACIÓN

Así como SQL nos permite obtener datos de una o más tablas, también nos permite obtener **valores simplificados o resumidos**, sobre datos específicos que necesitemos

Esto se conoce como **Funciones de Agregación o Agrupación**.

FUNCIONES DE AGRUPACIÓN



Podemos combinar funciones de ***totalización, conteo, promedios, valores mínimos y/o máximos***, entre otras, al momento de realizar la consulta

Las funciones de agregación se combinan con la cláusula **GROUP BY** y el uso de **AS**

REPASO POR LAS FUNCIONES

COUNT()

```
SELECT COUNT(*)  
AS total_level  
FROM level_game;
```

Devuelve el número total de
filas seleccionadas en una
consulta

MINO

```
SELECT MIN(id_level)  
AS min_level  
FROM level_game;
```

LEVEL_GAME	
id_level	description
1	level 1
2	level 2
3	level 3
4	level 4
...	
12	level 12
13	level 13
14	level 14
15	level 15

Devuelve el valor mínimo de un campo que especifiquemos

MAX()

```
SELECT MAX(id_level)  
AS max_level  
FROM level_game;
```

LEVEL_GAME	
id_level	description
1	level 1
2	level 2
3	level 3
4	level 4

12	level 12
13	level 13
14	level 14
15	level 15

Devuelve el valor máximo de un campo que especifiquemos

SUMO

```
SELECT SUM(value)
FROM vote
WHERE id_game = 1;
```

VOTE			
id_vote	value	id_game	id_user
123	0	1	614
136	10	1	281
297	1	1	944
307	1	1	145
483	6	1	905
663	5	1	62
671	6	1	717
739	10	1	135
757	9	1	876
763	2	1	263
671	6	2	717
739	10	2	135
740	4	2	898
757	9	2	876

Devuelve la suma de los valores de
un campo que especifiquemos

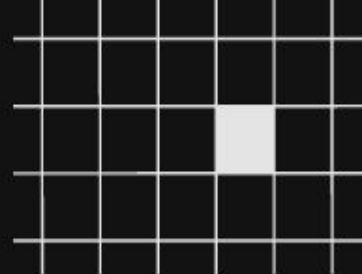
CODER HOUSE

AVG()

```
SELECT AVG(value)
FROM vote
WHERE id_game = 1;
```

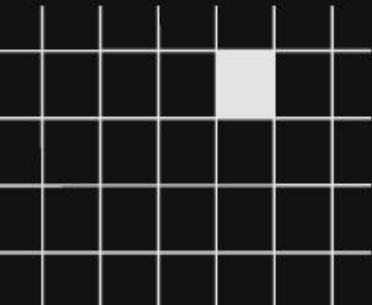
VOTE			
id_vote	value	id_game	id_user
123	0	1	614
136	10	1	281
297	1	1	944
307	1	1	145
483	6	1	905
663	5	1	62
671	6	1	717
739	10	1	135
757	9	1	876
763	2	1	263
671	6	2	717
739	10	2	135
740	4	2	898
757	9	2	876

Devuelve el valor promedio de un campo que especifiquemos



EJEMPLO EN VIVO

VEAMOS A LAS FUNCIONES DE AGRUPACIÓN EN ACCIÓN.



GROUP BY

GROUP BY

Ejemplo
en vivo



Como ya mencionamos, la cláusula **GROUP BY** es fundamental para **usarse junto** a las **funciones de agregación**, la debemos **utilizar** cuando debemos obtener información que nace de la agrupación de registros

Por lo tanto, será el aliado ideal para usarse junto a **COUNT()**, **SUM()** y **AVG()**

Ejemplo
en vivo



SELECT

id_system_user AS user,

COUNT(*) AS games_by_user

FROM play

GROUP BY id_system_user;

id_game	id_system_user	completed
17	1	1
47	2	1
48	3	1
49	3	1
74	3	1
77	5	0
43	7	0
64	7	0
53	8	0

user	games_by_user
1	1
2	1
3	3
5	1
7	2
8	2
9	1
10	1
12	1

CODEC MODEL

HAVING

HAVING



HAVING, al igual **WHERE**, permite **establecer condiciones para filtrar los resultados**. Para ello, **necesitamos generar campos con resultados** filtrados, para luego sumar a HAVING

Por lo tanto, debemos tener presente que esta sentencia solo funciona con campos generados a partir de una función

SELECT

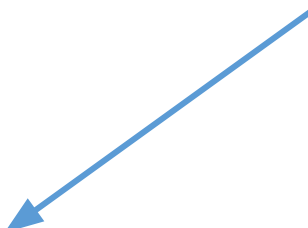
id_system_user AS user,

COUNT(*) AS games_by_user


FROM play

GROUP BY id_system_user

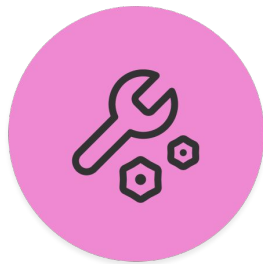
HAVING COUNT(*) > 1;



id_game	id_system_user	completed
17	1	1
47	2	1
48	3	1
49	3	1
74	3	1
77	5	0
43	7	0
64	7	0
53	8	0



user	games_by_user
3	3
7	2
8	2
13	3
15	2
26	2
28	2
33	2
34	2



PRÁCTICAS DE AGRUPAMIENTO

Implementemos las diferentes opciones de orden y agrupamiento de datos

Tiempo estimado: 10 minutos

PRÁCTICAS DE AGRUPAMIENTO

Desafío
generico



Partiendo de la tabla, debes determinar qué resultado obtendrás, implementando las consultas de la diapositiva siguiente

Comentaremos los resultados en la clase, a través del sistema de chat

PRÁCTICAS DE AGRUPAMIENTO

Desafío
generico



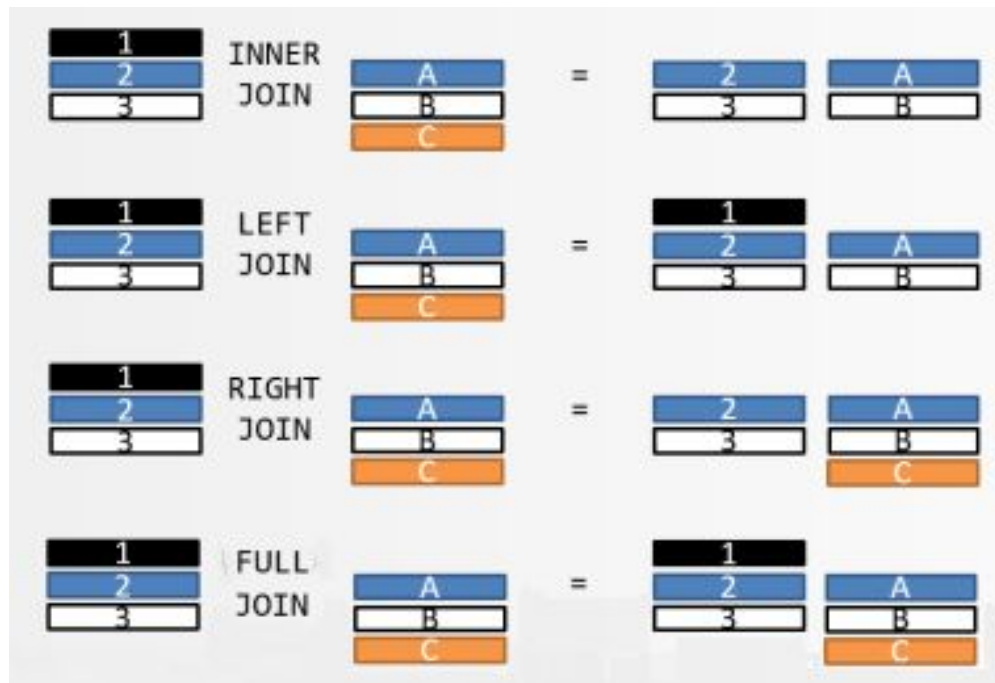
- 1) `SELECT * FROM commentary ORDER BY id_system_user desc;`
- 2) `SELECT * FROM commentary ORDER BY id_system_user LIMIT 3;`
- 3) `SELECT COUNT(id_system_user) AS comments, id_system_user
FROM commentary GROUP BY id_system_user ;`
- 4) `SELECT COUNT(id_system_user) AS comments, id_system_user
FROM commentary
GROUP BY id_system_user
HAVING comments > 2;`

JOIN

CODER HOUSE

CONCEPTO

JOIN permite **combinar registros de diferentes tablas**, complementándose con la cláusula **ON**, la cual establece la condición por la cual queremos unir las tablas. Generalmente son campos comunes entre tablas. Los **tipos de JOIN** importantes, son cuatro:



INNER JOIN

INNER JOIN

INNER JOIN, o **JOIN**, retorna **todas las filas de las dos tablas** siempre que haya **coincidencia** por el campo declarado en el **ON**.

El resultado es **NULL** cuando no hay coincidencia alguna.

```
SELECT      nombres_columnas
FROM        tabla1
INNER JOIN  tabla2
ON  tabla1.columna_relacion
=  tabla2.columna_relacion;
```

SINTAXIS: inner join

Armamos la consulta utilizando **INNER JOIN** para llegar al resultado deseado.

PLAY	"=	GAME	
id_system	id_game	name	id_level
user	game	name	level
1	17	Mario Party Superstars	7
2	47	Disco Elysium	13
3	48	Back 4 Blood	4
3	49	FIFA 22: Ultimate Team	13
3	74	Back 4 Blood	9
5	77	Dragon Ball Z Kakarot	1
7	64	My Friend Peppa Pig	12
7	43	Dungeon Encounters	8
8	71	The Good Life	6

```
SELECT id_system_user as user, g.id_game as game, name,  
       id_level as level  
FROM play p INNER JOIN game g ON (p.id_game = g.id_game);
```

LEFT JOIN

LEFT JOIN

LEFT JOIN, retorna **todas las filas de la tabla izquierda** que coincidan con las **filas de la tabla derecha**

El resultado es **NULL** del lado derecho, cuando no hay coincidencia.

```
SELECT      nombres_columnas
FROM        tabla1
LEFT JOIN   tabla2
ON  tabla1.columna_relacion
=  tabla2.columna_relacion;
```

SINTAXIS: left join

Armamos la consulta utilizando **LEFT JOIN** para llegar al resultado deseado

PLAY	"="	GAME	
id_system	id_game	name	id_level
user	game	name	level
62	1	Forza Horizon 5	2
135	1	Forza Horizon 5	2
82	2	Call of Duty: Vanguard	6
405	2	Call of Duty: Vanguard	6
577	2	Call of Duty: Vanguard	6
58	3	Shin Megami Tensei 5	3
null	4	Marvels Guardianes de la Galaxia	4
null	5	Age of Empires IV	2
176	6	Football Manager 22	8
265	6	Football Manager 22	8

```
SELECT id_system_user as user, g.id_game as game,  
       name, id_level as level  
FROM game g LEFT JOIN play p  
ON (p.id_game = g.id_game);
```

RIGHT JOIN

RIGHT JOIN

RIGHT JOIN, retorna **todas las filas de la tabla derecha** que coincidan con las **filas de la tabla izquierda**

El resultado es **NULL** cuando no hay coincidencia del lado izquierdo

```
SELECT      nombres_columnas
FROM        tabla1
RIGHT JOIN  tabla2
ON  tabla1.columna_relacion
=  tabla2.columna_relacion;
```

SINTAXIS: right join

Armamos la consulta utilizando **RIGHT JOIN** para llegar al resultado deseado

PLAY	"="	GAME	
id_system	id_game	name	id_level
user	game	name	level
62	1	Forza Horizon 5	2
135	1	Forza Horizon 5	2
82	2	Call of Duty: Vanguard	6
405	2	Call of Duty: Vanguard	6
577	2	Call of Duty: Vanguard	6
58	3	Shin Megami Tensei 5	3
null	4	Marvels Guardianes de la Galaxia	4
null	5	Age of Empires IV	2
176	6	Football Manager 22	8
265	6	Football Manager 22	8

```
SELECT id_system_user as user, g.id_game as game,  
       name, id_level as level  
FROM play p RIGHT JOIN game g  
ON (p.id_game = g.id_game);
```

FULL JOIN

FULL JOIN

FULL JOIN, retorna **todas las filas de la tabla derecha** y también las **filas de la tabla izquierda**.

Básicamente combina los resultados de **LEFT** y **RIGHT JOIN**, pudiendo tener valores nulo de ambos lados.

Nota: MySQL no soporta FULL JOIN.

```
SELECT      nombres_columnas
FROM        tabla1
OUTER JOIN  tabla2
ON  tabla1.columna_relacion
=  tabla2.columna_relacion;
```

SINTAXIS: full join

Armamos la consulta utilizando **FULL JOIN** para llegar al resultado deseado.

PLAY	"="	GAME	
id_system	id_game	name	id_level
user	game	name	level
4	null	null	null
6	null	null	null
62	1	Forza Horizon 5	2
135	1	Forza Horizon 5	2
82	2	Call of Duty: Vanguard	6
405	2	Call of Duty: Vanguard	6
577	2	Call of Duty: Vanguard	6
58	3	Shin Megami Tensei 5	3
null	4	Marvels Guardianes de la Gala	4
null	5	Age of Empires IV	2
176	6	Football Manager 22	8
265	6	Football Manager 22	8

```
SELECT s.id_system_user as user, g.id_game as game,  
       name, id_level as level  
FROM system_user s FULL JOIN play p  
      ON (s.id_system_user = p.id_system_user)  
      FULL JOIN game g ON (p.id_game = g.id_game);
```




DIAGRAMA ENTIDAD-RELACIÓN

Iniciaremos el diseño de la base de datos de nuestro proyecto final.

DIAGRAMA ENTIDAD-RELACIÓN

Formato: PPT o Slides nombrado como “Lista+Apellido”

Sugerencia: En caso de ser un archivo en línea, activar permisos de acceso.
Utilizar como guía la práctica y el desafío genérico de la clase.

Desafío
entregable



>> Consigna:

- Diseñar el modelo entidad-relación de al menos dos de las temáticas elegidas para el proyecto final.

>> Aspectos a incluir en el entregable:

- Definir al menos cinco tablas
- Crear el diagrama de entidad-relación con todos sus componentes:
 - Entidades
 - Acciones de relacionamiento
 - Tipos de relación
 - Campos clave

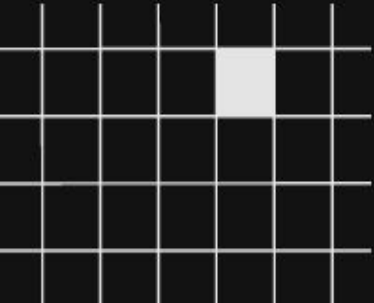
¿PREGUNTAS?





¡MUCHAS GRACIAS!

Resumen de lo visto en clase hoy:

- Prácticas con Operadores de comparación
 - ORDER BY - LIMIT - HAVING
 - GROUP BY - Funciones de agrupación
 - JOIN entre dos tablas
- 



OPINA Y VALORA ESTA CLASE

#DEMOCRATIZANDO LA EDUCACIÓN

CODER HOUSE