IIC2413

AYUDANTÍA 7

SQL 2

¿QUÉ VEREMOS?

ORDER BY

Operadores de Conjunto

LIKE

HAVING

Agregacion

Consultas Anidadas

Ejercicios

ORDER BY

Ordenar resultados según una o más columnas.

Gimnasta

id	nombre	edad
1	María	24
2	Elena	23
3	Raimundo	25

```
SELECT nombre, edad
FROM gimnasta
ORDER BY edad ASC;
```

nombre	edad
Elena	23
María	24
Raimundo	25

ORDER BY

Ordenar resultados según una o más columnas.

Gimnasta

id	nombre	edad
1	María	24
2	Elena	23
3	Raimundo	25

```
SELECT nombre, edad
FROM gimnasta
ORDER BY edad DESC;
```

nombre	edad
Raimundo	25
María	24
Elena	23

ORDER BY

Ordenar resultados según una o más columnas.

Gimnasta

id	nombre	edad	puntaje
1	María	24	8
2	Elena	23	8
3	Raimundo	25	9
4	Tomás	26	10

En caso de empate, se usa el segundo criterio

```
SELECT nombre, edad, puntaje
FROM gimnasta
ORDER BY puntaje DESC, edad DESC;
```

nombre	edad	puntaje
Tomás	26	10
Raimundo	25	9
María	24	8
Elena	23	8

ORDER BY

No solo sirve para números...

🧵 1. Strings (VARCHAR, TEXT, etc.)

Se ordenan alfabéticamente
Por defecto, va de la A a la Z con ASC, y de la Z a la A con DESC.

📅 2. Fechas (DATE, TIMESTAMP, TIME)

Se ordenan cronológicamente..
“antiguo” < “reciente”

✅ 3. Booleanos (BOOLEAN)

En PostgreSQL: false < true.
Con **ORDER BY** bool_col **ASC**, los false van primero.

OPERADORES DE CONJUNTOS

UNION

Une resultados sin duplicados

Gimnasta

id	nombre	edad
1	María	24
2	Elena	23
3	Raimundo	25

Juez

id	nombre	edad
1	Francisco	32
2	Elena	40
3	Raimundo	29

```
SELECT nombre FROM gimnasta
UNION
SELECT nombre FROM juez;
```

UNION

Une resultados sin duplicados

Las filas repetidas de “Elena” y “Raimundo” no se toman en cuenta

nombre
María
Francisco
Elena
Raimundo

```
SELECT nombre FROM gimnasta
UNION
SELECT nombre FROM juez;
```

UNION ALL

Une resultados (admite duplicados)

Gimnasta

id	nombre	edad
1	María	24
2	Elena	23
3	Raimundo	25

Juez

id	nombre	edad
1	Francisco	32
2	Elena	40
3	Raimundo	29

```
SELECT nombre FROM gimnasta
UNION ALL
SELECT nombre FROM juez;
```

UNION ALL

Une resultados (admite duplicados)

nombre
María
Elena
Raimundo
Francisco
Elena
Raimundo

```
SELECT nombre FROM gimnasta
UNION ALL
SELECT nombre FROM juez;
```

EXCEPT

"Muéstrame la tabla A menos lo que también están en la tabla B"

Gimnasta

id	nombre	edad
1	María	24
2	Elena	23
3	Raimundo	25
4	Tomás	26

Juez

id	nombre	edad
1	Francisco	32
2	Elena	40
3	Raimundo	29
4	Agustin	35

```
SELECT nombre FROM gimnasta
EXCEPT
SELECT nombre FROM juez;
```

EXCEPT

"Muéstrame la tabla A menos lo que también está en la tabla B"

“María” y “Tomás” son los nombres que estan en la tabla “A” que no estan en “B”

nombre
María
Tomás

```
SELECT nombre FROM gimnasta
EXCEPT
SELECT nombre FROM juez;
```

!!El orden es importante!!

Muestra las filas de la primera consulta que no están en la segunda

INTERSECT

Devuelve solo las filas que aparecen en ambas consultas, es decir, la intersección entre dos conjuntos de resultados.

Gimnasta

id	nombre	edad
1	María	24
2	Elena	23
3	Raimundo	25
4	Tomás	26

Juez

id	nombre	edad
1	Francisco	32
2	Elena	40
3	Raimundo	29
4	Agustin	35

```
SELECT nombre FROM gimnasta
INTERSECT
SELECT nombre FROM juez;
```

INTERSECT

Devuelve solo las filas que aparecen en ambas consultas, es decir, la intersección entre dos conjuntos de resultados.

“Elena” y “Raimundo” aparecen en ambas tablas

nombre
Elena
Raimundo

```
SELECT nombre FROM gimnasta
INTERSECT
SELECT nombre FROM juez;
```

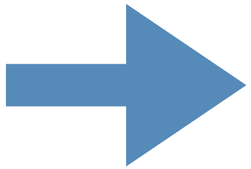
LIKE

Sirve para comparar strings
Utiliza los siguientes símbolos:

- "%" : Cualquier secuencia de caracteres
- "_" : Cualquier Caracter (uno)

```
SELECT * FROM gimnasta WHERE
nombre LIKE '%Rai%' ;
```

id	nombre	edad
1	María	24
2	Elena	23
3	Raimundo	25



id	nombre	edad
3	Raimundo	25

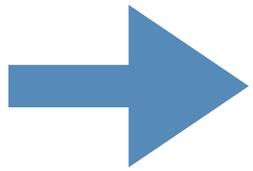
AGREGACION

Funciones que resumen o agrupan datos.

SUM, MIN, MAX, COUNT, AVG, etc

SELECT SUM(cantidad) **FROM** ventas **WHERE** precio >15;

id	producto	cantidad	precio
1	Agua	24	10
2	Bebida	23	20
3	Leche	25	30



sum
48

GROUP BY

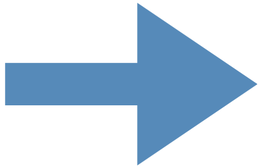
Agrupa los resultados según los atributos especificados con **GROUP BY**. Donde para cada grupo generado se aplica la agregación de manera independiente

id	curso	estudiante	nota	fecha_evaluaciones
1	Álgebra	Ana	5	01-03-2025
2	Álgebra	Tomás	4.5	03-03-2025
3	Programación	Ana	6	05-03-2025
4	Programación	Camila	5.5	10-03-2025
6	Física	Diego	4	15-03-2025
7	Física	Camila	4.5	18-03-2025

GROUP BY

SELECT curso, **AVG**(nota) **as** promedio **FROM** evaluaciones **WHERE** fecha >'05-03-2025' **GROUP BY** curso;

id	curso	estudiante	nota	fecha_evaluaciones
1	Álgebra	Ana	5	01-03-2025
2	Álgebra	Tomás	4.5	03-03-2025
3	Programación	Ana	6	05-03-2025
4	Programación	Camila	5.5	10-03-2025
6	Física	Diego	4	15-03-2025
7	Física	Camila	4.5	18-03-2025



curso	promedio
Programación	5.5
Física	4.25

HAVING

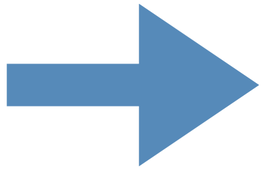
Se usa para filtrar resultados despues de agruparlos con **GROUP BY**

- **WHERE** filtra antes del agrupamiento
- **HAVING** filtra despues del agrupamiento (Sobre los grupos ya formados)

HAVING

SELECT curso, **AVG**(nota) **as** promedio **FROM** evaluaciones **WHERE** fecha >'05-03-2025' **GROUP BY** curso **HAVING AVG**(nota)>5.5;

id	curso	estudiante	nota	fecha_evaluaciones
1	Álgebra	Ana	5	01-03-2025
2	Álgebra	Tomás	4.5	03-03-2025
3	Programación	Ana	6	05-03-2025
4	Programación	Camila	5.5	10-03-2025
5	Programación	Tomás	6.5	12-03-2025
6	Física	Diego	4	15-03-2025
7	Física	Camila	4.5	18-03-2025



curso	promedio
Programacion	6

CONSULTAS ANIDADAS

Como Condicion

bandas

id	nombre	vocalista
1	A	Alvaro
2	B	Ana
3	C	Julieta
4	D	Adam

estudiantes

id	nombre
100	Alvaro
200	Julieta
300	Ana
400	Felipe

toco_en

nombre_banda	nombre_festival
C	'Cosquin Rock'
B	'Lollapalooza'
B	'Coachella'
D	'Lollapalooza'

CONSULTAS ANIDADAS

Como Condición

```
SELECT bandas.nombre FROM bandas,
estudiantes WHERE bandas.vocalista =
estudiantes.nombre AND  bandas.nombre
IN (
SELECT toco_en.nombre_banda FROM
toco_en WHERE toco_en.nombre_festival =
'Lollapalooza'
);
```

CONSULTAS ANIDADAS

Como Condición

SELECT bandas.nombre **FROM** bandas, estudiantes **WHERE** bandas.vocalista = estudiantes.nombre **...**



nombre
A
B
C



... AND bandas.nombre **IN** (
SELECT toco_en.nombre_banda **FROM** toco_en **WHERE** toco_en.nombre_festival = 'Lollapalooza'
);

nombre
B

CONSULTAS ANIDADAS

Como Condición

Tipos de condiciones

- s **IN** R
- s > **ALL** R
- s > **ANY** R
- **EXISTS** R

CONSULTAS ANIDADAS

Como Condición

```
SELECT nombre FROM estudiantes WHERE nombre
IN (SELECT vocalista
FROM bandas);
```

Devuelve estudiantes que son vocalistas de alguna banda.

```
SELECT nombre, calificacion FROM pelicula WHERE calificacion >
ALL (SELECT calificacion FROM pelicula
WHERE año = 2020);
```

Películas cuya calificación es mayor a todas las películas del 2020.
(Si se usa ANY, seria mayor que alguna película del 2020)

```
SELECT nombre FROM clientes  WHERE
EXISTS (
SELECT * FROM compras
WHERE compras.id_cliente = clientes.id);
```

Para cada cliente de la tabla clientes, busca si hay al menos una fila en compras que tenga el mismo id_cliente
Si existe → el cliente sí ha comprado algo → se incluye en el resultado.

CONSULTAS ANIDADAS

Como Joins

```
SELECT profesores.nombre, maximos.año
FROM profesores, (
SELECT hizo_clases.id_profesor AS id,
MAX(curso.año) AS año FROM hizo_clase, cursos
WHERE hizo_clase.id_curso=cursos.id
GROUP BY hizo_clase.id_profesor
) AS maximos
WHERE profesores.id=maximos.id
```

CONSULTAS ANIDADAS

Como Joins

profesores

id	nombre
1	Carolina
2	Javier
3	Francisca
4	Rodrigo

cursos

id	nombre_curso	año
10	Bases de Datos	2021
11	Álgebra	2020
12	Redes	2022
13	Programacion	2020
14	IA	2023

hizo_clase

id_profesor	id_curso
1	10
1	14
2	11
2	12
3	13
4	12

CONSULTAS ANIDADAS

Como Joins

```
SELECT profesores.nombre, maximos.año
FROM profesores, (
        SELECT hizo_clases.id_profesor AS id,
        MAX(curso.año) AS año FROM hizo_clase, cursos
        WHERE hizo_clase.id_curso=cursos.id
        GROUP BY hizo_clase.id_profesor
) AS maximos
WHERE profesores.id=maximos.id
```

CONSULTAS ANIDADAS

Como Joins

SELECT hizo_clases.id_profesor **AS** id, **MAX**(curso.año) **AS** año **FROM** hizo_clase, cursos **WHERE** hizo_clase.id_curso=cursos.id **GROUP BY** hizo_clase.id_profesor

id_profesor	id_curso	año
1	10	2021
1	14	2023
2	11	2020
2	12	2022
3	13	2020
4	12	2022



id	año
1	2023
2	2022
3	2020
4	2022

CONSULTAS ANIDADAS

Como Joins

```
SELECT profesores.nombre, maximos.año
FROM profesores, (
) AS maximos
WHERE profesores.id=maximos.id
```

maximos

id	año
1	2023
2	2022
3	2020
4	2022



nombre	año
Carolina	2023
Javier	2022
Francisca	2020
Rodrigo	2022

NULOS

Cuando no se sabe si el valor existe o no existe, o simplemente no se tiene la informacion

id	marca	modelo	patente	dueño_id
1	Toyota	Supra	ABCD12	101
2	Nissan	Sentra	EFGH34	NULL
3	Chevrolet	Silverado	IJKL56	242
4	Tesla	Model 3	NULL	123

NULOS

Condicion para ver si el elemento es nulo

SELECT * **FROM** R **WHERE** R.b **IS NULL**

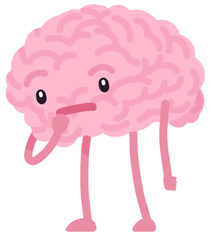
Condicion para ver si el elemento no es nulo

SELECT * **FROM** R **WHERE** R.b **IS NOT NULL**

Las consultas con operaciones (+, -, *, etc) con NULL se vuelven nulas

Las consultas de agregacion (SUM, MIN, COUNT) ignoran los NULL

EJERCICIO



- **pelicula**(nombre, anho, calificacion)
- **actor**(nombre, genero)
- **personaje**(p_nombre, p_anho, a_nombre, personaje)

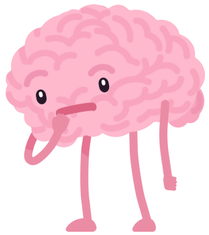
La tabla **personaje** usa llaves foráneas que hacen referencia a las tablas de **actor**(a_nombre) y **pelicula**(p_nombre, p_anho).

¿Qué actores tienen nombres que comienzan con "M"?

¿Qué actores han participado en más de 2 películas?

¿Qué películas tienen la calificación más alta de toda la base de datos?

SOLUCION



¿Qué actores tienen nombres que comienzan con "M"?

```
SELECT *
FROM actor
WHERE nombre LIKE 'M%';
```

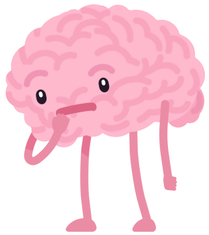
¿Qué actores han participado en más de 2 películas?

```
SELECT a_nombre, COUNT(a_nombre) AS cantidad_peliculas
FROM personaje
GROUP BY a_nombre
HAVING COUNT(a_nombre) > 2;
```

¿Qué películas tienen la calificación más alta de toda la base de datos?

```
SELECT nombre, anho, calificacion
FROM pelicula
WHERE calificacion = (
SELECT MAX(calificacion)
FROM pelicula
);
```

EJERCICIO



I1 2024-1

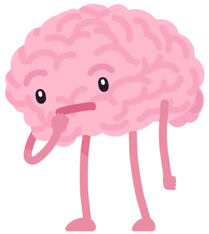
Considere el siguiente esquema:

- Cliente(id_cliente(PK), nombre, correo)
- Pedido(id_pedido(PK), id_cliente(FK), fecha_pedido, estado)
- Venta(id_venta(PK), id_cliente(FK), fecha venta, monto)

a) Considere que se requiere realizar seguimiento de los clientes que han realizado pedidos mas de una vez. Diseñe una consulta en SQL que identifique a dichos clientes junto con el numero de pedidos que han realizado.

b) Escriba una consulta en SQL que devuelva los clientes que hayan realizado al menos dos compras con un monto total superior a $500 en los ultimos tres meses.

SOLUCION

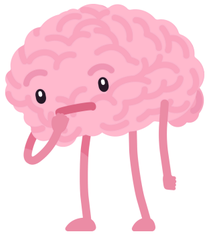


- Cliente(id_cliente(PK), nombre, correo)
- Pedido(id_pedido(PK), id_cliente(FK), fecha_pedido, estado)
- Venta(id_venta(PK), id_cliente(FK), fecha venta, monto)

a) Considere que se requiere realizar seguimiento de los clientes que han realizado pedidos mas de una vez. Diseñe una consulta en SQL que identifique a dichos clientes junto con el numero de pedidos que han realizado.

```
SELECT cliente.id_cliente, cliente.nombre, COUNT(pedido.id_pedido) AS numero_de_pedidos
FROM cliente, pedido
WHERE cliente.id_cliente = pedido.id_cliente
GROUP BY cliente.id_cliente, cliente.nombre
HAVING COUNT(pedido.id_pedido)>1;
```

SOLUCION

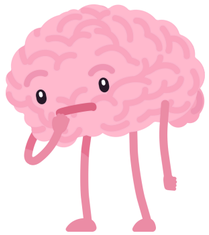


- Cliente(id_cliente(PK), nombre, correo)
- Pedido(id_pedido(PK), id_cliente(FK), fecha_pedido, estado)
- Venta(id_venta(PK), id_cliente(FK), fecha venta, monto)

b) Escriba una consulta en SQL que devuelva los clientes que hayan realizado al menos dos compras con un monto total superior a $500 en los ultimos tres meses (considere la fecha de hoy)

```
SELECT id_cliente
FROM ventas
WHERE fecha_venta >= '2025-01-11'
GROUP BY id_cliente
HAVING COUNT(id_venta) >= 2 AND SUM(monto) >500;
```

EJERCICIO



I1 2024-2

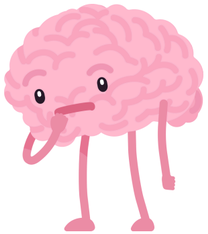
Una empresa de despacho cuenta con el siguiente esquema para la capacidad de despacho de los productos que venden

- zona_despacho(id_zona(PK), nombre_zona, comuna, region)
- slot(id_slot(PK), dia(FK), hora_inicio, hora_fin)
- asig_capac(id_capac(PK), id_zona(FK), id_zlot(FK), capac_sm, capac_med, capac_big, precio_sm, precio_med,precion_big)

a) Cree una consulta SQL que calcule el precio de enviar 1 producto small, 3 medios y 1 big a la zona con id 5.

b) Cree una consulta SQL que entregue por zona de despacho el slot con menor capacidad de despacho total de productos de todos los tamaños

SOLUCION

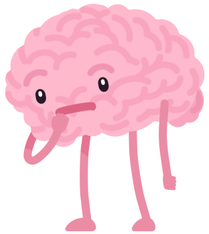


- zona_despacho(id_zona(PK), nombre_zona, comuna, region)
- slot(id_slot(PK), dia(FK), hora_inicio, hora_fin)
- asig_capac(id_capac(PK), id_zona(FK), id_zlot(FK), capac_sm, capac_med, capac_big, precio_sm, precio_med,precion_big)

a) Cree una consulta SQL que calcule el precio de enviar 1 producto small, 3 medios y 1 big a la zona con id 5.

```
SELECT precio_sm + 3*precio_med + precio_big
FROM asig_capac
WHERE id_zona=5;
```

SOLUCION



- zona_despacho(id_zona(PK), nombre_zona, comuna, region)
- slot(id_slot(PK), dia(FK), hora_inicio, hora_fin)
- asig_capac(id_capac(PK), id_zona(FK), id_zlot(FK), capac_sm, capac_med, capac_big, precio_sm, precio_med,precion_big)

b) Cree una consulta SQL que entregue por zona de despacho el slot con menor capacidad de despacho total de productos de todos los tamaños

```
SELECT id_zona, id_slot, (capac_sm + capac_med + capac_big)
FROM asig_capac
WHERE (capac_sm + capac_med + capac_big) <  ALL (
        SELECT capac_sm + capac_med + capac_big
        FROM asig_capac AS asig_capac2
        WHERE asig_capac2.id__zona = asig_capac.id_zona
        AND asig_capac2.id_slot <> asig_capac.id_slot);
```

IIC2413

AYUDANTÍA 7

SQL 2