Contenido

[Select 2](#_Toc202854582)

[Ejercicio 1 2](#_Toc202854583)

[Ejercicio 2 4](#_Toc202854584)

[Basics joins 9](#_Toc202854585)

[Ejercicio 1 9](#_Toc202854586)

[Ejercicio 2 11](#_Toc202854587)

[Ejercicio 3 15](#_Toc202854588)

[Funciones de agregación básicas 19](#_Toc202854589)

[Ejercicio 1 19](#_Toc202854590)

[Ejercicio 2 22](#_Toc202854591)

[Clasificación y agrupación 25](#_Toc202854592)

[Ejercicio 1 25](#_Toc202854593)

[Selección avanzada y uniones 28](#_Toc202854594)

[Subconsultas 29](#_Toc202854595)

[Funciones de cadena avanzadas 29](#_Toc202854596)

[Ejercicio 1 29](#_Toc202854597)

Genera un ejercicio nivel Y para practicar Z en SQL. Dame también, la salida esperada con el código solución. Además, agrega la tabla o tablas según sea el caso (en formato de código y en forma de tabla directamente)

Estructura de la sección de chatgpt: cada ejercicio con título 3 contiene 3 subejercicios de nivel fácil, intermedio y difícil

Con amarillo pongo mis soluciones de código

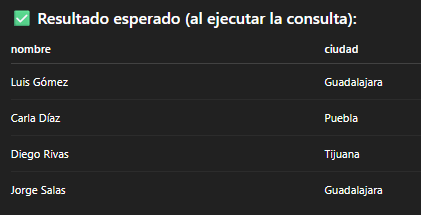
# Select

## Ejercicio 1



💡 Ejercicio básico:

**Pregunta**:  
Escribe una consulta SQL que muestre los nombres y las ciudades de todos los clientes que han realizado compras por más de 2000 unidades monetarias.



Select c.nombre, c.ciudad

From clientes c

Where total\_compras > 2000;

🧠 **Ejercicio intermedio**:

**Pregunta**:  
Escribe una consulta SQL que muestre el **nombre, ciudad y total\_compras** de los **3 clientes con mayores compras**, ordenados de mayor a menor.



Select c.nombre, c.ciudad, c.total\_compras

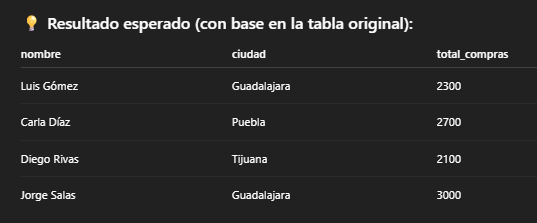
From clientes c

Order by total\_compras desc

Limit 3;

🧩 **Ejercicio difícil**:

**Pregunta**:  
Escribe una consulta SQL que muestre el **nombre, ciudad y total\_compras** de todos los clientes cuya cantidad de compras sea **superior al promedio de compras** de todos los clientes.



Recuerde que las funciones de agregación no se usan directamente en el web necesitamos una subconsulta para poder hacerlo

Select c.nombre, c.ciudad, c.total\_compras

From clientes c

WHERE c.total\_compras > (

SELECT AVG(total\_compras)

FROM clientes

);

## Ejercicio 2

Ejercicio fácil

CREATE TABLE Empleados (

ID INT,

Nombre VARCHAR(50),

Departamento VARCHAR(50),

Salario INT

);

INSERT INTO Empleados (ID, Nombre, Departamento, Salario) VALUES

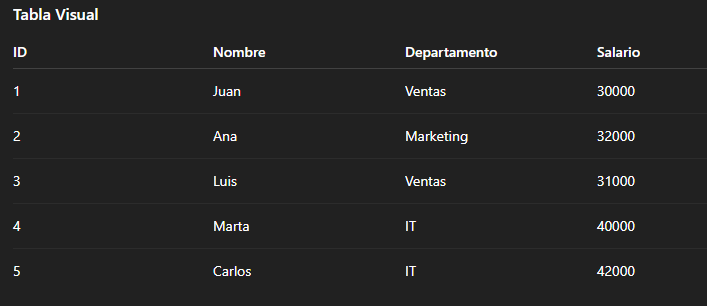
(1, 'Juan', 'Ventas', 30000),

(2, 'Ana', 'Marketing', 32000),

(3, 'Luis', 'Ventas', 31000),

(4, 'Marta', 'IT', 40000),

(5, 'Carlos', 'IT', 42000);



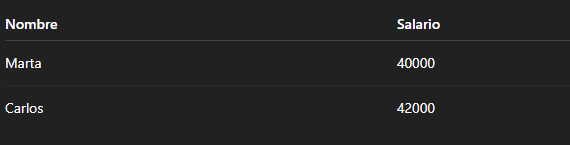
Mi solucion

**Enunciado:**  
Selecciona el nombre y el salario de todos los empleados que trabajan en el departamento "IT".

Select e.nombre, e.salario

From empleados e

Where departamento = ‘IT’;



Ejercicio intermedio

CREATE TABLE Productos (

ID INT,

Nombre VARCHAR(50),

Categoria VARCHAR(50),

Precio DECIMAL(6,2),

Stock INT

);

INSERT INTO Productos (ID, Nombre, Categoria, Precio, Stock) VALUES

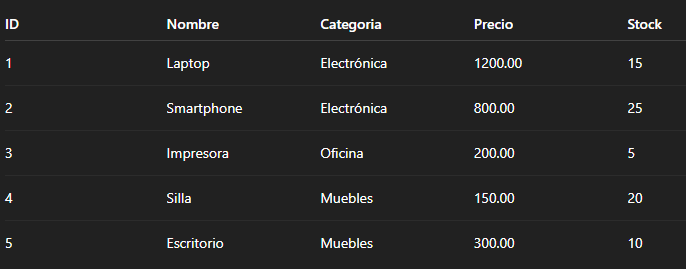
(1, 'Laptop', 'Electrónica', 1200.00, 15),

(2, 'Smartphone', 'Electrónica', 800.00, 25),

(3, 'Impresora', 'Oficina', 200.00, 5),

(4, 'Silla', 'Muebles', 150.00, 20),

(5, 'Escritorio', 'Muebles', 300.00, 10);

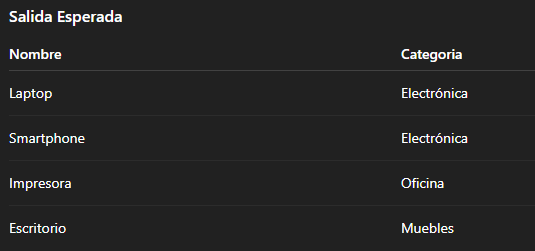


**Enunciado:**  
Selecciona el nombre y la categoría de los productos cuyo precio es mayor a 300 o el stock es menor o igual a 10.

Select p.nombre, p.categoria

From productos p

Where p.precio > 300 or p.stock <= 10;



Ejercicio difícil

CREATE TABLE Clientes (

ID INT,

Nombre VARCHAR(50),

Ciudad VARCHAR(50)

);

INSERT INTO Clientes (ID, Nombre, Ciudad) VALUES

(1, 'Pedro', 'Madrid'),

(2, 'Lucía', 'Barcelona'),

(3, 'Andrés', 'Madrid'),

(4, 'María', 'Valencia');

CREATE TABLE Pedidos (

ID INT,

ClienteID INT,

Importe DECIMAL(8,2)

);

INSERT INTO Pedidos (ID, ClienteID, Importe) VALUES

(1, 1, 250.00),

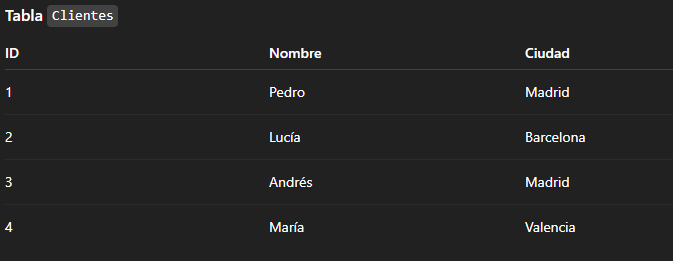
(2, 2, 450.00),

(3, 1, 150.00),

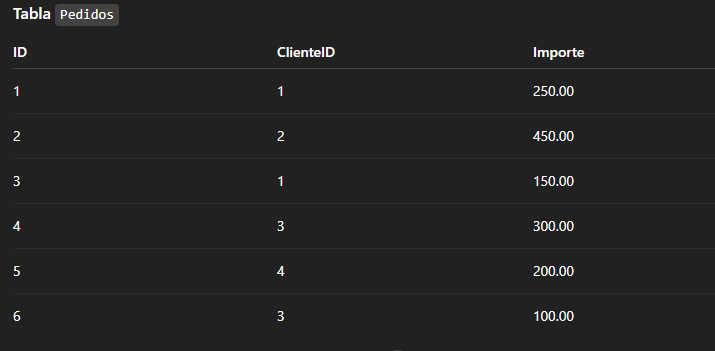
(4, 3, 300.00),

(5, 4, 200.00),

(6, 3, 100.00);



ID es la clave primaria de la tabla clientes



ID es la clave primaria de la tabla pedidos y clienteID es la llave foránea que relaciona ambas tablas

**Enunciado:**  
Muestra el nombre de los clientes de "Madrid" junto con la suma total de sus pedidos.

Select c.nombre, sum(p.importe) as SumaImporteMadrid

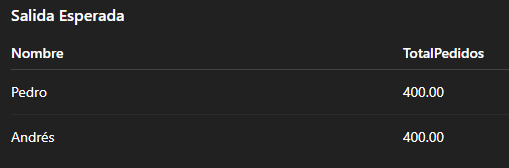
From clientes c

Inner join pedidos p

On c.ID = p.clienteID

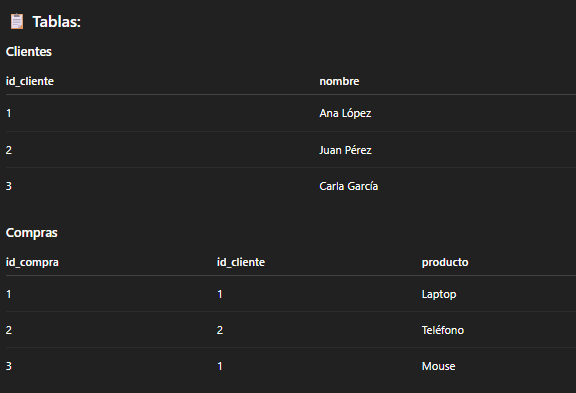
Where c.ciudad = ‘Madrid’

Group by c.nombre;



# Basics joins

## Ejercicio 1

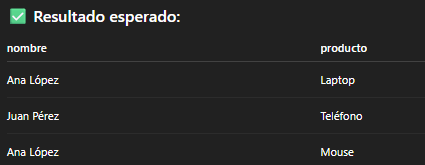


* La llave primaria de la tabla clientes es id\_cliente
* La llave primaria de la tabla compras es id\_compras
* La llave foránea de la tabla compras es id\_cliente y esta llave es la que relaciona a ambas tablas
* La relación que tienen ambas tablas es de uno a muchos. Ya que un cliente puede hacer muchas compras, pero una compra (id de compra) solo le puede pertenecer a un cliente (aunque sea el mismo producto, tiene id diferente ya uqe es una unidad diferente)
* La tabla padre es la tabla clientes y la tabla hija es la tabla compras. En un esquema de ven la tabla de la izquierda (A) seria la tabla clientes y la tabla de la derecha (B) sería la tabla seria la tabla compras
* La base de datos relacional (en este caso el conjunto de tablas “clientes” y “compras”) ya está normalizada (ya se hicieron las 3 formas normales) y lista para hacer consultas

🔹 Ejercicio 1: Nivel Fácil

🎯 Objetivo:

Mostrar los nombres de los clientes y los productos que han comprado.



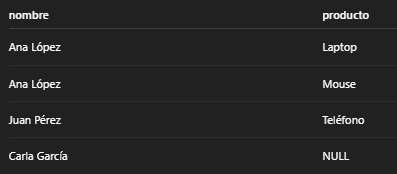
Select cl.nombre, co.producto

From clientes cl

Left join compras co

On cl.id\_cliente= co.id\_cliente;

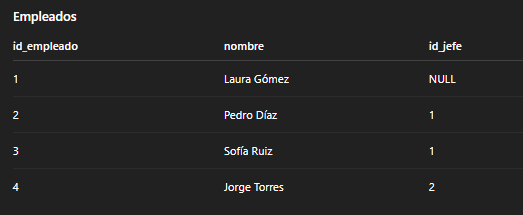
El resultado esperado en la imagen anterior es el resultado cuando se usa el inerio o la intersección para las dos tablas. Yo utilicé el LED joven que trae todos los registros de la tabla a aunque no estén en la tabla B, es decir:

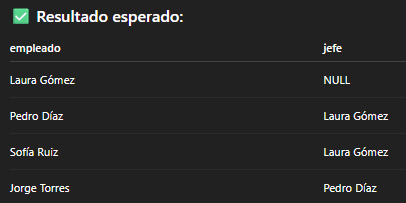


🔸 Ejercicio 2: Nivel Intermedio

🎯 Objetivo:

Mostrar todos los empleados y, si tienen un jefe, mostrar también el nombre del jefe.





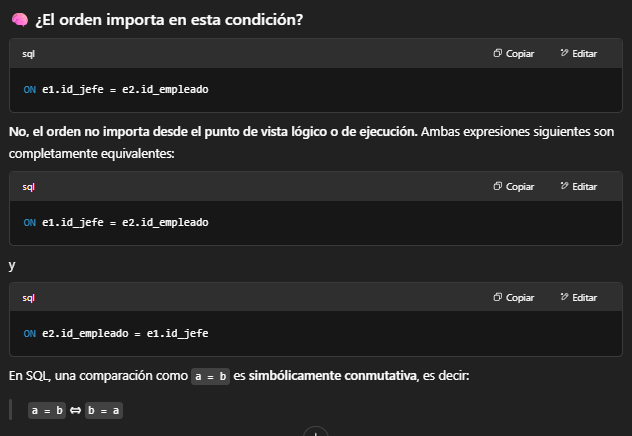
"La estrategia para resolver este ejercicio es duplicar artificialmente la tabla y trabajarla como una segunda tabla para poder hacer las uniones correspondientes." Es decir haremos una Unión automática self join

Select e1.nombre as empleado, e2.nombre as jefe

From empleados e1

Left join empleados e2

On e1.id\_jefe = e2.id\_empleado;



## Ejercicio 2

Fácil

Enunciado:

Obtén los nombres de los empleados y el nombre del departamento al que pertenecen.

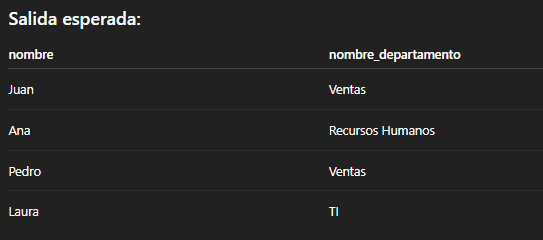


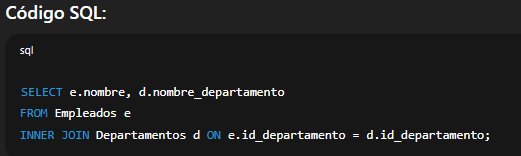
Select e.nombre, d.nombre\_departamento

From empleados e

Inner join departamentos d

On e.id\_departamento = d.id\_departamento;





Intermedio



Enunciado:

Queremos obtener una lista de **todos los clientes**, sin importar si han hecho pedidos o no.

* Si el cliente **hizo pedidos**, se mostrará su nombre y el total del pedido.
* Si el cliente **no ha hecho pedidos**, se mostrará su nombre y el valor NULL en la columna de total.

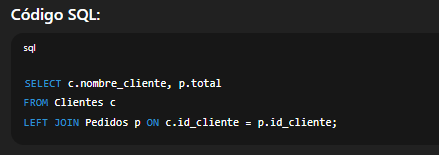
Select c.nombre\_cliente, p.total

From clientes c

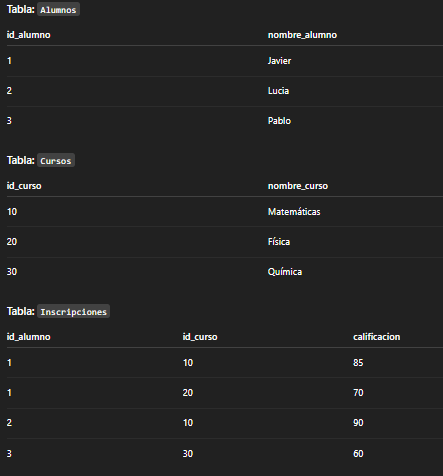
Left join pedidos p

On c.id\_clientes = p.id\_cliente;





Ejercicio difícil



Enunciado:

Muestra el nombre de los alumnos, el curso y la calificación, **solo de aquellos que aprobaron (calificación >= 70)**, incluyendo su curso. Si un alumno no aprobó ningún curso, no debe aparecer.

Select a.nombre, c.nombre\_curso, i.calificacion

From inscripciones i

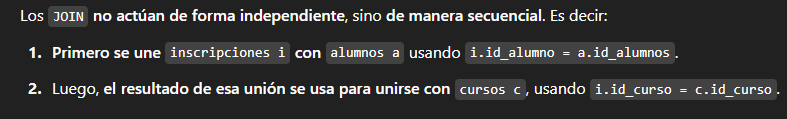
Inner join alumnos a

On i.id\_alumno = a.id\_alumnos

Inner join cursos c

On i.id\_curso = c.id\_curso

Where i.calificacion >= 70;



la tabla hija tiene **2 coincidencias** del ID, se generarán **2 filas** en el JOIN.



## Ejercicio 3

CREATE TABLE empleados (

id\_empleado INT PRIMARY KEY,

nombre VARCHAR(50)

);

CREATE TABLE departamentos (

id\_departamento INT PRIMARY KEY,

nombre\_departamento VARCHAR(50)

);

CREATE TABLE empleado\_departamento (

id\_empleado INT,

id\_departamento INT

);

CREATE TABLE proyectos (

id\_proyecto INT PRIMARY KEY,

nombre\_proyecto VARCHAR(50)

);

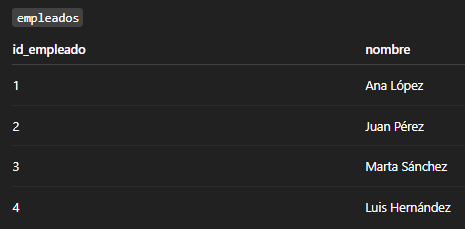
CREATE TABLE empleado\_proyecto (

id\_empleado INT,

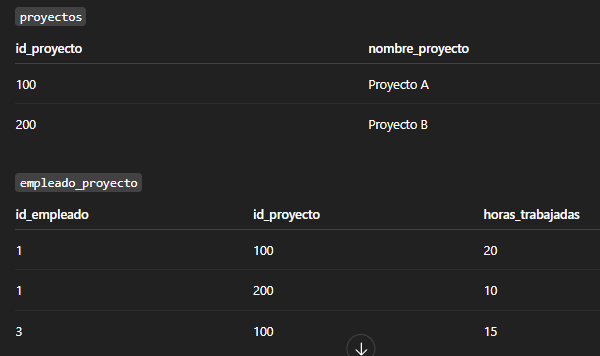
id\_proyecto INT,

horas\_trabajadas INT

);







 Mostrar todos los empleados, aunque no tengan departamento o proyecto asignado.

 Si el empleado no tiene proyecto, dejar el nombre del proyecto y horas como NULL.

* empleado\_departamento es la tabla que relaciona a la tabla empleados y departamentos mediante una llave foradea para cada tabla (id\_empleado, id\_departamento)
* empleado\_proyecto es la tabla mediante la cual se relaciona la tabla empleados y proyectos mendiante la llave foránea para cada tabla (id\_empleado, id\_proyecto)

SELECT

e.nombre AS nombre\_empleado,

d.nombre\_departamento,

p.nombre\_proyecto,

epj.horas\_trabajadas

FROM empleados e

LEFT JOIN empleado\_departamento ed

ON e.id\_empleado = ed.id\_empleado

LEFT JOIN departamentos d

ON ed.id\_departamento = d.id\_departamento

LEFT JOIN empleado\_proyecto epj

ON e.id\_empleado = epj.id\_empleado

LEFT JOIN proyectos p

ON epj.id\_proyecto = p.id\_proyecto

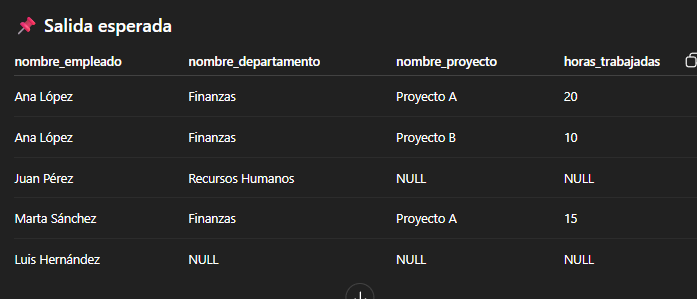
ORDER BY e.id\_empleado, p.id\_proyecto;

Una vez que se ejecuta el primer left join, el siguiente lelf join se ejecuta a partir del estado de la tabla actual (es decir, cuando el primer left join ya actuó)

En otras palabras:

Cada LEFT JOIN se aplica **encima del resultado actual** de la consulta, que incluye ya los efectos del LEFT JOIN anterior.

Se recomienda ejecutar la consulta en cada join, antes de agregar el siguiente join para saber cómo el siguiente join ha´ra la unión con el resultado del join previo



# Funciones de agregación básicas

## Ejercicio 1

🔹 Ejercicio 1– Nivel **Fácil**

Escenario

Dispones de una tabla llamada **empleados** con información sobre el salario y departamento de varios trabajadores.

**Tarea:**  
Obtén el **salario promedio por departamento** y ordénalos de **mayor a menor**.



Obtén el **salario promedio por departamento** y ordénalos de **mayor a menor**.

Select departamento, avg(salario) as promedio

From empleados

Group by departamento

Order by promedio desc;

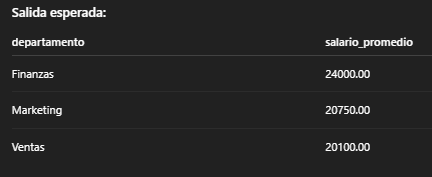
O bien:

Select departamento, avg(salario) as promedio

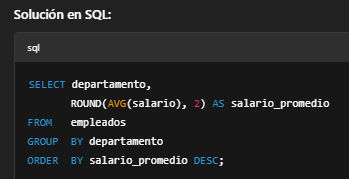
From empleados

Group by departamento

Order by avg(salario) desc;



Si queremos redondear el promedio simplemente tenemos que aplicarle una función a la función del promedio como se ve en seguida:



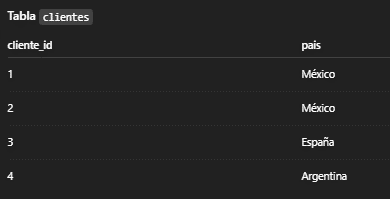
🔹 Ejercicio – Nivel **Intermedio \***

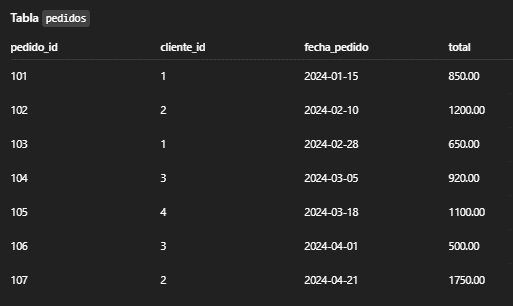
Escenario

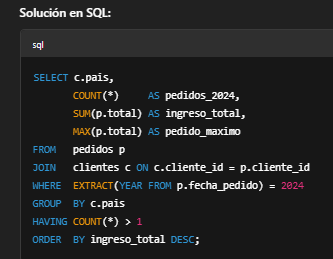
Tienes dos tablas: una de **clientes** con su país de origen, y otra de **pedidos** con los montos de sus compras.

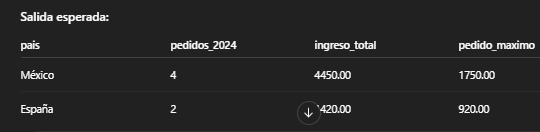
**Tarea:**  
Para el año **2024**, muestra por país:

* Número de pedidos
* Ingreso total
* El valor del pedido más alto  
  Incluye **solo los países con más de 1 pedido** y ordénalos por **ingreso total descendente**.









## Ejercicio 2

CREATE TABLE ventas (

id\_venta INT,

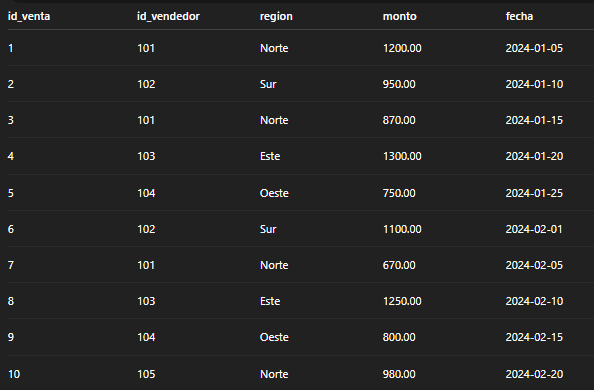
id\_vendedor INT,

region VARCHAR(20),

monto DECIMAL(10,2),

fecha DATE

);



Calcula el **total de ventas, el promedio de ventas y la venta máxima** por región, pero **solo incluye las regiones cuyo total de ventas supere los $2,500**.

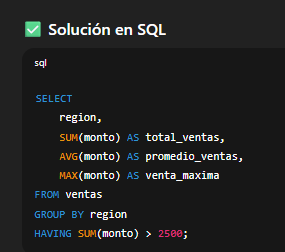
Select region, sum(monto) as venta\_total, avg(monto) as promedio\_venta, max(monto) as maximo\_venta

From ventas

Group by region

Having sum(monto) > 2500;



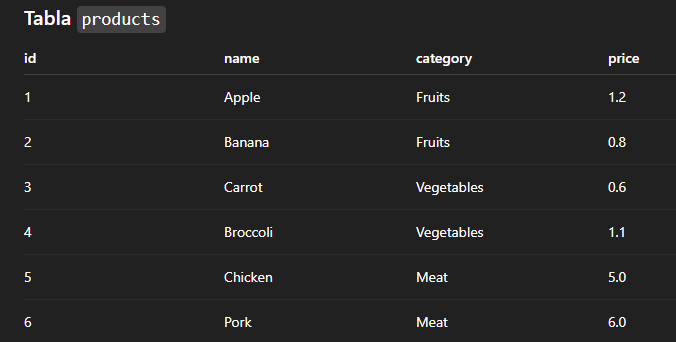


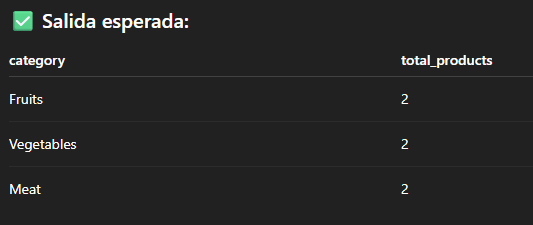
# Clasificación y agrupación

## Ejercicio 1

🟢 Nivel Fácil: Clasificación básica por categoría

**Enunciado:**  
A partir de la tabla products, clasifica los productos por category y muestra cuántos productos hay en cada una.





Select category, count(\*) as total\_products

From products

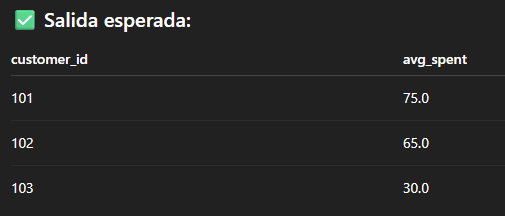
Group by category;

Nivel Difícil: Agrupación con clasificación condicional

🟡 Nivel Intermedio: Agrupación y promedio

**Enunciado:**  
De la tabla orders, calcula el gasto promedio por cada cliente.





Select customer\_id, avg(amount) as gasto\_promedio

From orders

Group by customer\_id;

Ejercicio difícil

**Enunciado:**  
Usa la tabla employees para contar cuántos empleados hay en cada nivel de antigüedad:

* 'Junior' si years\_of\_experience < 3
* 'Mid' si entre 3 y 6 años
* 'Senior' si más de 6 años



SELECT

CASE

WHEN years\_of\_experience < 3 THEN 'Junior'

WHEN years\_of\_experience BETWEEN 3 AND 6 THEN 'Mid'

ELSE 'Senior'

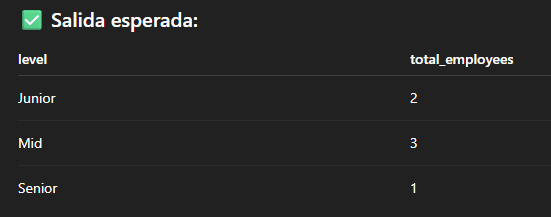
END AS level,

COUNT(\*) AS total\_employees

FROM employees

GROUP BY level;

1. **SELECT CASE ... END AS level**  
   Clasifica a cada empleado según sus años de experiencia:
   * < 3: 'Junior'
   * 3–6: 'Mid'
   * > 6: 'Senior'
2. **COUNT(\*) AS total\_employees**  
   Cuenta cuántos empleados hay en cada nivel.
3. **FROM employees**  
   Toma los datos de la tabla employees.
4. **GROUP BY level**  
   Agrupa por el nivel (Junior, Mid, Senior) para contar por categoría.



# Selección avanzada y uniones

# Subconsultas

# Funciones de cadena avanzadas

## Ejercicio 1