# Formato fechas:

‘año-mes-dia’. Ejemplo: ‘2005-10-21’

# Borrar datos:

****Borra un tipo de producto**** de la tabla TiposProducto que no esté relacionado con ningún producto en Productos.

**DELETE** **FROM** tiposproducto

**WHERE** id **NOT** **IN** (

**SELECT** **DISTINCT** tipoProductoId **FROM** Productos

);

# Comillas

Hay que usar comillas simples ‘ ’ , no comillas dobles “ ”

# Meter id al crear tabla

Puede pasar que al hacer CREATE TABLE se me olvide poner al atributo id, de la siguiente manera:  
**CREATE** **TABLE** tabla(

id **INT** **PRIMARY** **KEY** **AUTO\_INCREMENT**,

. . .

### **🔹 ¿Por qué **no** usar WHERE en lugar de ON al hacer un JOIN?**

#### 🔧 1. ****Semántica y legibilidad:****

Cuando haces un JOIN, **la cláusula** ON **especifica cómo se unen las tablas**, es decir, cuál es la condición de unión. Es el propósito de ON.

-- CORRECTO

SELECT ...

FROM tabla1

JOIN tabla2 ON tabla1.id = tabla2.id;

Usar WHERE también puede funcionar en algunos casos simples, pero **mezcla** condiciones de unión con filtros, lo que puede causar confusión:

-- FUNCIONA, pero es menos claro

SELECT ...

FROM tabla1, tabla2

WHERE tabla1.id = tabla2.id;

# Uso de ON

ON se usa solo con JOIN para unir tablas

WHERE se usa para filtrar filas después de que las tablas han sido seleccionadas (con o sin JOIN)

# Uso de GROUP BY en consultas

Si uso COUNT( ) necesito el GROUP BY para que el sondeo se haga por cada tipo de lo que sea que estoy buscando, no me vale solo con poner el ON

# Consulta complicada:

*/\**

*Cliente que ha pedido la menor variedad de productos.*

*\*/*

***SELECT*** u.nombre **AS** cliente

**FROM** usuarios u

**JOIN** clientes c **ON** u.id=c.usuarioId

**JOIN** pedidos p **ON** c.id=p.clienteId

**JOIN** lineaspedido lp **ON** p.id=lp.pedidoId

**GROUP** **BY** c.id

**ORDER** **BY** **COUNT**( **DISTINCT** lp.productoId) **ASC**

**LIMIT** 1;

Código comentado:

*-- Seleccionamos el nombre del cliente y la cantidad de productos diferentes que ha pedido*

**SELECT** u.nombre **AS** cliente,

**COUNT**(**DISTINCT** lp.productoId) **AS** numero\_productos\_distintos

*-- De la tabla Clientes (alias c)*

**FROM** clientes **AS** c

*-- Unimos con la tabla Pedidos, conectando cliente → pedidos*

**JOIN** pedidos **AS** p

**ON** c.id = p.clienteId

*-- Unimos con la tabla LineasPedido, conectando pedidos → productos pedidos*

**JOIN** lineaspedido **AS** lp

**ON** p.id = lp.pedidoId

*-- Unimos con la tabla Usuarios, para obtener el nombre del cliente*

**JOIN** usuarios **AS** u

**ON** c.usuarioId = u.id

*-- Agrupamos por cliente (id y nombre) para contar productos por cada cliente*

**GROUP** **BY** c.id, u.nombre

*-- Ordenamos de menor a mayor cantidad de productos distintos*

**ORDER** **BY** numero\_productos\_distintos **ASC**

*-- Nos quedamos con el primero: el que pidió menos variedad*

**LIMIT** 1;

# Uso del left/right join cuando quiero buscar una condición que NO se cumple

JOIN normal lo que hace es devolver las filas de la tabla que coinciden de la manera indicada

LEFT JOIN pone todas las filas de la izquierda y las que no coinciden con la otra tabla las rellena con NULL. Justo esto es lo que nos interesa cuando buscamos una condicion que no se cumple, porque luego esas las filtramos con WHERE ---- IS NULL

(RIGHT JOIN es igual pero con la tabla de la derecha)

# No es un error pero quiero comentarlo porque me parece importante

CREATE OR REPLACE VIEW nombre\_vista AS  
SELECT ……………………..

Esto se hace para crear y guardarme una vista, luego puedo llamarla de la siguiente manera:  
SELECT \* FROM nombre\_vista

# Tampoco es un error pero es una cosa curiosa que no sabia que se podia hacer:

Cuando creo un procedimiento, puedo poner dentro de ese procedimiento distintas consultas SELECT y luego cuando llamo al procedimiento me aparece mas de una tabla

# Diferencia entre IF y CASE:

La estructura IF...ELSE (para controlar el flujo) no se puede usar directamente fuera de un procedimiento almacenado, función, trigger o bloque de código procedural en SQL.  
  
Es decir, siempre que use IF tiene que ser dentro de un PROCEDURE o similar

CASE sí que lo puedo usar fuera de ese contexto por ejemplo para hacer una consulta:

**SELECT** nombre,

**CASE**

**WHEN** puedeVenderseAMenores = **TRUE** **THEN** 'Para todas las edades'

**ELSE** 'Solo mayores de edad'

**END** **AS** restriccion\_edad

**FROM** productos;

### ✅ Cuándo usar CASE y cuándo usar IF

| ¿Qué quieres hacer? | ¿Qué usar? |
| --- | --- |
| Evaluar condiciones y devolver un valor | CASE |
| Ejecutar diferentes acciones según una condición | IF |

# Declarar todas las variables primero

Lo primero que tengo que hacer en un procedimiento es declarar todas las variables

# Distinto de

El simbolo de no es igual, se escribe asi: < >

También se puede usar: !=

# No se pueden usar subconsultas derivadas en bloques de código como triggers, procedimientos, funciones y eventos

En consultas normales como SELECT sí que se puede usar, pero en estos casos no.

# MONTH y YEAR

Son funciones útiles que se pueden usar  
Otra importante:  
DATE\_SUB(fecha, INTERVAL cantidad tipo) es una función de MySQL que **resta una cantidad de tiempo a una fecha**.

**SELECT** **DATEDIFF**('2025-07-10', '2025-07-01'); *-- Resultado: 9*

# LAST\_INSERT\_ID( )

# En los TRIGGER recordar lo que estás insertando

A la hora insertar algo en un trigger recuerda que también puedes tener que meter lo que hay en el NEW

Ejemplo:

**SELECT** **IFNULL**(**SUM**(lp.unidades), 0) **INTO** v\_cantidad\_unidades\_pedido

**FROM** lineaspedido lp

**WHERE** lp.pedidoId = **NEW**.pedidoId;

**SET** v\_cantidad\_unidades\_pedido = v\_cantidad\_unidades\_pedido + **NEW**.unidades; *-- se le suman las unidades nuevas*

# If, elseif, else, then

El THEN se pone tras el IF y el ELSEIF, pero nunca tras ELSE

# Esto es clave para hacer funciones:

DETERMINISTIC: **siempre devuelve el mismo resultado** para un mismo input.

# IFNULL y COALESCE

IFNULL: Se usa para reemplazar valores NULL por otro valor **por defecto**.  
Esto es útil en cálculos para evitar que el resultado final sea NULL.

Ejemplo: IFNULL(valor, 0); (Si el valor es NULL, devuelve 0)

El IFNULL cuando uso COUNT no hace falta pero cuando uso SUM sí.

Sería IFNULL(SUM(lp.unidades \*lp.precio), 0)

COALESCE(expr1, expr2, ..., exprN)  
Devuelve **el primer valor que NO sea NULL** de la lista que le des.

# Solución de diciembre/enero cuando me piden algo del mes anterior (aunque no es del todo necesario ya que el profesor lo hace de forma normal)

WHERE p.fechaRealizacion >= DATE\_FORMAT(DATE\_SUB(CURDATE(), INTERVAL 1 MONTH), '%Y-%m-01')

AND p.fechaRealizacion <= LAST\_DAY(DATE\_SUB(CURDATE(), INTERVAL 1 MONTH))

Forma fácil que es la que usa el profesor pero no soluciona lo de diciembre enero:  
MONTH(p.fechaRealizacion) = MONTH(CURDATE()) - 1 AND YEAR(p.fechaRealizacion) = YEAR(CURDATE())

# Para que me aparezcan todos los clientes

Me pasaba que haciendolo yo a mi manera los solo me aparecia un cliente en la vista porque era el unico que cumplia las restricciones del problema. Para que aparezcan todos los demás he tenido que hacer LEFT JOIN para que no importen los NULL

y tambíen he puesto la condición del WHERE en el ON de un JOIN

Poner condiciones en el WHERE **filtra filas ya combinadas**, mientras que ponerlas en el ON **filtra cómo se combinan** (sin excluir completamente los NULL).

# Hacer que la relacion entre dos tablas sea unica. Por ejemplo que cada usuario solo pueda hacer una reseña de un mismo producto (ya que evidentemente no le va a poner dos reseñas distintas a un mismo producto)

Al crear la tabla, antes de poner las FOREIGN KEY, hay que poner:  
UNIQUE (usuarioId, productoId)

Y luego ya ponemos las FOREIGN KEY tanto de usuarioId como de productoId

Bueno, en realidad da igual ponerlo antes o después

# Cuidado con los punto y coma

No poner punto y coma ( ; ) al final de la declaracion de parametros de un procedimiento porque son errores dificiles de detectar.

# Subconsultas

No tengas miedo de poner subconsultas. Se hace de la siguiente manera:

despues de haber hecho lo que querias,

WHERE c.id IN (SELECT …)

Ojo! Dentro de un IN no se puede hacer una subconsulta con mas de una columna

# Cuándo usar HAVING

HAVING **únicamente cuando quieras filtrar resultados después de aplicar una función de agregación** (SUM, COUNT, AVG, MAX, MIN, etc.).

### ✅ ¿Cuándo se usa ROLLBACK?

Se usa **dentro de una transacción** para **deshacer todos los cambios realizados desde el último** START TRANSACTION.