99

CONSULTAS SQL

SOLUCIONES PROPUESTAS

ÍNDICE DE CONTENIDO

1. SOLUCIONES CICLISMO

DIAGRAMA CONCEPTUAL

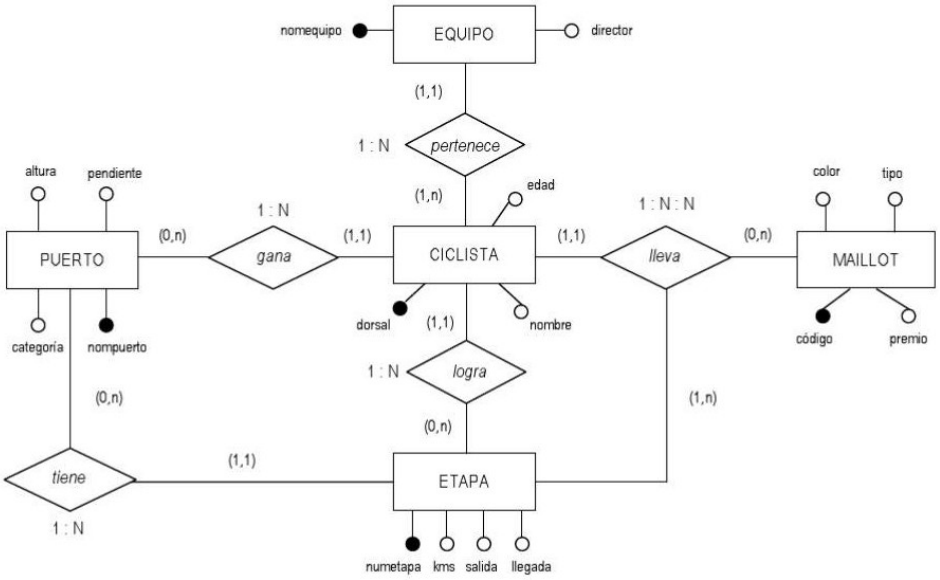
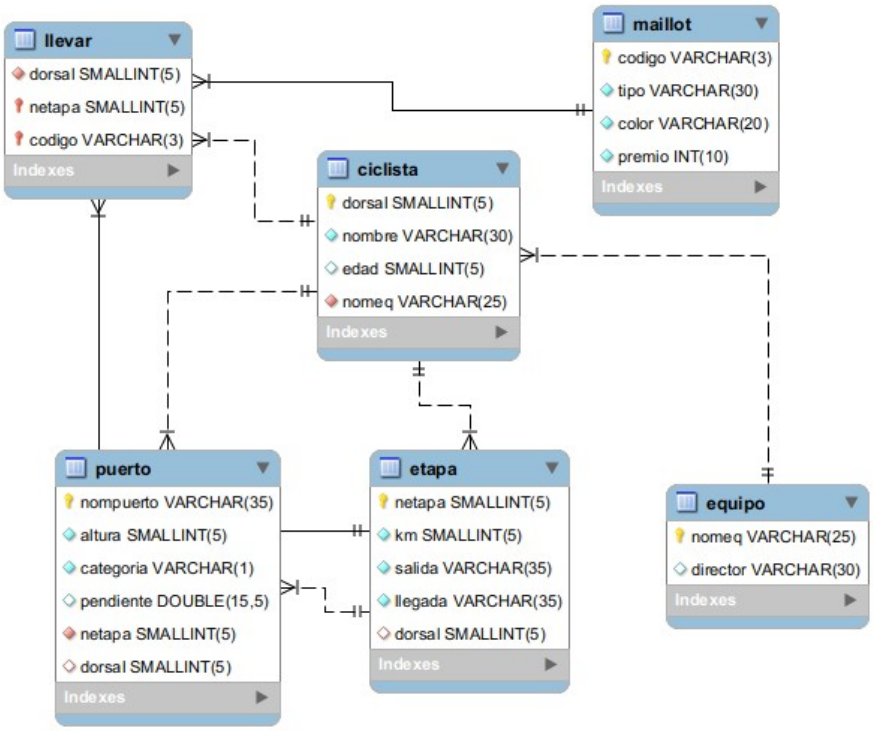


DIAGRAMA FÍSICO



```
 !================================================
!CONSULTAS SOBRE UNA SOLA RELACIÓN
!================================================
!============================
!CICLISMO #1. Obtener el código, el tipo, el color y el premio de todos los maillots que hay.
!============================
SELECT * FROM maillot;
!============================
!CICLISMO #2. Obtener el dorsal y el nombre de los ciclistas cuya edad sea menor o igual que 25 años.
!============================
SELECT dorsal, nombre
FROM ciclista
WHERE edad <= 25;
!============================
!CICLISMO #3. Obtener el nombre y la altura de todos los puertos de categoría 'E' (Especial).
!============================
SELECT nompuerto, altura
FROM puerto
WHERE categoria = 'E';
!============================
!CICLISMO #4. Obtener el valor del atributo netapa de aquellas etapas con salida y llegada en la misma ciudad.
!============================
SELECT netapa
FROM etapa
WHERE salida=llegada;
!============================
!CICLISMO #5. ¿Cuántos ciclistas hay?
!============================
SELECT COUNT(*) FROM ciclista;
```

```
!============================
!CICLISMO #6. ¿Cuántos ciclistas hay con edad superior a 25 años?.
!============================
SELECT COUNT(*)
FROM ciclista
WHERE edad > 25;
!============================
!CICLISMO #7. ¿Cuántos equipos hay?.
!============================
SELECT COUNT(*)
FROM equipo;
!============================
!CICLISMO #8. Obtener la media de edad de los ciclistas.
!============================
SELECT AVG(edad)
FROM ciclista;
!============================
! Ejercicio nº 9. Obtener la altura mínima y máxima de los puertos de montaña.
!============================
SELECT MIN(altura), MAX(altura)
FROM Puerto;
!================================================
!CONSULTAS SOBRE VARIAS TABLAS
!================================================
!============================
!CICLISMO #10. Obtener el nombre y la categoría de los puertos ganados por ciclistas del equipo 'Banesto'.
!============================
SELECT nompuerto, categoria
FROM ciclista c, puerto p
WHERE c.dorsal=p.dorsal AND c.nomeq like 'Banesto';
!============================
SELECT nompuerto, categoria
FROM ciclista c, puerto p
WHERE c.dorsal=p.dorsal AND c.nomeq='Banesto';
```

!==========

!CICLISMO #11. Obtener el nombre de cada puerto indicando el número (netapa) y los kilómetros de la etapa en la que se encuentra el puerto.

!==========

```
SELECT nompuerto, e.netapa, km
FROM puerto p, etapa e
WHERE p.netapa=e.netapa;
```

!==========

!CICLISMO #12. Obtener el nombre y el director de los equipos a los que pertenezca algún ciclista mayor de 33 años.

!==========

```
SELECT DISTINCT e.nomeq, director
FROM equipo e, ciclista c
WHERE e.nomeq=c.nomeq AND c.edad >33;
```

!==========

```
SELECT nomeq, director
FROM equipo
WHERE nomeq IN
    (SELECT nomeq
    FROM ciclista
    WHERE edad>33);
```

!==========

!CICLISMO #13. Obtener el nombre de los ciclistas con el color de cada maillot que hayan llevado.

!==========

```
SELECT DISTINCT c.nombre,m.color
FROM maillot m, llevar l, ciclista c
WHERE m.codigo=l.codigo AND c.dorsal=l.dorsal;
```

!============================

!CICLISMO #14. Obtener pares de nombre de ciclista y número de etapa tal que ese ciclista haya ganado esa etapa habiendo llevado el maillot de color 'Amarillo' al menos una vez.

!============================

```
SELECT DISTINCT c.nombre, e.netapa
FROM ciclista c, etapa e, llevar l, maillot m
WHERE e.dorsal=c.dorsal AND l.dorsal=c.dorsal AND l.codigo=m.codigo
AND m.color like 'Amarillo';
```

!============================

```
SELECT DISTINCT c.nombre, e.netapa
FROM ciclista c, etapa e, llevar l, maillot m
WHERE e.dorsal=c.dorsal AND l.dorsal=c.dorsal AND l.codigo=m.codigo
AND m.color='Amarillo';
```

!============================

!CICLISMO #15. Obtener el valor del atributo netapa de las etapas que no comienzan en la misma ciudad en que acabó la anterior etapa.

!============================

```
SELECT e2.netapa
FROM etapa e1, etapa e2
WHERE e1.llegada <> e2.salida AND e1.netapa + 1 = e2.netapa;
```

!==

!CONSULTAS CON SUBCONSULTAS

!==

!============================

!CICLISMO #16. Obtener el valor del atributo netapa y la ciudad de salida de aquellas etapas que no tengan puertos de montaña.

!============================

```
SELECT netapa, salida
FROM etapa e
WHERE NOT EXISTS
    (SELECT * FROM puerto p
    WHERE p.netapa=e.netapa );

SELECT netapa, salida
FROM etapa
WHERE netapa NOT IN
    (SELECT netapa FROM puerto);
```

!==========

!CICLISMO #17. Obtener la edad media de los ciclistas que han ganado alguna etapa

!==========

```
SELECT AVG(edad)
FROM ciclista
WHERE dorsal IN
    (SELECT dorsal FROM etapa);
```

Este cuenta varias veces los que han ganado varias etapas->

```
SELECT AVG(DISTINCT edad)
FROM ciclista c, etapa e
WHERE c.dorsal= e.dorsal
```

!==========

!CICLISMO #18. Selecciona el nombre de los puertos con una altura superior a la altura media de todos los puertos.

!==========

```
SELECT nompuerto
FROM puerto
WHERE altura>
    (SELECT AVG(altura) FROM puerto);
```

!==========

!CICLISMO #19. Obtener el nombre de la ciudad de salida y de llegada de las etapas donde estén los puertos con mayor pendiente.

!==========

```
SELECT DISTINCT e.salida, e.llegada
FROM etapa e, puerto p
WHERE e.netapa=p.netapa AND
p.pendiente=
    (SELECT MAX(pendiente) FROM puerto );
```

!==========

!CICLISMO #20. Obtener el dorsal y el nombre de los ciclistas que han ganado los puertos de mayor altura.

!==========

```
SELECT DISTINCT c.dorsal, c.nombre
FROM puerto p, ciclista c
WHERE c.dorsal=p.dorsal AND p.altura=
    (SELECT MAX(altura) FROM puerto);
```

```
!============================
!CICLISMO #21. Obtener el nombre del ciclista más joven.
!============================
SELECT nombre
FROM ciclista
WHERE edad =
    ( SELECT MIN(edad) FROM ciclista );

SELECT nombre
FROM ciclista c
WHERE NOT EXISTS
    (SELECT *
    FROM ciclista c1
    WHERE c1.edad < c.edad);
!============================
!CICLISMO #22. Obtener el nombre del ciclista más joven que ha ganado al menos una etapa.
!============================
SELECT DISTINCT c.nombre
FROM ciclista c, etapa e
WHERE c.dorsal=e.dorsal AND
c.edad =
    (SELECT MIN(c2.edad) FROM ciclista c2, etapa e2
    WHERE c2.dorsal=e2.dorsal);
!============================
!CICLISMO #23. Obtener el nombre de los ciclistas que han ganado más de un puerto.
!============================
SELECT nombre
FROM ciclista c
WHERE 1<
    (SELECT COUNT(*) FROM puerto p
    WHERE p.dorsal = c.dorsal );

SELECT DISTINCT nombre
FROM ciclista c, puerto p1, puerto p2
WHERE c.dorsal = p1.dorsal AND c.dorsal=p2.dorsal AND p1.nom_puerto <> p2.nom_puerto
```

```
!================================================
!CONSULTAS CON CUANTIFICACIÓN UNIVERSAL
!================================================
!============================
```

!CICLISMO #24. Obtener el valor del atributo netapa de aquellas etapas tales que todos los puertos que están en ellas tienen más de 700 metros de altura.!

```
============================
SELECT e.netapa
FROM etapa e
WHERE e.netapa IN
    (SELECT netapa FROM puerto)
AND NOT EXISTS
    (SELECT * FROM puerto p WHERE p.altura <=700 AND e.netapa =p.netapa);

SELECT DISTINCT e.netapa
FROM etapa e, puerto p2
WHERE e.netapa=p2.netapa AND
NOT EXISTS
    (SELECT * FROM puerto p WHERE p.altura <=700 AND e.netapa =p.netapa);

SELECT DISTINCT p2.netapa
FROM puerto p2
WHERE NOT EXISTS
    (SELECT * FROM puerto p
    WHERE p.altura <=700 AND p2.netapa =p.netapa);
```

!==========

!CICLISMO #25. Obtener el nombre y el director de los equipos tales que todos sus ciclistas son mayores de 25 años.

!==========

```
SELECT e.nomeq, e.director
FROM Equipo e
WHERE e.nomeq IN
    (SELECT nomeq FROM Ciclista c)
AND NOT EXISTS
    (SELECT * FROM Ciclista c
    WHERE c.edad<26 AND c.nomeq=e.nomeq);

SELECT DISTINCT e.nomeq, e.director
FROM Equipo e, Ciclista C2
WHERE e.nomeq = c2.nomeq
AND NOT EXISTS
    (SELECT * FROM Ciclista c
    WHERE c.edad<26 AND c.nomeq=e.nomeq);
```

!==========

!CICLISMO #26. Obtener el dorsal y el nombre de los ciclistas tales que todas las etapas que han ganado tienen más de 170 km (es decir que sólo han ganado etapas de más de 170 km).

!==========

```
SELECT c.dorsal,c.nombre
FROM ciclista c
WHERE c. dorsal IN
    (SELECT dorsal FROM etapa)
AND NOT EXISTS
    (SELECT * FROM Etapa e2
    WHERE e2.km <=170 AND e2.dorsal= c.dorsal);

SELECT DISTINCT c.dorsal,c.nombre
FROM ciclista c, etapa e3
WHERE c. dorsal= e3.dorsal
AND NOT EXISTS
    (SELECT * FROM Etapa e2
    WHERE e2.km <=170 AND e2.dorsal= c.dorsal);
```

!==========

!CICLISMO #27. Obtener el nombre de los ciclistas que han ganado todos los puertos de una etapa y además han ganado esa misma etapa.

!==========

```
SELECT DISTINCT c.nombre
FROM ciclista c, etapa e
WHERE e.dorsal=c.dorsal AND
NOT EXISTS
    (SELECT * FROM puerto p
    WHERE p.netapa=e.netapa AND c.dorsal <> p.dorsal )
AND EXISTS
    ( SELECT * FROM puerto p
    WHERE p.netapa=e.netapa );
SELECT DISTINCT c.nombre
FROM ciclista c, etapa e, puerto p
WHERE e.dorsal=c.dorsal AND p.netapa=e.netapa
AND NOT EXISTS
    (SELECT * FROM puerto p2
    WHERE p2.netapa=e.netapa AND
    c.dorsal <> p.dorsal );
```

!==========

!CICLISMO #28. Obtener el nombre de los equipos tales que todos sus corredores han llevado algún maillot o han ganado algún puerto. Equipo | Para todo ciclista perteneciente a Equipo -> (Ciclista llevar maillot) o (Ciclista gana puerto)

!==========

```
SELECT e.nomeq
FROM equipo e
WHERE NOT EXISTS
    ( SELECT * FROM ciclista c
    WHERE c.nomeq=e.nomeq
    AND NOT EXISTS
        (SELECT * FROM llevar ll  WHERE c.dorsal=ll.dorsal )
    AND NOT EXISTS
        (SELECT * FROM puerto p WHERE p.dorsal=c.dorsal )
    )
AND EXISTS
    ( SELECT * FROM ciclista c WHERE c.nomeq=e.nomeq);
```

!==========

!CICLISMO #29. Obtener el código y el color de aquellos maillots que sólo han sido llevados por ciclistas de un mismo equipo.

!==========

```
SELECT DISTINCT m.codigo, m.color
FROM maillot m, llevar l, ciclista c
WHERE c.dorsal=l.dorsal AND m.codigo=l.codigo
AND NOT EXISTS
    (SELECT * FROM llevar l2, ciclista c2
    WHERE c2.dorsal=l2.dorsal AND
    c2.nomeq<>c.nomeq AND l2.codigo=l.codigo);
```

!==========

!CICLISMO #30. Obtener el nombre de aquellos equipos tales que sus ciclistas sólo hayan ganado puertos de 1ª categoría.

!==========

```
SELECT e.nomeq
FROM equipo e
WHERE NOT EXISTS
    (SELECT * FROM ciclista c, puerto p
    WHERE c.dorsal = p.dorsal
    AND p.categoria NOT like '1'
    AND c.nomeq=e.nomeq)
AND EXISTS
    ( SELECT * FROM ciclista c2, puerto p2
    WHERE c2.dorsal =p2.dorsal AND c2.nomeq=e.nomeq);
```

Otra: Obtener el nombre de aquellos equipos tales que sus ciclistas sólo hayan ganado puertos de 1ª categoría.

```
SELECT DISTINCT c.nomeq
FROM ciclista c, puerto p
WHERE c.dorsal = p.dorsal
AND NOT EXISTS
(SELECT * FROM puerto p2, ciclista c2
WHERE p2.dorsal = c2.dorsal AND p2.categoria <> '1' AND c2.nomeq = c.nomeq)
```

```
!================================================
!CONSULTAS AGRUPADAS
!================================================
!============================
!CICLISMO #31 Obtener el valor del atributo netapa de aquellas etapas que tienen puertos de montaña indicando cuántos tiene.
!============================
SELECT e.netapa, COUNT(*)
FROM etapa e, puerto p
WHERE e.netapa=p.netapa
GROUP BY e.netapa;

SELECT netapa, COUNT(*)
FROM puerto
GROUP BY netapa;
!============================
!CICLISMO #32. Obtener el nombre de los equipos que tengan ciclistas indicando cuántos tiene cada uno.
!============================
SELECT E.nomeq, COUNT(*)
FROM EQUIPO E left join ciclista c on E.NOMEQ=C.NOMEQ
GROUP BY E.nomeq;

!============================
!CICLISMO #33 Obtener el nombre de todos los equipos indicando cuántos ciclistas tiene cada uno.
!============================
SELECT nomeq, COUNT(*) FROM ciclista
GROUP BY nomeq
UNION
    SELECT nomeq, 0  FROM equipo
    WHERE nomeq NOT IN (SELECT nomeq FROM ciclista);

Otra opción:
SELECT E.nomeq, COUNT(*)
FROM EQUIPO E left join ciclista c on E.NOMEQ=C.NOMEQ
GROUP BY E.nomeq;
```

!==========

!CICLISMO #34 Obtener el director y el nombre de los equipos que tengan más de 3 ciclistas y cuya edad media sea inferior o igual a 30 años.

!==========

```
SELECT e.director, e.nomeq
FROM ciclista c, equipo e
WHERE c.nomeq=e.nomeq
GROUP BY e.director, e.nomeq
HAVING COUNT(*) >3 AND AVG(edad) <= 30;
```

!==========

!CICLISMO #35 Obtener el nombre de los ciclistas que pertenezcan a un equipo que tenga más de cinco corredores y que hayan ganado alguna etapa indicando cuántas etapas ha ganado.

!==========

```
SELECT c.nombre, COUNT(*)
FROM ciclista c, etapa e
WHERE c.dorsal=e.dorsal
AND 5<
    ( SELECT COUNT(*) FROM ciclista c2
    WHERE c2.nomeq=c.nomeq )
GROUP BY c.nombre, c.dorsal;
```

!==========

!CICLISMO #36. Obtener el nombre de los equipos y la edad media de sus ciclistas de aquellos equipos que tengan la media de edad máxima de todos los equipos.

!==========

```
SELECT C.nomeq, AVG(c.edad)
FROM ciclista c
GROUP BY c.nomeq
HAVING AVG(c.edad) >= ALL
    (SELECT AVG(d.edad)  FROM ciclista d  GROUP BY d.nomeq);

SELECT C.nomeq, AVG(c.edad)
FROM ciclista c
GROUP BY c.nomeq
HAVING AVG(c.edad) =
    SELECT MAX(edad)  FROM
        (SELECT AVG(d.edad) as edad FROM ciclista d GROUP BY d.nomeq);
```

!============================

!CICLISMO #37 Obtener el director de los equipos cuyos ciclistas han llevado más días maillots de cualquier tipo. Nota: cada tupla de la relación Llevar indica que un ciclista ha llevado un maillot un día

!============================

```
SELECT e.director
FROM equipo e, ciclista c, llevar l
WHERE e.nomeq = c.nomeq AND c.dorsal = l.dorsal
GROUP BY e.director, e.nomeq
HAVING COUNT(*) >= ALL
    (SELECT COUNT(*)
    FROM ciclista d, llevar m
    WHERE d.dorsal = m.dorsal
    GROUP BY d.nomeq);
```

!==

!CONSULTAS GENERALES

!============================

!CICLISMO #38. Obtener el código y el color del maillot que ha sido llevado por algún ciclista que no ha ganado ninguna etapa.

!============================

```
SELECT DISTINCT m.codigo, m.color
FROM ciclista c, llevar l, maillot m
WHERE c.dorsal=l.dorsal AND m.codigo=l.codigo AND
c.dorsal NOT IN
     (SELECT e.dorsal FROM etapa e);

SELECT DISTINCT m.codigo, m.color
FROM llevar l, maillot m
WHERE m.codigo=l.codigo AND
l.dorsal NOT IN
    (SELECT e.dorsal FROM etapa e);

SELECT DISTINCT m.codigo, m.color
FROM llevar l, maillot m
WHERE m.codigo=l.codigo AND
NOT EXIST  (SELECT e.dorsal FROM etapa e WHERE e.dorsal = l.dorsal);
```

!==========

!CICLISMO #39. Obtener el valor del atributo netapa, la ciudad de salida y la ciudad de llegada de las etapas de más de 190 km. Y que tengan por lo menos dos puertos.

!==========

```
SELECT e.netapa, e.salida, e.llegada
FROM etapa e
WHERE e.km>190 AND 2<=
    (SELECT COUNT(*) FROM puerto p
    WHERE p.netapa=e.netapa );

SELECT DISTINCT e.netapa, e.salida, e.llegada
FROM etapa e, puerto p, puerto p2
WHERE .e.km>190 AND
e.netapa = p.netapa AND p2.netapa=e.netapa AND p.nompuerto <> p2.nompuerto;

SELECT e.netapa, e.salida, e.llegada
FROM etapa e
WHERE e.km>190 AND e.netapa IN
    (SELECT e2.netapa
    FROM etapa e2, puerto p
    WHERE e2.netapa=p.netapa
    GROUP BY e2.netapa
    HAVING COUNT(*) >1);
```

!==========

!CICLISMO #40. Obtener el dorsal y el nombre de los ciclistas que no han llevado todos los maillots que ha llevado el ciclista de dorsal 20

!==========

```
SELECT c.dorsal, c.nombre
FROM ciclista c
WHERE EXISTS
    (SELECT * FROM llevar l
    WHERE l.dorsal=20 AND
    NOT EXISTS
        (SELECT * FROM llevar l2
        WHERE l2.dorsal=c.dorsal AND
        l2.codigo=l.codigo)
    );
```

Tenemos que encontrar un maillot del 20 que no haya llevado el ciclista C. En la subquery obtengo todos los maillots del ciclista C (SELECT * FROM llevar l2 WHERE l2.dorsal=c.dorsal) y digo que no tiene que existir 1 que haya llevado el 20 (l2.codigo = l.codigo)

!==========

!CICLISMO #41. Obtener el dorsal y el nombre de los ciclistas que han llevado al menos un maillot de los que ha llevado el ciclista de dorsal 20.

!==========

```
SELECT DISTINCT c.dorsal, c.nombre
FROM ciclista c, llevar l, llevar l2
WHERE c.dorsal<>20 AND l.dorsal=c.dorsal AND l2.dorsal=20 AND l2.codigo=l.codigo;

SELECT DISTINCT c.dorsal, c.nombre
FROM ciclista c, llevar l
WHERE c.dorsal<>20 AND l.dorsal=c.dorsal
AND l.codigo IN
    (SELECT l2.codigo FROM llevar l2 WHERE l2.dorsal=20);
```

!==========

!CICLISMO #42. Obtener el dorsal y el nombre de los ciclistas que no han llevado ningún maillot de los que ha llevado el ciclista de dorsal 20.

!==========

~~SELECT c.dorsal, c.nombre~~

~~FROM ciclista c~~

~~WHERE NOT EXISTS~~

~~(SELECT * FROM llevar l, llevar l2~~

~~WHERE l.dorsal=c.dorsal AND l2.dorsal=20 AND l2.codigo=l.codigo);~~

Esta MAL. Solo revisa uno a uno y no con todos los maillots del ciclista.

```
SELECT DISTINCT c.dorsal, c.nombre
FROM ciclista c, llevar l
WHERE l.dorsal=c.dorsal AND c.dorsal<>20
AND NOT EXISTS
    ( SELECT * FROM llevar l2
    WHERE l2.dorsal=20 AND
    l2.codigo=l.codigo);
```

!==========

!CICLISMO #43. Obtener el dorsal y el nombre de los ciclistas que han llevado todos los maillots que ha llevado el ciclista de dorsal 20.

!==========

```
    DORSAL     NOMBRE
---------- ------------------------------
         1 Miguel Induráin
1 fila seleccionada.
```

¿Qué NO nos dice el enunciado y SÍ debemos detectar?

1. **Nos piden que comprobemos los maillots que ha llevado un ciclista concreto y que veamos si hay algún ciclista que ha llevado COMO MÍNIMO esos mismos maillots.**
2. **Como no nos piden (ni en el enunciado ni en la captura de la salida) ningún dato de ese maillot no accederemos a la tabla MAILLOT para ahorrar recursos y optimizar tiempos**
3. **Cuando nos pidan los registros que cumpla (o no) las condiciones de otro ciclista, tenemos que excluir ese otro ciclista del resultado final (WHERE C1.dorsal<>20).**

¿Qué haremos?

4. **Buscaremos los ciclistas que NO tienen el dorsal 20, para los que NO EXISTE un maillot (código de maillot) que haya llevado el dorsal 20 y NO lo hayan llevado esos ciclistas.**

```
-- Buscaremos los ciclistas ...
    SELECT C1.dorsal, C1.nombre
    FROM ciclista  C1
-- que NO tienen el dorsal 20
    WHERE C1.dorsal<>20
    AND NOT EXISTS
-- para los que NO EXISTE un maillot (código de maillot)
-- que haya llevado el dorsal 20
--  y que ese maillot NO exista en el conjunto de maillots que han llevado esos ciclistas
        (SELECT LL1.codigo
        FROM llevar LL1
        WHERE LL1.dorsal=20
        AND NOT EXISTS
                (SELECT LL2.codigo
                FROM llevar LL2
                WHERE LL2.dorsal=C1.dorsal
                AND LL2.codigo=LL1.codigo)
        );
```

!==========

!CICLISMO #44. Obtener el dorsal y el nombre de los ciclistas que han llevado exactamente los mismos maillots que ha llevado el ciclista de dorsal 67.

!==========

```
DORSAL     NOMBRE
---------- ------------------------------
0 filas seleccionadas.
```

Si pruebas con el dorsal 67, debería mostrarte solo el dorsal 69 y viceversa. Iremos con el 67...

¿Qué NO nos dice el enunciado y SÍ debemos detectar?

1. **Nos piden que comprobemos los maillots que ha llevado un ciclista concreto y que veamos si hay algún ciclista que ha llevado EXACTAMENTE esos mismos maillots.**
2. **Como no nos piden (ni en el enunciado ni en la captura de la salida) ningún dato de ese maillot no accederemos a la tabla MAILLOT para ahorrar recursos y optimizar tiempos**
3. **Cuando nos pidan los registros que cumpla (o no) las condiciones de otro ciclista, tenemos que excluir ese otro ciclista del resultado final (WHERE C1.dorsal<>67).**

¿Qué haremos?

4. **Buscaremos los ciclistas que NO tienen el dorsal 67, que han llevado TODOS los maillots que ha llevado el maillot 67 y que no han llevado ningún maillot que no haya llevado el dorsal 67.**

Dicho de otra manera:

5. **Buscaremos ciclistas que NO son el 67 para los que no existe un maillot que haya llevado el 67 (y no ellos) y para los que no existe un maillot que hayan llevado ellos (y no el 67) .**

```
    SELECT C.dorsal, C.nombre
-- Buscaremos los ciclistas que NO son el ciclista con dorsal 67
    FROM ciclista C WHERE C .dorsal<>67
--  para los que no existe
--  un maillot (llevar.codigo) que haya llevado el 67
--  y que ese maillot NO exista en el conjunto de maillots que han llevado esos ciclistas
    AND NOT EXISTS
        (       SELECT LL1.codigo FROM llevar LL1 WHERE LL1.dorsal=67
                AND NOT EXISTS
                (SELECT * FROM llevar LL2 WHERE LL2.dorsal=C.dorsal AND LL2.codigo=LL1.codigo)
        )
-- y ADEMÁS
-- para los que no existe
--  un maillot (llevar.codigo) que haya llevado alguno de esos ciclistas
--  que NO exista en el conjunto de maillots que  ha llevado el 67
        AND NOT EXISTS
        (       SELECT * FROM llevar LL3 WHERE LL3.dorsal=C.dorsal
                AND NOT EXISTS
                        (SELECT * FROM llevar LL4
                        WHERE LL4.dorsal=67
                        AND LL3.codigo=LL4.codigo)
                );
```

Alternativamente, podemos cambiar la condición NO EXISTE por NO ESTÁ (NOT IN)

```
    SELECT DISTINCT C.dorsal, C.nombre
    FROM ciclista C INNER JOIN llevar LL1
    ON C.dorsal=LL1.dorsal AND C.dorsal<>67
    WHERE LL1.codigo IN
        (SELECT LL2.codigo  FROM llevar LL2
        WHERE LL2.dorsal=67)
    AND LL1.codigo NOT IN
        (SELECT M.codigo  FROM maillot M
        WHERE M.codigo NOT IN
                (SELECT LL3.codigo  FROM llevar LL3
                WHERE LL3.dorsal=67)
        );
```

!==========

!CICLISMO #45. Obtener el dorsal y el nombre del ciclista que ha llevado durante más kilómetros un mismo maillot e indicar también el color de dicho maillot.

!==========

```
    DORSAL     NOMBRE                         COLOR
---------- ------------------------------ -------
        20 Alfonso Gutiérrez              Verde
1 fila seleccionada.
```

¿Qué DATOS necesitamos?

1. **Necesitamos el nombre del ciclista y el color del maillot por lo que, necesariamente, tendremos que leer de las tablas CICLISTA y MAILLOT y, para saber qué ciclista llevó cada maillot necesitamos la tabla LLEVAR.**

Paso 1: Construimos la SELECT principal que nos servirá como punto de partida.

Podemos usar solo alias para ir más rápido. También ayuda hacer un boceto de qué devolverá:

```
SELECT C.dorsal, C.nombre, M.color
FROM C, LL, M
WHERE C=LL AND LL=M
```

Dorsal	Nombre	Color
25	Pedro García	Verde
31	Ana López	Amarillo
46	Juan García	Verde

Paso 2: Nos piden la suma de los kilómetros en los que ha llevado cada maillot.

Esto requiere añadir la tabla ETAPA en el FROM, añadir una función de agregado (SUM) y agrupar como mínimo por TODOS los campos del SELECT. Además, es conveniente incluir todas las claves primarias de las tablas indicadas en la SELECT por si hubieran varias filas con esos campos repetidos (dos ciclistas llamados "Pedro García"… por ejemplo… o dos maillots con el mismo color "Verde" y diferente código).

Podemos usar solo alias para ir más rápido. También ayuda hacer un boceto de qué devolverá:

```
SELECT C.dorsal, C.nombre, M.color, SUM(E.km)
FROM C, LL, M, E
WHERE C=LL AND LL=M AND M=E
GROUP BY  C.dorsal, C.nombre, M.codigo, M.color
```

Dorsal	Nombre	Color	SUM(km)
25	Pedro García	Verde	20+30=50
31	Ana López	Amarillo	10+50=60
46	**Juan García**	**Verde**	**30+40=70**

Paso 3: Ya podemos ver el resultado que queremos obtener, solo nos queda dar un paso más para obtener el resultado con un mayor mayor que todos en la columna agregada.

Sería muy tentador hacer algo como MAX(SUM(km), pero concatenar funciones de agregado no está permitido en casi ningún SGBD.

En su lugar, tenemos dos opciones:

A) La más rápida y sencilla: usando LIMIT.

El LIMIT no es estándar y solo está disponible en MySQL. El resto de SGBD hay alternativas similares como TOP que se coloca junto al SELECT en POSTGRES.

Podemos usar solo alias para ir más rápido. También ayuda hacer un boceto de qué devolverá:

```
SELECT C.dorsal, C.nombre, M.color, SUM(E.km)
FROM C, LL, M, E
WHERE C=LL AND LL=M AND M=E
GROUP BY  C.dorsal, C.nombre, M.codigo, M.color
ORDER BY SUM(E.km) DESC
LIMIT 1;
```

Dorsal	Nombre	Color	SUM(km)
46	**Juan García**	**Verde**	**30+40=70**

B) La más compleja pero estándar (vale en cualquier SGDB): anidar esta consulta en otra consulta

Obtendremos la fila del resultado anterior que sea >= que TODOS esos mismos resultados.

La subconsulta solo puede devolver un valor para poder usar el cuantificador universal ALL. En este caso, eliminamos los campos del SELECT no agregados y mantenemos el GROUP BY y el FROM para que se pueda calcular la suma de manera esperada.

Algo como:

```
SELECT A,B,C, SUM(W)
FROM/WHERE <varias tablas>
GROUP BY A,B,C, D
HAVING SUM(W) >= ALL (
        SELECT A,B,C, SUM(W)
        FROM <varias tablas>
        GROUP BY A,B,C, D
)
```

```
SELECT C.dorsal, C.nombre, M.color, SUM(E.km)
FROM C, LL, M, E WHERE C=LL AND LL=M AND M=E
GROUP BY  C.dorsal, C.nombre, M.codigo, M.color
HAVING SUM(E2.km) >= ALL (
        SELECT A,B,C, SUM(E2.km)
        FROM <mismas tablas con otros alias>
        GROUP BY <mismos campos>
)
```

Dorsal	Nombre	Color	SUM(km)
46	**Juan García**	**Verde**	**30+40=70**

Paso 3: Solo nos queda adaptar la salida a lo que nos piden, que incluye únicamente el dorsal, el nombre y el color.

A) La más rápida y sencilla: usando LIMIT.

Tenemos que pasar de lo de la izquierda a lo de la derecha.

```
SELECT C.dorsal, C.nombre, M.color, SUM(E.km)
FROM C, LL, M, E
WHERE C=LL AND LL=M AND M=E
GROUP BY  C.dorsal, C.nombre, M.codigo, M.color
ORDER BY SUM(E.km) DESC
LIMIT 1;
```

Dorsal	Nombre	Color	SUM(km)
46	**Juan García**	**Verde**	**30+40=70**

```
SELECT C.dorsal, C.nombre, M.color, ~~SUM(E.km)~~
FROM C, LL, M, E
WHERE C=LL AND LL=M AND M=E
GROUP BY  C.dorsal, C.nombre, M.codigo, M.color
ORDER BY SUM(E.km) DESC
LIMIT 1;
```

Dorsal	Nombre	Color	~~SUM(km)~~
46	**Juan García**	**Verde**	**~~30+40=70~~**

B) La más compleja pero estándar (vale en cualquier SGDB): anidar esta consulta en otra consulta

Tenemos que pasar de lo de la izquierda a lo de la derecha.

Usamos solo los alias para simplificar.

```
SELECT A,B,C, SUM(W)
FROM <varias tablas>
GROUP BY A,B,C, D
HAVING SUM(W) >= ALL (
        SELECT SUM(W)
        FROM <varias tablas>
        GROUP BY A,B,C, D
)
```

Dorsal	Nombre	Color	SUM(km)
46	**Juan García**	**Verde**	**30+40=70**

```
SELECT A,B,C, ~~SUM(W)~~
FROM <varias tablas>
GROUP BY A,B,C, D
HAVING SUM(W) >= ALL (
        SELECT SUM(W)
        FROM <varias tablas>
        GROUP BY A,B,C, D
)
```

Dorsal	Nombre	Color	~~SUM(km)~~
46	**Juan García**	**Verde**	**~~30+40=70~~**

Paso 4: Eliminamos las tablas innecesarias.

A) La más rápida y sencilla: usando LIMIT.

```
SELECT C.nombre, C.dorsal, M.color
FROM ciclista C, maillot M, etapa E, llevar LL
WHERE C.dorsal=LL.dorsal AND LL.codigo=M.codigo AND LL.netapa=E.netapa
GROUP BY C.nombre, C.dorsal, M.color
ORDER BY SUM(E.km) DESC
LIMIT 1;
```

B) La más compleja pero estándar (vale en cualquier SGDB): anidar esta consulta en otra consulta

```
SELECT C.dorsal, C.nombre, M.color
FROM ciclista C, llevar LL, etapa E, maillot M
WHERE E.netapa = LL.netapa AND C.dorsal = LL.dorsal AND M.codigo=LL.codigo
GROUP BY C.dorsal, C.nombre, M.codigo, M.color
HAVING SUM(E.km) >= ALL
        (SELECT SUM(E2.km)
        FROM ~~ciclista C~~, llevar LL2, etapa E2, ~~maillot M~~
        WHERE E2.netapa = LL2.netapa
        GROUP BY LL2.dorsal, LL2.codigo);
```

```
SELECT C.dorsal, C.nombre, M.color
FROM ciclista C, llevar LL, etapa E, maillot M
WHERE E.netapa = LL.netapa AND C.dorsal = LL.dorsal AND M.codigo=LL.codigo
GROUP BY C.dorsal, C.nombre, M.codigo, M.color
HAVING SUM(E.km) >= ALL
        (SELECT SUM(E2.km)
        FROM llevar LL2, etapa E2
        WHERE E2.netapa = LL2.netapa
        GROUP BY LL2.dorsal, LL2.codigo);
```

!==========

!CICLISMO #46.Obtener el dorsal y el nombre de los ciclistas que han llevado tres tipos de maillot menos de los que ha llevado el ciclista de dorsal 1.

!==========

¿Qué NO nos dice el enunciado y SÍ debemos detectar?

1. **Como no nos piden (ni en el enunciado ni en la captura de la salida) ningún dato de ese maillot no accederemos a la tabla MAILLOT para ahorrar recursos y optimizar tiempos**
2. **Cuando nos pidan los registros que cumpla (o no) las condiciones de otro ciclista, tenemos que excluir ese otro ciclista del resultado final (WHERE C1.dorsal<>1).**
3. **La tabla LLEVAR comprende una relación ternaria que incluye tres claves ajenas que pueden repetirse de manera individual. Por tanto, como podemos tener varios maillots (LLEVAR.codigo) para un mismo dorsal y misma etapa, tenemos que usar sí o sí DISTINCT.**

Respecto a usar números enteros en las queries (uno, dos, tres..).. suele ser muy común ponerlos a la izquierda del igual, pero es válido ponerlos a la derecha cambiado de signo y viceversa.

Por ejemplo: SELECT … WHERE 1 = (SELECT COUNT…);
O, también, SELECT … WHERE (SELECT COUNT…) = 1;
Por ejemplo: SELECT … HAVING COUNT(*) = (SELECT COUNT()…) + 10;
O, también, SELECT … HAVING COUNT(*) -10 = (SELECT COUNT()…);

Esta sería una solución:

```
SELECT C.dorsal, C.nombre
FROM ciclista C, llevar LL
WHERE C.dorsal = LL.dorsal
GROUP BY C.nombre, C.dorsal
HAVING COUNT(DISTINCT LL.codigo) +3 =
    (SELECT COUNT(DISTINCT LL2.codigo) FROM llevar LL2 WHERE LL2.dorsal=1);
```

Igual de válida que esta otra:

```
SELECT C.dorsal, C.nombre
FROM ciclista C, llevar LL
WHERE C.dorsal = LL.dorsal
GROUP BY C.nombre, C.dorsal
HAVING COUNT(DISTINCT LL.codigo)  =
    (SELECT COUNT(DISTINCT LL2.codigo) FROM llevar LL2 WHERE LL2.dorsal=1)-3;
```

Importante: Incluye SOLO las tablas necesarias evitando accesos innecesarios a tablas que pueden contener miles de registros y ralentizar nuestra consulta.

!==========

!CICLISMO #47 Obtener el valor del atributo netapa y los km de las etapas que tienen puertos de montaña

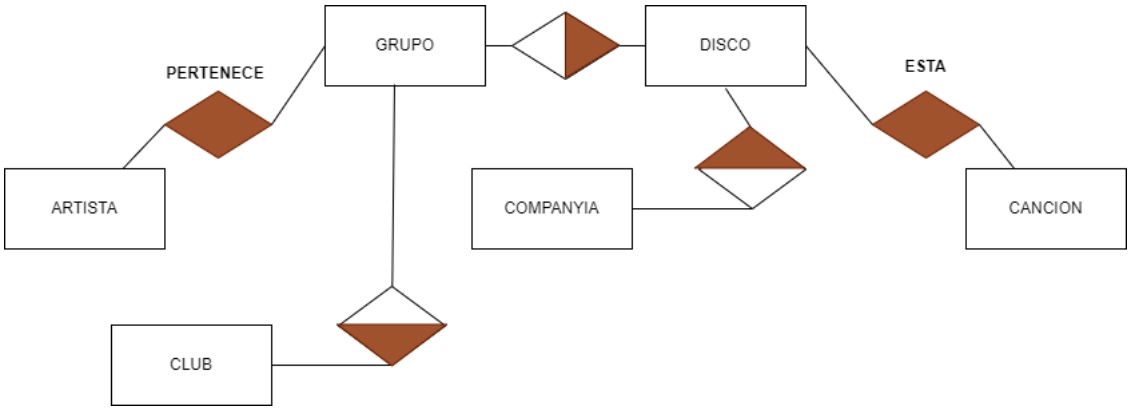
!==========

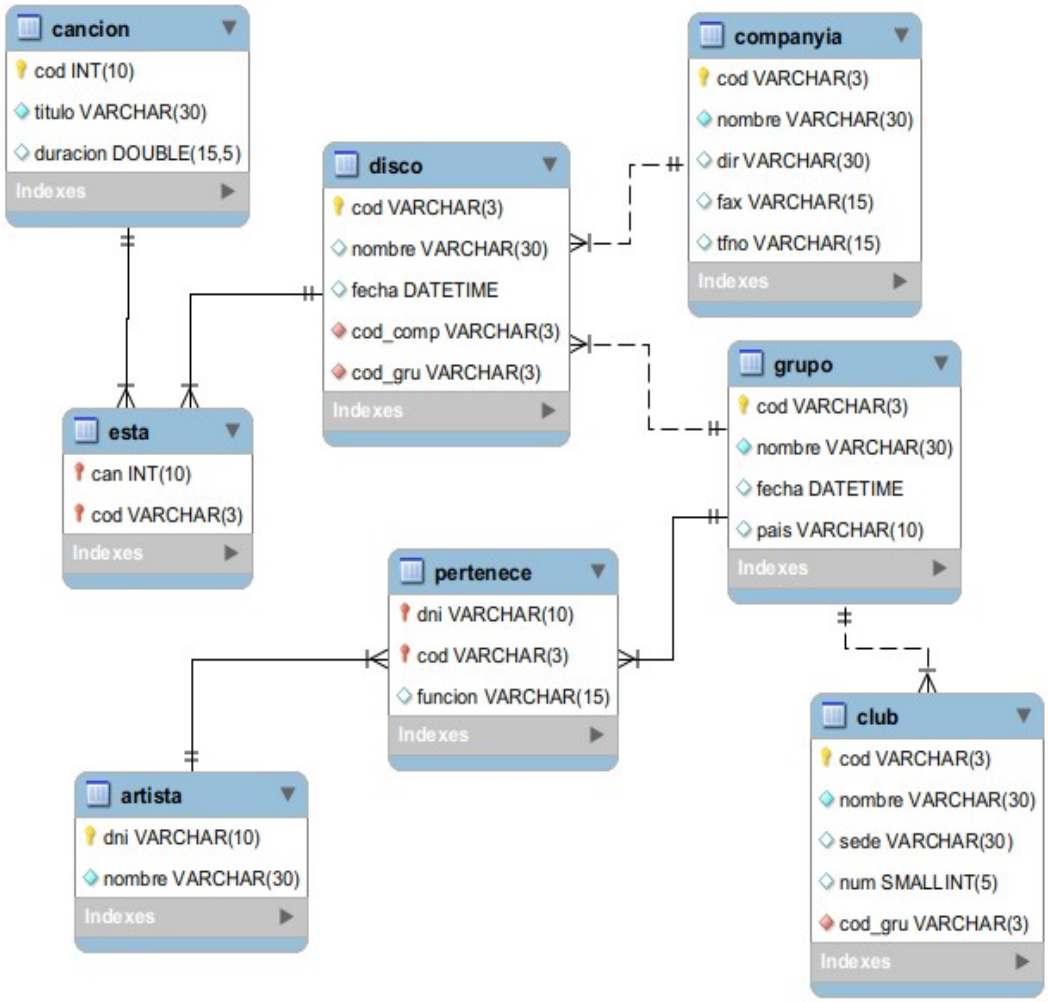
Hay muchas maneras de obtener el mismo resultado.

```
SELECT e.netapa, e.km
FROM etapa e, puerto p
WHERE e.netapa=p.netapa
GROUP BY e.netapa, e.km;

SELECT DISTINCT e.netapa, km
FROM etapa e, puerto p
WHERE e.netapa=p.netapa;

SELECT e.netapa, e.km
FROM etapa e
WHERE e.netapa in (SELECT p.netapa FROM puerto p);

SELECT E.netapa, E.km
FROM etapa E
WHERE EXISTS
    (SELECT P.nompuerto
    FROM puerto P
    WHERE P.netapa=E.netapa);
```

Importante: Recuerda que el DISTINCT, al igual que el *, pueden ser la mejor opción en muchos casos, pero no los uses a la ligera, ya que suelen ser síntoma de MALA PRAXIS.

2. SOLUCIONES MÚSICA





```
!================================================
!CONSULTAS SOBRE UNA SOLA RELACIÓN
!================================================
!============================
!MÚSICA #1. ¿Cuántos discos hay?
!============================
SELECT COUNT(*) FROM disco ;
!============================
!MÚSICA #2. Selecciona el nombre de los grupos que no sean de España.
!============================
SELECT nombre FROM grupo
WHERE pais <> 'España' ;
!============================
!MÚSICA #3. Obtener el título de las canciones con más de 5 minutos de duración.
!============================
SELECT titulo FROM cancion
WHERE duracion > 5.0 ;
!============================
```

!MÚSICA #4. Según los datos en la base de datos, obtener la lista de las distintas funciones que se pueden realizar en un grupo.

```
!============================
SELECT DISTINCT funcion FROM pertenece ;
!============================
!MÚSICA #5. Selecciona el nombre y la sede de los clubes de fans con más de 500 socios.
!============================
SELECT nombre,sede FROM club
WHERE num>500 ;
```

```
!================================================
!Consultas con varias tablas
!================================================
!============================
!MÚSICA #6. Obtener el nombre y la sede de cada club de fans de grupos de España así como el nombre del grupo al que admiran.
!============================
SELECT C.nombre, C.sede, G.nombre FROM club C, grupo G
WHERE C.cod_gru = G.cod AND G.pais = 'España' ;
!============================
!MÚSICA #7. Obtener el nombre de los artistas que pertenezcan a un grupo de España.
!============================
SELECT DISTINCT A.nombre
FROM artista A, pertenece P, grupo G
WHERE P.dni = A.dni AND P.cod = G.cod AND G.pais = 'España';
!============================
!MÚSICA #8. Obtener el nombre de los discos que contienen alguna canción que dure más de 5 minutos.
!============================
SELECT DISTINCT D.nombre
FROM disco D, cancion C, esta E
WHERE E.can = C.cod AND E.cod = D.cod AND C.duracion > 5.00;

SELECT D.nombre FROM disco D
WHERE D.cod in (SELECT e.cod
FROM cancion C, esta E
WHERE C.duracion > 5.00 AND E.can = C.cod) ;
!============================
!MÚSICA #9. Obtener los nombres de las canciones que dan nombre al disco en el que aparecen.
!============================
SELECT DISTINCT C.titulo
FROM disco D, cancion C, esta E
WHERE D.nombre = C.titulo AND E.can = C.cod AND E.cod = D.cod ;
```

```
!============================
!MÚSICA #10. Obtener los nombres de compañías y direcciones postales de aquellas compañías que han grabado algún disco que empiece por 'A'.
!============================
SELECT DISTINCT C.nombre, C.dir FROM disco D, companyia C
WHERE D.cod_comp = C.cod AND D.nombre LIKE 'A%';
!================================================
!Consultas con subconsultas
!================================================
!============================
!MÚSICA #11 . Obtener el nombre de los discos del grupo más viejo.
!============================
SELECT d.nombre FROM disco d, grupo g
WHERE g.cod=d.cod_gru AND g.fecha=
      ( SELECT MIN(fecha) FROM grupo ) ;
!============================
!MÚSICA #12. Obtener el nombre de los discos grabados por grupos con club de fans con más de 5000 personas.
!============================
SELECT d.nombre FROM disco d, grupo g WHERE g.cod=d.cod_gru AND g.cod IN
( SELECT f.cod_gru FROM club f WHERE f.NUM>5000) ;

SELECT d.nombre FROM disco d WHERE d.cod_gru IN
( SELECT f.cod_gru FROM club f WHERE f.NUM>5000) ;

SELECT DISTINCT(d.nombre)
FROM disco d, club cl
WHERE d.cod_grup = cl.cod_gru AND cl.NUM > 5000 ;

SELECT DISTINCT(d.nombre)
FROM disco d, grupo g, club cl
WHERE d.cod_grup = g.cod AND g.cod = cl.cod_gru AND cl.NUM > 5000 ;
```

```
!============================
```

!MÚSICA #13. Obtener el nombre de los clubes con mayor número de fans indicando ese número.

```
!============================
SELECT nombre, num FROM club
WHERE num =
    (SELECT MAX(C.NUM) FROM club C) ;
!============================
```

!MÚSICA #14. Obtener el título de las canciones de mayor duración indicando la duración.

```
!============================
SELECT C.titulo, C.duracion FROM cancion C
WHERE C.duracion =
    (SELECT MAX(D.duracion) FROM cancion D) ;
!================================================
! Consultas con cuantificador universal
!================================================
!============================
```

!MÚSICA #15. Obtener el nombre de las compañías discográficas que no han trabajado con grupos españoles.

```
!============================
SELECT C.nombre FROM companyia C
WHERE NOT EXISTS
    (SELECT * FROM disco D, grupo G
    WHERE D.cod_gru = G.cod AND G.pais = 'España' AND C.cod = D.cod_comp) ;
```

!==========

!MÚSICA #16. Obtener el nombre de las compañías discográficas que sólo han trabajado con grupos españoles.

!==========

```
SELECT C.nombre FROM companyia C WHERE
NOT EXISTS
    (SELECT *
    FROM disco D, grupo G
    WHERE D.cod_gru = G.cod AND G.pais <> 'España' AND C.cod = D.cod_comp)
    AND EXISTS
        (SELECT *
        FROM disco D, grupo G
        WHERE D.cod_gru = G.cod AND G.pais = 'España' AND C.cod = D.cod_comp) ;

SELECT DISTINCT C.nombre
FROM companyia C, disco D WHERE
c.cod = d.cod_comp AND NOT EXISTS
    (SELECT *
    FROM disco D2, grupo G
    WHERE D2.cod_gru = G.cod AND G.pais <> 'España' AND C.cod = D2.cod_comp)
```

!==========

!MÚSICA #17. Obtener el nombre y la dirección de aquellas compañías discográficas que han grabado todos los discos de algún grupo.

!==========

```
SELECT DISTINCT C.nombre, C.dir
FROM companyia C, disco D
WHERE D.cod_comp = C.cod AND NOT EXISTS
    (SELECT * FROM disco E
    WHERE E.cod_gru = D.cod_gru AND E.cod_comp <>
    D.cod_comp);
```

```
!================================================
! Consultas agrupadas
!================================================
!============================
!MÚSICA #18. Obtener el nombre de los grupos que sean de España y la suma de sus fans.
!============================
SELECT g.nombre, SUM(cl.NUM)
FROM grupo g, club cl
WHERE cl.cod_gru=g.cod AND g.pais like 'España'
GROUP BY g.cod, g.nombre ;

SELECT g.nombre, SUM(cl.NUM)
FROM grupo g, club cl
WHERE cl.cod_gru=g.cod AND g.pais = 'España'
GROUP BY g.cod, g.nombre ;
!============================
!MÚSICA #19 Obtener para cada grupo con más de dos componentes el nombre y el número de componentes del grupo.
!============================
SELECT G.nombre, COUNT(P.dni)
FROM grupo G, pertenece P
WHERE P.cod = G.cod
GROUP BY G.cod ,G.nombre
HAVING COUNT(P.dni) > 2 ;

SELECT G.nombre, COUNT(*)
FROM grupo G, pertenece P
WHERE P.cod = G.cod
GROUP BY G.cod ,G.nombre
HAVING COUNT(*) > 2 ;
```

```
!============================
!MÚSICA #20 Obtener el número de discos de cada grupo.
!============================
SELECT g.nombre, COUNT(d.cod)
FROM disco d, grupo g
WHERE g.cod=d.cod_gru
GROUP BY g.cod, g.nombre ;

Esta no mostraría los grupos que tienen 0 discos.
SELECT g.nombre, COUNT(d.cod)
FROM grupo g LEFT JOIN disco d ON g.cod=d.cod_gru
GROUP BY g.cod, g.nombre ;

!================================================
! Otras Consultas (union, join)
!================================================
!============================
!MÚSICA #21. Obtener el número de canciones que ha grabado cada compañía discográfica y su dirección.
!============================
SELECT C.nombre, COUNT (DISTINCT E.can), C.dir
FROM companyia C, disco D, esta E
WHERE C.cod = D.cod_comp AND E.cod = D.cod
GROUP BY C.cod, C.nombre, C.dir
UNION
    SELECT C.nombre, 0, C.dir FROM companyia C
    WHERE NOT EXISTS
        (SELECT E.can FROM disco D, esta E
        WHERE C.cod = D.cod_comp AND E.cod = D.cod) ;

SELECT C.nombre, COUNT (DISTINCT E.can), C.dir
FROM companyia C LEFT JOIN disco D ON C.cod = D.cod_comp
LEFT JOIN esta E ON E.cod = D.cod
GROUP BY C.cod, C.nombre, C.dir

!================================================
!Consultas generales
```

!==========

!==========

!MÚSICA #22 Obtener los nombre de los artistas de grupos con clubes de fans de más de 500 personas y que el grupo sea de Inglaterra

!==========

```
SELECT DISTINCT a.nombre
FROM artista a, grupo g, club cl, pertenece p
WHERE g.pais like 'Inglaterra' AND p.cod=g.cod AND a.dni=p.dni AND cl.cod_gru=g.cod AND cl.NUM >500 ;
```

!==========

!MÚSICA #23 Obtener el título de las canciones de todos los discos del grupo U2.

!==========

```
SELECT DISTINCT c.titulo
FROM disco d, cancion c, grupo g, esta e
WHERE c.cod=e.can AND e.cod=d.cod AND g.cod=d.cod_gru AND g.nombre like 'U2' ;
```

!==========

!MÚSICA #24 El dúo dinámico por fin se jubila; para sustituirles se pretende hacer una selección sobre todos los pares de artistas de grupos españoles distintos tales que el primero sea voz y el segundo guitarra. Obtener dicha selección.

!==========

```
SELECT DISTINCT A1.nombre AS Voz, A2.nombre AS Guitarra
FROM artista A1, artista A2, pertenece P1, pertenece P2, grupo G1, grupo G2
WHERE A1.dni <> A2.dni AND P1.dni = A1.dni AND P2.dni = A2.dni AND P1.cod = G1.cod AND
      P2.cod = G2.cod AND G1.cod <> G2.cod AND G1.pais = 'España' AND G2.pais = 'España' AND
      P1.funcion = 'voz' AND P2.funcion = 'guitarra' ;
```

!==========

!MÚSICA #25 Obtener el nombre de los artistas que pertenecen a más de un grupo.

!==========

Solución algo enrevesada, pero válida consistente en usar una tabla de manera reflexiva:

```
SELECT DISTINCT A.nombre
FROM artista A, pertenece P, pertenece Q
WHERE P.dni = A.dni AND Q.dni = A.dni
AND P.cod <> Q.cod;
```

Otra alternativa más sencilla:

```
SELECT A.nombre
FROM artista A
WHERE 1<
        (SELECT COUNT(*)
        FROM pertenece P
        WHERE P.dni = A.dni);
```

Si usamos agrupaciones: **¡CUIDADO! PUEDE HABER VARIOS ARTISTAS CON EL MISMO NOMBRE POR LO QUE DEBO AGRUPAR POR LA CLAVE PRIMARIA**

```
SELECT A.nombre
FROM artista A, pertenece P
WHERE P.dni = A.dni
GROUP BY A.dni, A.nombre
HAVING COUNT(P.cod)>1;
```

```
SELECT A.nombre
FROM artista A INNER JOIN pertenece P ON P.dni = A.dni
GROUP BY A.dni, A.nombre
HAVING COUNT(P.cod)>1;
```

Importante: Recuerda que todos los campos no agregados que aparecen en el SELECT deben estar en el GROUP BY y que, es recomendable incluir siempre alguna PK en el GROUP BY

!==========

!MÚSICA #26 Obtener el título de la canción de mayor duración si es única.

!==========

a) Si entendemos como única la duración:

¡CUIDADO! No me piden que muestre la duración, solo el título

```
SELECT C1.titulo
FROM cancion C1
WHERE 1 = (SELECT COUNT(*) FROM cancion C2  WHERE C2.duracion >= C1.duracion) ;

SELECT C1.titulo
FROM cancion C1
WHERE C1.duracion =
    (SELECT MAX(C2.duracion)  FROM cancion C2 )
AND NOT EXISTS
    (SELECT * FROM cancion C3 WHERE C1.cod <>C3.cod AND C3.duracion = C1.duracion);
```

b) Si entendemos como única la canción (su título):

```
SELECT C1.titulo
FROM cancion C1
WHERE NOT EXISTS
    (SELECT C2.cod FROM cancion C2 WHERE C2.titulo=C1.titulo AND C1.cod<>C2.cod)
 AND NOT EXISTS
    (SELECT C3.cod FROM cancion C3 WHERE C3.cod<>C1.cod AND C1.duracion<C3.duracion);
```

```
+---------------+
| titulo        |
+---------------+
| 7 Deadly Sins |
| Lemon         |
| So Cruel      |
| Zooropa       |
+---------------+
4 rows in set (0.00 sec)
```

!==========

!MÚSICA #27 Obtener el décimo (debe haber solo 9 por encima de él) club con mayor número de fans indicando ese número

!==========

```
SELECT C1.nombre, C1.NUM
FROM club C1
WHERE 9 = (SELECT COUNT(*) FROM club C2 WHERE C2.NUM > C1.NUM) ;

SELECT C1.nombre, C1.NUM FROM club C1
WHERE 10= (SELECT COUNT(*) FROM club C2 WHERE C2.NUM >= C1.NUM) ;
```

!==========

!MÚSICA #28. Obtener el nombre de los artistas que tengan la función de bajo en un único grupo y que además éste tenga más de dos miembros.

!==========

```
SELECT DISTINCT A.nombre
FROM artista A, pertenece P1
WHERE P1.dni = A.dni AND P1.funcion = 'bajo'
AND NOT EXISTS
    (SELECT P2.dni FROM pertenece  P2
    WHERE P2.cod <> P1.cod AND P2.funcion = 'bajo' AND P1.dni = P2.dni)
AND EXISTS
    (SELECT P3.dni FROM pertenece  P3
    WHERE P3.cod = P1.cod AND P3.dni <> P1.dni) ;

SELECT A.nombre
FROM artista A, pertenece P1
WHERE A.dni = P1.dni AND P1.funcion = 'bajo'
AND NOT EXISTS
    (SELECT * FROM pertenece P2
    WHERE A.dni = P2.dni AND P2.funcion = 'bajo' AND P1.cod<>P2.cod)
AND 2 <
    (SELECT COUNT(p3.dni)
    FROM pertenece p3 WHERE p3.cod = P1.cod);
```

```
SELECT A.nombre
FROM artista a , pertenece P1
WHERE a.dni = p1.dni AND p1.funcion = 'bajo'
AND a.dni NOT IN
    (SELECT p2.dni
    FROM pertenece p2
    WHERE p2.funcion = 'bajo' AND p1.cod<>p2.cod)
AND 2 <
    (SELECT COUNT(p3.dni)
    FROM pertenece p3 WHERE p3.cod = p1.cod);

SELECT A.nombre
FROM artista A INNER JOIN pertenece P1
ON A.dni = P1.dni AND P1.funcion = 'bajo'
WHERE  2 <
    (SELECT COUNT(P2.dni)
    FROM pertenece P2 WHERE P2.cod = P1.cod)
GROUP BY A.dni, A.nombre
HAVING COUNT(P1.cod)=1;
```

!==========

!MÚSICA #29 ¿Cuál es la compañía discográfica que más canciones ha grabado?

!==========

```
SELECT C1.nombre, COUNT(DISTINCT E1.can) AS canciones
FROM companyia C1, disco D1, esta E1
WHERE D1.cod_comp = C1.cod AND E1.cod = D1.cod
GROUP BY C1.cod, C1.nombre
HAVING COUNT(DISTINCT E1.can) >= ALL
        (SELECT COUNT(DISTINCT E1.can)
        FROM companyia C1, disco D1, esta E1
        WHERE D1.cod_comp = C1.cod AND E1.cod = D1.cod GROUP BY C1.nombre, C1.cod) ;
```

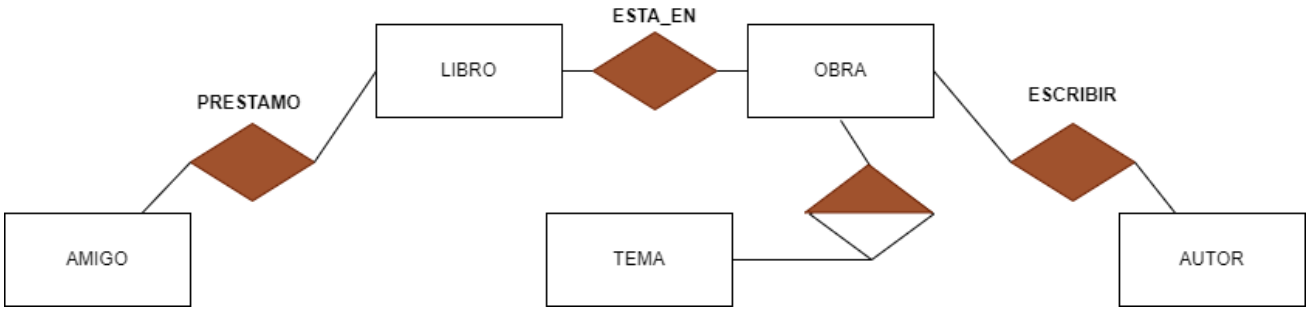
Se podría ahorrar C1 en el HAVING:

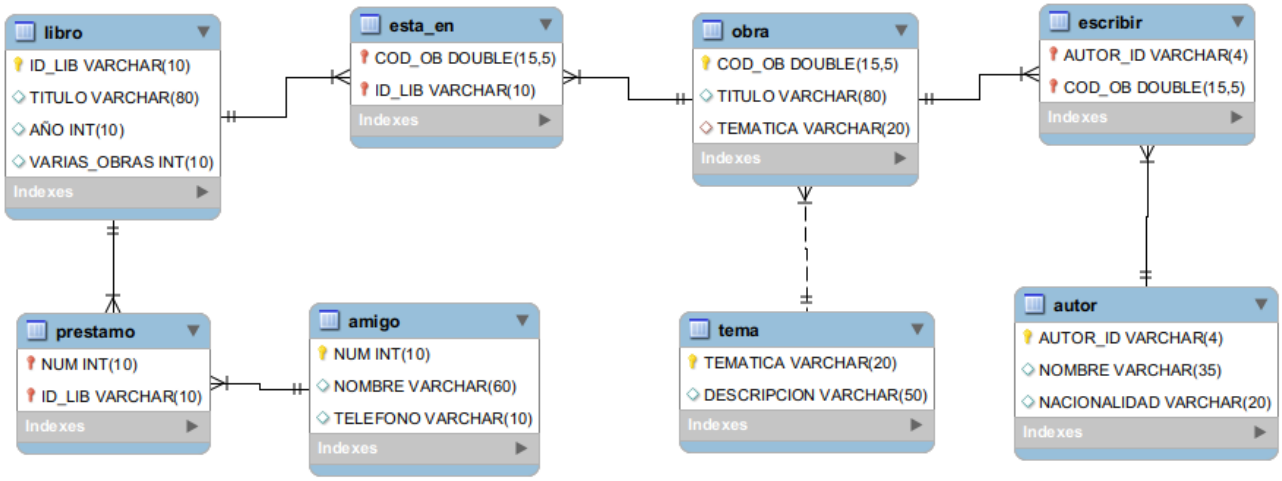
```
SELECT C1.nombre, COUNT(E1.can) AS canciones
FROM companyia C1, disco D1, esta E1
WHERE D1.cod_comp = C1.cod AND E1.cod = D1.cod
GROUP BY C1.cod, C1.nombre
HAVING COUNT(E1.can)=
(
        SELECT MAX(X.cuenta)
        FROM
                (SELECT COUNT(E2.can) as cuenta FROM companyia C2, disco D2, esta E2
                WHERE D2.cod_comp = C2.cod AND E2.cod = D2.cod
                GROUP BY C2.cod, C2.nombre)
                XX
        );
```

También con LIMIT (menos elegante), ya que LIMIT no es estándar y solo sirve en MySQL.

```
SELECT C.nombre, COUNT(E.cod) AS  canciones
FROM companyia C INNER JOIN disco D INNER JOIN esta E
ON D.cod_comp = C.cod AND E.cod = D.cod
GROUP BY C.cod, C.nombre
ORDER BY canciones DESC LIMIT 1;
```

3. SOLUCIONES BIBLIOTECA





```
!============================
!BIBLIOTECA #1 ¿Cuántos libros hay de los que se conozca el año de adquisición?
!============================
SELECT COUNT(*) as lib_año FROM libro
WHERE año is NOT NULL;

SELECT COUNT(año) as lib_año FROM libro;

!============================
```

!BIBLIOTECA #2 ¿Cuántos libros tienen más de una obra? Resolver este ejercicio utilizando el atributo num_obras y sin utilizarlo.

```
!============================
SELECT COUNT(*) as Más_1_ob FROM libro
WHERE num_obras >1;

SELECT COUNT(DISTINCT e.ID_LIB) FROM esta_en E, esta_en E2
WHERE E.ID_LIB=E2.ID_LIB AND E.COD_OB <> E2.COD_OB;

SELECT COUNT(*)
FROM (SELECT COUNT(*)
FROM esta_en GROUP BY ID_LIB
HAVING COUNT(COD_OB)>1);

SELECT COUNT(*) FROM libro l
WHERE (SELECT COUNT(*)
FROM esta_en e
WHERE e.ID_LIB=l.ID_LIB)>1;

Esta está mal porque para cada lib da un cont
SELECT COUNT(*)
FROM esta_en GROUP BY ID_LIB HAVING COUNT(*)>1;
```

```
!============================
!BIBLIOTECA #3 ¿Cuántos autores hay en la base de datos de los que no se tiene ninguna obra?
!============================
SELECT COUNT(*) Sin_obra FROM autor
WHERE AUTOR_ID NOT IN (SELECT AUTOR_ID
FROM escribir);

SELECT COUNT(*) Sin_obra
FROM autor a LEFT JOIN escribir e on a.AUTOR_ID=e.author_id WHERE e.AUTOR_ID is NULL;

!============================
!BIBLIOTECA #4 Obtener el nombre de esos autores.
!============================
SELECT nombre
FROM autor
WHERE AUTOR_ID NOT IN (SELECT AUTOR_ID FROM escribir);

SELECT a.nombre FROM autor a
WHERE NOT EXISTS (SELECT *
FROM escribir e
WHERE a.AUTOR_ID= e.AUTOR_ID);

!============================
!BIBLIOTECA #5 Obtener el título de las obras escritas sólo por un autor si éste es de
nacionalidad “Francesa” indicando también el nombre del autor.
!============================
SELECT titulo, nombre FROM obra, autor
WHERE COD_OB in (SELECT COD_OB
FROM escribir GROUP BY COD_OB HAVING COUNT(*)=1)
AND (COD_OB, AUTOR_ID) in (SELECT COD_OB, AUTOR_ID FROM escribir e) AND nacionalidad
='Francesa';

SELECT titulo, nombre FROM obra o, autor a, escribir e1
WHERE o.COD_OB=e1.COD_OB AND e1.AUTOR_ID=a.AUTOR_ID AND
a.nacionalidad=”Francesa”
AND NOT EXISTS (SELECT * FROM escribir e WHERE e.COD_OB=e1.COD_OB AND
e.author_id<>e1.author_id);
```

!==========

!BIBLIOTECA #6 Obtener el título y el identificador de los libros que tengan título y más de dos obras, indicando el número de obras.

!==========

```
SELECT l.ID_LIB, l.titulo, COUNT(*) FROM libro l, esta_en e
WHERE l.titulo is NOT NULL AND e.id_Lib=l.ID_LIB GROUP BY l.ID_LIB, l.titulo
HAVING COUNT(*) >2;

SELECT ID_LIB, titulo, num_obras FROM libro l
WHERE l.titulo is NOT NULL AND num_obras >2;
```

!==========

!BIBLIOTECA #7 Obtener el nombre de los autores de nacionalidad “Española” que han escrito dos o más obras.

!==========

```
SELECT nombre
FROM autor a
WHERE nacionalidad ='Española' AND
2<= (SELECT COUNT(*) FROM escribir e WHERE a.AUTOR_ID= e.AUTOR_ID);

SELECT nombre FROM autor a
WHERE a.nacionalidad ='Española' AND
(SELECT COUNT(*) FROM escribir e
WHERE a.AUTOR_ID= e.AUTOR_ID) >= 2;
```

!==========

!BIBLIOTECA #8 Obtener el nombre de los autores de nacionalidad “Española” que tienen obras en dos o más libros.

!==========

```
SELECT nombre
FROM autor a
WHERE nacionalidad ='Española' AND
2<= (SELECT COUNT(DISTINCT ID_LIB) FROM escribir e, esta_en ee WHERE a.AUTOR_ID= e.AUTOR_ID AND e.COD_OB=ee.COD_OB);

SELECT nombre FROM autor a
WHERE nacionalidad ='Española' AND
(SELECT COUNT(DISTINCT ID_LIB) FROM escribir e, esta_en ee
WHERE a.AUTOR_ID= e.AUTOR_ID AND e.COD_OB=ee.COD_OB) >=2;

SELECT nombre FROM autor a
WHERE nacionalidad ='Española' AND
2<= (SELECT COUNT(*) FROM escribir e, esta_en ee
WHERE a.AUTOR_ID= e.AUTOR_ID AND e.COD_OB=ee.COD_OB GROUP BY ee.ID_LIB);
```

!BIBLIOTECA #9 Obtener el título y el código de las obras que tengan más de un autor.

```
SELECT COD_OB, titulo FROM obra o
WHERE 1 < (SELECT COUNT(*) FROM escribir e WHERE o.COD_OB=e.COD_OB);

SELECT COD_OB, titulo FROM obra o
WHERE (SELECT COUNT(*) FROM escribir e WHERE o.COD_OB=e.COD_OB) > 1;

SELECT COD_OB, titulo FROM obra o, escribir e WHERE o.COD_OB=e.COD_OB GROUP BY COD_OB, titulo HAVING COUNT(*)>1
```

!==========

!BIBLIOTECA #10 Obtener el título y el identificador de los libros que tengan título y que contengan sólo una obra.

!==========

```
SELECT l.ID_LIB, l.titulo FROM libro l
WHERE titulo is NOT NULL AND
1= (SELECT COUNT(*) FROM esta_en e WHERE l.ID_LIB=e.ID_LIB);

SELECT titulo FROM libro l
WHERE titulo is NOT NULL AND
(SELECT COUNT(*) FROM esta_en e WHERE l.ID_LIB=e.ID_LIB) = 1;
```

!==========

!BIBLIOTECA #11 Como se concluye del resultado de la consulta anterior, los libros con una sola obra no tienen título propio. Asumiendo en este caso que su título es el de la obra que contienen, obtener la lista de todos los títulos de libros que hay en la base de datos tengan las obras que tengan título.

!==========

```
SELECT titulo FROM libro
WHERE titulo is NOT NULL
UNION
SELECT titulo
FROM obra o, esta_en e WHERE o.COD_OB =e.COD_OB
AND 1 = (SELECT COUNT(*) FROM esta_en e1
WHERE e.ID_LIB=e1.ID_LIB);

SELECT titulo as titulo FROM libro
WHERE titulo is NOT NULL
UNION
SELECT o.titulo as titulo
FROM obra o, esta_en e, libro l
WHERE o.COD_OB =e.COD_OB AND l.ID_LIB=e.ID_LIB AND l.titulo is NULL
```

```
!============================
!BIBLIOTECA #12 Obtener el nombre del autor (o autores) que más obras han escrito?
!============================
SELECT nombre
FROM autor a, escribir e WHERE a.AUTOR_ID=e.AUTOR_ID GROUP BY a.AUTOR_ID, nombre
HAVING COUNT(*) >= ALL (SELECT COUNT(*) FROM escribir GROUP BY AUTOR_ID);

!============================
!BIBLIOTECA #13 Obtener la nacionalidad (o nacionalidades) menos frecuentes.
!============================
SELECT nacionalidad
FROM autor
GROUP BY nacionalidad
HAVING COUNT(*) <= ALL (SELECT COUNT(*) FROM autor GROUP BY nacionalidad);
```

!BIBLIOTECA #14 Obtener el nombre de los amigos que han leído alguna obra del autor de identificador 'RUKI'.

```
NOMBRE
------------------------------------------------
Isabel Peiró García
Eloy Prim Gros
2 filas seleccionadas.
```

¿Qué NO nos dice el enunciado y SÍ debemos detectar?

1. **LEÍDA=PRESTADA**
2. **Como no nos piden (ni en el enunciado ni en la captura de la salida) ningún dato concreto del autor no accederemos a la tabla AUTOR para ahorrar recursos y optimizar tiempos. De los amigos sí que nos piden el nombre, por lo que tendremos que acceder a la tabla AMIGO.**
3. **Una estrategia que suele tener buen resultado es navegar el diagrama entidad relación de un extremo a otro enlazando las tablas con sus tablas de cruce (N:N) hasta llegar al extremo que queremos.**
4. **En este caso, si vemos el diagrama ER, observamos que si queremos enlazar AMIGO con AUTOR tenemos que recorrer AMIGO → PRESTAMO → LIBRO → ESTA_EN → OBRA → ESCRIBIR, dejando fuera la tabla AUTOR al no necesitar ningún dato concreto como acabamos de ver (luego veremos como LIBRO y OBRA no las necesitamos tampoco).**
5. **Haremos un "boceto" de la consulta que nos servirá de partida y pintaremos a boli una tabla de una posible salida (nosotros pondremos un ejemplo directo de la BD).**

-- Estrategia enlazado total de cruces extremo a extremo

El "boceto" de lo que queremos sería este:

SELECT AM.NOMBRE, ESC.AUTOR_ID

FROM AM, P, L, EST, O, ESC

WHERE AM=P AND P=L AND L=EST AND EST=O AND O=ESC;

Que se transformaría en esta consulta:

SELECT AM.NOMBRE, ESC.AUTOR_ID

FROM amigo AM, prestamo P, libro L, esta_en EST, obra O, escribir ESC

WHERE AM.NUM=P.NUM AND P.ID_LIB=L.ID_LIB AND L.ID_LIB=EST.ID_LIB AND EST.COD_OB=O.COD_OB AND O.COD_OB=ESC.COD_OB;

nombre	AUTOR_ID
Pepe Pérez Pérez	CAMA
Pepe Pérez Pérez	CAMA
Isabel Peiró García	PANE
Isabel Peiró García	AMMA
Isabel Peiró García	AMMA
Isabel Peiró García	AMMA
Isabel Peiró García	CAMA
Isabel Peiró García	RUKI
Isabel Peiró García	RUKI
Eloy Prim Gros	RUKI
Yolanda Milanés Cuba	BOVI
Yolanda Milanés Cuba	BOVI
Yolanda Milanés Cuba	BOVI
Yolanda Milanés Cuba	BOVI
Yolanda Milanés Cuba	BOVI
Isidro Catalá Ferrer	CAMA

16 rows in set (0,01 sec)

Como puedes ver, necesitamos filtrar esa salida para solo mostrar los amigos que estén relacionados con el autor "RUKI", mediante una condición en el where extra.

Atención: Esta BD tiene atributos en mayúsculas que serán sensibles en sistemas operativos distintos a Windows.

Esta sería una solución casi óptima:

```
SELECT AM.NOMBRE
-- Estrategia de enlazado total de cruces de extremo a extremo
FROM amigo AM, prestamo P, libro L, esta_en EST, obra O, escribir ESC
WHERE AM.NUM=P.NUM AND P.ID_LIB=L.ID_LIB AND L.ID_LIB=EST.ID_LIB AND EST.COD_OB=O.COD_OB AND O.COD_OB=ESC.COD_OB
AND ESC.AUTOR_ID='RUKI';
```

Realmente LIBRO y OBRA tampoco son necesarias, por lo que la consulta ÓPTIMA sería esta:

```
SELECT AM.NOMBRE
FROM amigo AM, prestamo P, esta_en EST, escribir ESC
WHERE AM.NUM=P.NUM AND P.ID_LIB=EST.ID_LIB AND EST.COD_OB=ESC.COD_OB
AND ESC.AUTOR_ID='RUKI';
```

¿Cómo lo harías con JOINS?

```
SELECT AM.NOMBRE
FROM amigo AM
INNER JOIN prestamo  P  INNER JOIN esta_en EST  INNER JOIN escribir ESC
ON AM.NUM=P.NUM AND P.ID_LIB=EST.ID_LIB AND EST.COD_OB=ESC.COD_OB
AND ESC.AUTOR_ID='RUKI';
```

Otra opción es hacer dos subconsultas anidadas. Para gustos, los colores :-)

```
-- AMIGOS ...
SELECT AM.NOMBRE
FROM amigo AM
-- (AMIGOS) PARA LOS QUE EXISTE ALGÚN ...
WHERE EXISTS
    -- AUTOR (QUE HAYA ESCRITO ALGO)...
    (SELECT  E.AUTOR_ID
    FROM escribir E
    -- (AUTOR QUE HAYA ESCRITO ALGO) QUE SE LLAME RUKI PARA EL QUE EXISTE...
    WHERE E.AUTOR_ID='RUKI' AND EXISTS
        -- AL MENOS UNA OBRA...
        (SELECT EE.COD_OB
        FROM esta_en EE, prestamo P
        -- QUE HA LEÍDO (TOMADO PRESTADO) ESE AMIGO...
        WHERE P.NUM=A.NUM AND EE.COD_OB = E.COD_OB AND EE.ID_LIB=P.ID_LIB));
```

!BIBLIOTECA #15 Obtener el nombre de los amigos que han leído todas las obras del autor de identificador 'RUKI'.

```
NOMBRE
--------------------------------------------
Isabel Peiró García

    1 fila seleccionada.
```

Actuaremos de manera similar a la consulta anterior. En este caso, buscamos todos los amigos para los que no exista una obra de RUKI que no hayan leído esos amigos.

OPCION 1: SUBCONSULTAS

Esta sería una solución casi óptima (fíjate que no necesitamos acceder a la tabla OBRA):

```
SELECT AM.NOMBRE
FROM amigo AM
WHERE NOT EXISTS
  (SELECT ESC.COD_OB
  FROM escribir ESC
  WHERE ESC.AUTOR_ID='RUKI'
  AND ESC.COD_OB NOT IN
    (SELECT ESTA.COD_OB
-- Estrategia de enlazado total de cruces de extremo a extremo
     FROM prestamo P, libro L, esta_en ESTA
     WHERE AM.NUM=P.NUM AND P.ID_LIB=L.ID_LIB AND L.ID_LIB=ESTA.ID_LIB)    );
```

Realmente la tabla LIBRO tampoco es necesaria, por lo que la consulta ÓPTIMA sería esta:

```
SELECT AM.NOMBRE
FROM amigo AM
WHERE NOT EXISTS
(   SELECT ESC.COD_OB
  FROM escribir ESC
  WHERE ESC.AUTOR_ID='RUKI'
  AND ESC.COD_OB NOT IN
    (SELECT ESTA.COD_OB
     FROM prestamo P, esta_en ESTA
     WHERE AM.NUM=P.NUM AND P.ID_LIB=ESTA.ID_LIB) );
```

OPCION 2: GROUP BY

Esta sería una solución casi óptima (sobran las tablas LIBRO y OBRA):

```
SELECT AM.NOMBRE
-- Estrategia de enlazado total de cruces de extremo a extremo
FROM amigo AM, prestamo P, libro L, esta_en EST, obra O, escribir ESC
WHERE AM.NUM=P.NUM AND P.ID_LIB=L.ID_LIB AND L.ID_LIB=EST.ID_LIB AND EST.COD_OB=O.COD_OB AND O.COD_OB=ESC.COD_OB AND ESC.AUTOR_ID = 'RUKI'
GROUP BY AM.NUM, AM.NOMBRE
HAVING COUNT(DISTINCT ESC.COD_OB) =
    (SELECT COUNT(*)
    FROM escribir E2
    WHERE E2.AUTOR_ID = 'RUKI');
```

Fíjate que volvemos a no necesitar la tabla LIBRO ni la tabla OBRA:

```
SELECT AM.NOMBRE
FROM amigo AM, prestamo P, esta_en EST, escribir ESC
WHERE AM.NUM=P.NUM AND P.ID_LIB=EST.ID_LIB AND EST.COD_OB=ESC.COD_OB AND ESC.AUTOR_ID = 'RUKI'
GROUP BY AM.NUM, AM.NOMBRE
HAVING COUNT(DISTINCT ESC.COD_OB) =
    (SELECT COUNT(*)
    FROM escribir E2
    WHERE E2.AUTOR_ID = 'RUKI');
```

!BIBLIOTECA #16 Obtener el nombre de los amigos que han leído todas las obras del autor de identificador 'JAGR'.

REPETIDA. Igual que la 15

```
NOMBRE
------------------------------------------------------------
0 filas seleccionadas.
```

!BIBLIOTECA #17 Obtener nombre de los amigos que han leído todas las obras de algún autor

```
NOMBRE
--------------------------------------------
Isabel Peiró García
Yolanda Milanés Cuba
```

Buscamos: Nombre de los amigos para los que existe algún autor que haya escrito algo para el que no existe una obra escrita por ese autor (que ha escrito algo) que no esté en la lista de obras leídas (PRESTADAS) por ese amigo.

- **nombre de los amigos => tabla AMIGO**
- **para los que EXISTE algún autor que haya escrito algo => tabla ESCRIBIR ESC**
- **para el que NO EXISTE una obra de ese autor (que ha escrito algo) => tabla ESCRIBIR ESC2**
- **que no esté en la lista de las obras leídas por ese amigo => tablas PRESTAMO y ESTA_EN**

```
-- AMIGOS ...
SELECT AM.NOMBRE
FROM amigo AM
-- (AMIGOS) PARA LOS QUE EXISTE ALGÚN ...
WHERE EXISTS
  (  -- AUTOR (QUE HAYA ESCRITO ALGO)...
     SELECT ESC.AUTOR_ID
     FROM escribir ESC
     -- (AUTOR QUE HAYA ESCRITO ALGO) PARA EL QUE NO EXISTE...
     WHERE NOT EXISTS
          (-- UNA OBRA...
          SELECT ESC2.COD_OB
          FROM escribir ESC2
          -- (UNA OBRA) ESCRITA POR ESE AUTOR QUE HA ESCRITO ALGO...
          WHERE ESC2.AUTOR_ID=ESC.AUTOR_ID
          -- (UNA OBRA ESCRITA POR ESE AUTOR QUE HA ESCRITO ALGO) QUE NO ESTÉ...
          AND ESC2.COD_OB NOT IN
               (-- EN LA LISTA DE (TODAS LAS) OBRAS...
               SELECT EE.COD_OB
               FROM prestamo P, esta_en EE
               WHERE P.ID_LIB=EE.ID_LIB
               -- (EN LA LISTA DE TODAS LAS OBRAS) QUE HA LEÍDO ESE AMIGO...
               AND P.NUM=AM.NUM)));
```

!BIBLIOTECA #18 Resolver la consulta anterior indicando también el nombre de ese autor.

nombre	nombre
Isabel Peiró García	Maalouf, Amin
Isabel Peiró García	Kipling, Rudyard
Yolanda Milanés Cuba	Vian, Boris

Para obtener el nombre del autor, cambiaremos el primer EXISTS por un IN…

En la consulta anterior (17) buscábamos: Nombre de los amigos para los que existe algún autor con ciertas condiciones …

En esta consulta (18) buscamos: Nombre de los amigos cruzados con autores para los que el AUTOR está en la lista de autores que cumplen con ciertas condiciones...

LISTADO DE AMIGOS...	**LISTADO DE AMIGOS Y AUTORES...**
-- (AMIGOS) ... SELECT AM.NOMBRE FROM amigo AM **-- (AMIGOS) PARA LOS QUE EXISTE ALGÚN ...** WHERE EXISTS (**-- AUTOR (QUE HAYA ESCRITO ALGO)...** SELECT ESC.AUTOR_ID [...])	**-- (AMIGOS) ...** SELECT AM.NOMBRE, AU.NOMBRE FROM amigo AM, autor AU **-- (AMIGOS) PARA LOS QUE EXISTE ALGÚN ...** WHERE AU.AUTOR_ID IN (**-- AUTOR (QUE HAYA ESCRITO ALGO)...** SELECT ESC.AUTOR_ID [...])

```
-- AMIGOS ...
SELECT AM.NOMBRE, AU.nombre
FROM amigo AM, autor AU
-- (AMIGOS) PARA LOS QUE EXISTE ALGÚN ...
WHERE AU.AUTOR_ID IN
  (  -- AUTOR (QUE HAYA ESCRITO ALGO)...
     SELECT ESC.AUTOR_ID
     FROM escribir ESC
     -- (AUTOR QUE HAYA ESCRITO ALGO) PARA EL QUE NO EXISTE...
     WHERE NOT EXISTS
          (-- UNA OBRA...
          SELECT ESC2.COD_OB
          FROM escribir ESC2
          -- (UNA OBRA) ESCRITA POR ESE AUTOR QUE HA ESCRITO ALGO...
          WHERE ESC2.AUTOR_ID=ESC.AUTOR_ID
          -- (UNA OBRA ESCRITA POR ESE AUTOR QUE HA ESCRITO ALGO) QUE NO ESTÉ...
          AND ESC2.COD_OB NOT IN
               (-- EN LA LISTA DE (TODAS LAS) OBRAS...
               SELECT EE.COD_OB
               FROM prestamo P, esta_en EE
               WHERE P.ID_LIB=EE.ID_LIB
               -- (EN LA LISTA DE TODAS LAS OBRAS) QUE HA LEÍDO ESE AMIGO...
               AND P.NUM=AM.NUM)));
```

!BIBLIOTECA #19 Obtener el nombre de los amigos que han leído alguna obra del autor de identificador 'CAMA'.

REPETIDA. Igual que la 14

```
NOMBRE
--------------------------------------------------
Pepe Pérez Pérez
Isabel Peiró García
Isidro Catalá Ferrer
3 filas seleccionadas.
```

!BIBLIOTECA #20 Obtener el nombre de los amigos que solo han leído obras del autor de identificador 'CAMA'.

```
NOMBRE
--------------------------------------------------
Pepe Pérez Pérez
Isidro Catalá Ferrer
2 filas seleccionadas.
```

```
-- AMIGOS ...
SELECT AM.NOMBRE
FROM amigo AM
-- (AMIGOS) PARA LOS QUE NO EXISTE ...
WHERE NOT EXISTS
        (-- UN AUTOR...
        SELECT ESC.AUTOR_ID
-- Estrategia de enlazado total de cruces de extremo a extremo
        FROM prestamo P, esta_en EE, escribir ESC
        -- (UN AUTOR) HA ESCRITO DE UNA OBRA PRESTADA POR ESE AMIGO...
        WHERE P.ID_LIB=EE.ID_LIB AND EE.COD_OB=ESC.COD_OB
        AND P.NUM=AM.NUM
        -- (UN AUTOR...) Y QUE NO SE LLAMA "CAMA"
        AND ESC.AUTOR_ID<>'CAMA')
AND
-- Y HAN LEÍDO ALGO DE ESE AUTOR...
EXISTS
        (-- UN AUTOR...
        SELECT ESC2.AUTOR_ID
        FROM prestamo P2, esta_en EE2, escribir ESC2
        -- (UN AUTOR) HA ESCRITO DE UNA OBRA PRESTADA POR ESE AMIGO...
        WHERE P2.ID_LIB=EE2.ID_LIB AND EE2.COD_OB=ESC2.COD_OB
        AND P2.NUM=AM.NUM
        -- (UN AUTOR...) Y QUE SE LLAMA "CAMA"
        AND ESC2.AUTOR_ID='CAMA');
```

Otra opción:

```
SELECT nombre
FROM amigo a
WHERE num in
    (SELECT num
    FROM prestamo)
AND
NOT EXISTS
    (SELECT *
    FROM escribir e
    WHERE e.AUTOR_ID<>'CAMA' AND EXISTS
        (SELECT *
        FROM esta_en ee, prestamo p
        WHERE p.NUM=a.NUM AND ee.COD_OB = e.COD_OB AND ee.ID_LIB=p.ID_LIB));
```

Otra opción:

```
SELECT DISTINCT a.nombre
FROM escribir e, esta_en ee, prestamo p, amigo a
WHERE e.AUTOR_ID = 'CAMA'
AND e.COD_OB = ee.COD_OB AND ee.ID_LIB = p.ID_LIB AND p.NUM = a.NUM
AND NOT EXISTS
    (SELECT *
    FROM escribir e2, esta_en ee2, prestamo p2
    WHERE a.NUM = p2.NUM
    AND e2.AUTOR_ID <> e.AUTOR_ID
    AND e2.COD_OB = ee2.COD_OB AND ee2.ID_LIB = p2.ID_LIB);
```

!BIBLIOTECA #21 Obtener el nombre de los amigos que solo han leído obras de un autor.

```
NOMBRE
--------------------------------------------------
Pepe Pérez Pérez
Eloy Prim Gros
Yolanda Milanés Cuba
Isidro Catalá Ferrer
4 filas seleccionadas.
```

Estrategia: enlazado TOTAL y group by

```
SELECT AM.NOMBRE
-- Estrategia de enlazado total de cruces de extremo a extremo
FROM amigo AM, prestamo P, libro L, esta_en EST, obra O, escribir ESC
WHERE AM.NUM=P.NUM AND P.ID_LIB=L.ID_LIB AND L.ID_LIB=EST.ID_LIB AND EST.COD_OB=O.COD_OB AND O.COD_OB=ESC.COD_OB
GROUP BY AM.NOMBRE
HAVING COUNT(DISTINCT ESC.AUTOR_ID) =1;
```

Versión óptima (eliminamos tablas innecesarias y optimizamos el GROUP BY)

```
SELECT AM.NOMBRE
FROM amigo AM, prestamo P, esta_en EST, escribir ESC
WHERE AM.NUM=P.NUM AND P.ID_LIB =EST.ID_LIB AND EST.COD_OB = ESC.COD_OB
GROUP BY AM.NUM, AM.NOMBRE
HAVING COUNT(DISTINCT ESC.AUTOR_ID) =1;
```

Estrategia: subconsulta

```
SELECT AM.NOMBRE
FROM amigo AM
WHERE AM.NUM in
    (SELECT P.NUM
    FROM prestamo P, esta_en EE, escribir ESC
    WHERE P.ID_LIB =EE.ID_LIB AND EE.COD_OB = ESC.COD_OB
    GROUP BY P.NUM
    HAVING COUNT(DISTINCT ESC.AUTOR_ID) =1);
```

!BIBLIOTECA #22 Resolver la consulta anterior indicando también el nombre del autor.

```
+---------------------+---------------------+
| NOMBRE_AMIGO        | NOMBRE_AUTOR        |
+---------------------+---------------------+
| Pepe Pérez Pérez    | Martín Gaite, Carmen |
| Isidro Catalá Ferrer | Martín Gaite, Carmen |
| Yolanda Milanés Cuba | Vian, Boris         |
| Eloy Prim Gros      | Kipling, Rudyard    |
+---------------------+---------------------+
4 rows in set (0.00 sec)
```

Haremos un enlazado total de cruces y comprobaremos que no hay otro cruce total para ese mismo AMIGO con diferente AUTOR. Realizaremos tres pasos:

PASO 1: Construimos los dos cruces y vemos qué obtenemos (puedes hacerlo con papel y boli inventándote los resultados si no dispones de acceso a la BD)

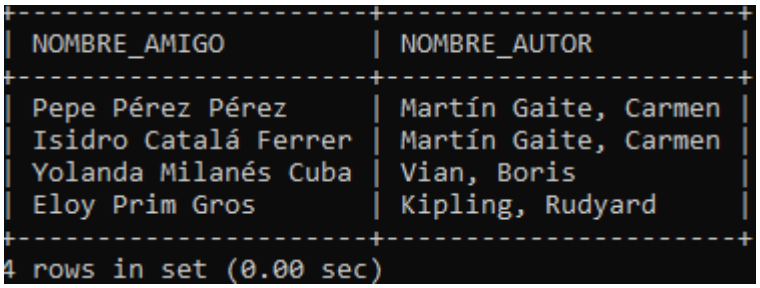
```
SELECT AM.NOMBRE AS NOMBRE_AMIGO, AU.NOMBRE AS NOMBRE_AUTOR
-- Estrategia de enlazado total de cruces de extremo a extremo (1º)
FROM amigo AM, prestamo P, libro L, esta_en EST, obra O, escribir ESC , autor AU
WHERE AM.NUM=P.NUM AND P.ID_LIB=L.ID_LIB AND L.ID_LIB=EST.ID_LIB AND EST.COD_OB=O.COD_OB AND O.COD_OB=ESC.COD_OB AND ESC.AUTOR_ID=AU.AUTOR_ID AND AM.NUM NOT IN
    (SELECT P2.NUM
-- Estrategia de enlazado total de cruces de extremo a extremo (2º)
FROM amigo AM2, prestamo P2, libro L2, esta_en EST2, obra O2, escribir ESC2 , autor AU2
WHERE AM2.NUM=P2.NUM AND P2.ID_LIB=L2.ID_LIB AND L2.ID_LIB=EST2.ID_LIB AND EST2.COD_OB=O2.COD_OB AND O2.COD_OB=ESC2.COD_OB AND ESC2.AUTOR_ID=AU2.AUTOR_ID
    AND ESC2.AUTOR_ID<>AU.AUTOR_ID);
```

```
+---------------------+---------------------+
| NOMBRE_AMIGO        | NOMBRE_AUTOR        |
+---------------------+---------------------+
| Pepe Pérez Pérez    | Martín Gaite, Carmen |
| Isidro Catalá Ferrer | Martín Gaite, Carmen |
| Pepe Pérez Pérez    | Martín Gaite, Carmen |
| Yolanda Milanés Cuba | Vian, Boris         |
| Yolanda Milanés Cuba | Vian, Boris         |
| Yolanda Milanés Cuba | Vian, Boris         |
| Yolanda Milanés Cuba | Vian, Boris         |
| Yolanda Milanés Cuba | Vian, Boris         |
| Eloy Prim Gros      | Kipling, Rudyard    |
+---------------------+---------------------+
9 rows in set (0.00 sec)
```

PASO 2: Aplicamos un DISTINCT a la consulta anterior para eliminar repetidos

SELECT DISTINCT AM.NOMBRE AS NOMBRE_AMIGO, AU.NOMBRE AS NOMBRE_AUTOR

[….]



```
+---------------------+---------------------+
| NOMBRE_AMIGO        | NOMBRE_AUTOR        |
+---------------------+---------------------+
| Pepe Pérez Pérez    | Martín Gaite, Carmen |
| Isidro Catalá Ferrer | Martín Gaite, Carmen |
| Yolanda Milanés Cuba | Vian, Boris         |
| Eloy Prim Gros      | Kipling, Rudyard    |
+---------------------+---------------------+
4 rows in set (0.00 sec)
```

PASO 3: Eliminamos las tablas que no necesitamos. Presta atención a que no necesitamos las tablas de los extremos en el segundo cruce (AMIGO y AUTOR) al no necesitar datos específicos de esas tablas y tener sus PK en las tablas de cruce próximas (PRESTAMO y ESCRIBE).

Versión óptima (eliminamos tablas innecesarias)

SELECT DISTINCT AM.NOMBRE AS NOMBRE_AMIGO, AU.NOMBRE AS NOMBRE_AUTOR

-- Estrategia de enlazado total de cruces de extremo a extremo (1º)

FROM amigo AM, prestamo P, esta_en EST, escribir ESC , autor AU

WHERE AM.NUM=P.NUM AND P.ID_LIB=EST.ID_LIB AND EST.COD_OB=ESC.COD_OB AND ESC.AUTOR_ID=AU.AUTOR_ID AND AM.NUM NOT IN

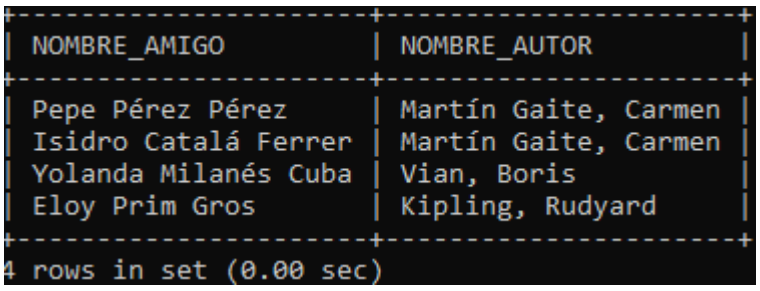
(SELECT P2.NUM

-- Estrategia de enlazado total de cruces de extremo a extremo (2º)

FROM prestamo P2, esta_en EST2, escribir ESC2

WHERE P2.ID_LIB=EST2.ID_LIB AND EST2.COD_OB=ESC2.COD_OB

AND ESC2.AUTOR_ID<>AU.AUTOR_ID);



```
+---------------------+---------------------+
| NOMBRE_AMIGO        | NOMBRE_AUTOR        |
+---------------------+---------------------+
| Pepe Pérez Pérez    | Martín Gaite, Carmen |
| Isidro Catalá Ferrer | Martín Gaite, Carmen |
| Yolanda Milanés Cuba | Vian, Boris         |
| Eloy Prim Gros      | Kipling, Rudyard    |
+---------------------+---------------------+
4 rows in set (0.00 sec)
```

!BIBLIOTECA #23 Obtener el nombre de los amigos que han leído todas las obras de algún autor y no han leído nada de ningún otro indicando también el nombre del autor.

```
NOMBRE                         NOMBRE
-------------------------------------------------------
Yolanda Milanés Cuba           Vian, Boris
1 fila seleccionada.
```

Buscamos: Nombre de los amigos y autores leídos por esos amigos (enlazado total de cruces de extremo a extremo), para los que NO EXISTE una obra de ese autor que NO EXISTA en la lista de obras leídas por ese amigo. Además, que el contador de autores leídos por ese amigo sea de 1.

Mostraremos la opción ÓPTIMA directamente, eliminando las tablas de cruce innecesarias y aplicando el DISTINCT por la misma razón que lo hemos aplicado en la consulta anterior.

```
-- AMIGOS y AUTORES...
SELECT DISTINCT AM.NOMBRE, AU.NOMBRE
-- Estrategia de enlazado total de cruces de extremo a extremo
FROM amigo AM, prestamo P, esta_en EST,  escribir ESC , autor AU
WHERE AM.NUM=P.NUM AND P.ID_LIB=EST.ID_LIB AND EST.COD_OB=ESC.COD_OB AND ESC.AUTOR_ID=AU.AUTOR_ID
-- (AMIGOS Y AUTORES) PARA LOS QUE NO EXISTE ALGUNA ..
AND NOT EXISTS
  (
    -- OBRA (QUE HAYA ESCRITO ALGUIEN)...
    SELECT ESC2.COD_OB
    FROM escribir ESC2
    -- (OBRA QUE HAYA ESCRITO…) UNO DE LOS AUTORES DE ARRIBA…
    WHERE ESC.AUTOR_ID=ESC2.AUTOR_ID
    AND ESC2.COD_OB NOT IN
                (
                -- EN LA LISTA DE (TODAS LAS) OBRAS...
                SELECT EE3.COD_OB
-- Estrategia de enlazado total de cruces de extremo a extremo
                FROM prestamo P3, esta_en EE3
                WHERE P3.ID_LIB=EE3.ID_LIB
                -- (EN LA LISTA DE TODAS LAS OBRAS) QUE HA LEÍDO ESE AMIGO...
                AND P3.NUM=AM.NUM
                )
        )
AND
    -- Y QUE EL NÚMERO DE AUTORES DISTINTOS LEÍDOS SEA 1
    (1=
        (SELECT COUNT(DISTINCT ESC5.AUTOR_ID)
-- Estrategia de enlazado total de cruces de extremo a extremo
        FROM prestamo P5, esta_en EST5,  escribir ESC5
        WHERE P5.ID_LIB=EST5.ID_LIB AND EST5.COD_OB=ESC5.COD_OB
        AND P5.NUM=AM.NUM)
    );
```