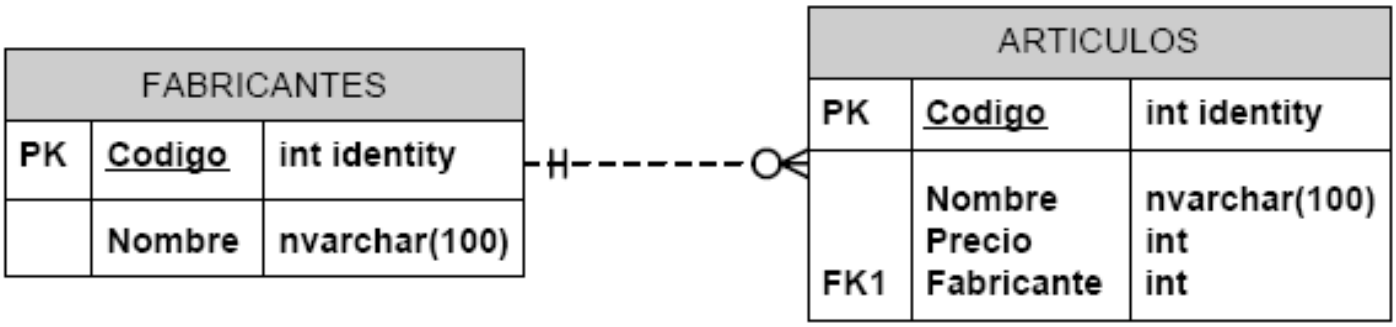
ARTÍCULOS



1. Obtener los nombres de los productos de la tienda

```
SELECT Nombre FROM ARTICULOS
```

2. Obtener los nombres y los precios de los productos de la tienda

```
SELECT Nombre, Precio FROM ARTICULOS
```

3. Obtener el nombre de los productos cuyo precio sea menor o igual a 200 €

```
SELECT Nombre FROM ARTICULOS WHERE Precio <= 200
```

4. Obtener todos los datos de los artículos cuyo precio esté entre los 60 € y los 120€ (ambas cantidades incluidas).

```
/* Con AND */
SELECT * FROM ARTICULOS
WHERE Precio >= 60 AND Precio <= 120

/* Con BETWEEN */
SELECT * FROM ARTICULOS
WHERE Precio BETWEEN 60 AND 120
```

5. Obtener el nombre y el precio en pesetas (es decir, el precio en euros multiplicado por 166,386).

```
/* Sin AS */
SELECT Nombre, Precio * 166.386 FROM ARTICULOS

/* Con AS */
SELECT Nombre, Precio * 166.386 AS PrecioPtas FROM ARTICULOS
```

6. Seleccionar el precio medio de todos los productos.

```
SELECT AVG(Precio) FROM ARTICULOS
```

7. Obtener el precio medio de los artículos cuyo código de fabricante sea 2

```
SELECT AVG(Precio) FROM ARTICULOS WHERE Fabricante=2
```

8. Obtener el número de artículos cuyo precio sea mayor o igual a 180€.

```
SELECT COUNT(*) FROM ARTICULOS WHERE Precio >=180
```

9. Obtener el nombre y precio de los artículos cuyo precio sea mayor o igual a 180€ y ordenarlos descendentemente por precio, y luego ascendentemente por nombre.

```
SELECT Nombre, Precio FROM ARTICULOS
WHERE Precio >= 180
ORDER BY Precio DESC, Nombre
```

10. Obtener un listado completo de artículos, incluyendo por cada artículo los datos del artículo y de su fabricante.

```
/* Sin INNER JOIN */
SELECT * FROM ARTICULOS, FABRICANTES
WHERE ARTICULOS.Fabricante = FABRICANTES.Codigo

/* Con INNER JOIN */
SELECT *
FROM ARTICULOS INNER JOIN FABRICANTES
ON ARTICULOS.Fabricante = FABRICANTES.Codigo
```

11. Obtener un listado de artículos, incluyendo el nombre del artículo, su precio y el nombre de su fabricante.

```
/* Sin INNER JOIN */
SELECT ARTICULOS.Nombre, Precio, FABRICANTES.Nombre
FROM ARTICULOS, FABRICANTES
WHERE ARTICULOS.Fabricante = FABRICANTES.Codigo

/* Con INNER JOIN */
SELECT ARTICULOS.Nombre, Precio, FABRICANTES.Nombre
FROM ARTICULOS INNER JOIN FABRICANTES
ON ARTICULOS.Fabricante = FABRICANTES.Codigo
```

12. Obtener el precio medio de los productos de cada fabricante, mostrando solo los códigos de fabricante.

```
SELECT AVG(Precio), Fabricante FROM ARTICULOS
GROUP BY Fabricante
```

13. Obtener el precio medio de los productos de cada fabricante, mostrando el nombre del fabricante.

```
/* Sin INNER JOIN */
SELECT AVG(Precio), FABRICANTES.Nombre
FROM ARTICULOS, FABRICANTES
WHERE ARTICULOS.Fabricante = FABRICANTES.Codigo
GROUP BY FABRICANTES.Nombre

/* Con INNER JOIN */
SELECT AVG(Precio), FABRICANTES.Nombre
FROM ARTICULOS INNER JOIN FABRICANTES
ON ARTICULOS.Fabricante = FABRICANTES.Codigo
GROUP BY FABRICANTES.Nombre
```

14. Obtener los nombres de los fabricantes que ofrezcan productos cuyo precio medio sea mayor o igual a 150 €.

```
/* Sin INNER JOIN */
SELECT AVG(Precio), FABRICANTES.Nombre
FROM ARTICULOS, FABRICANTES
WHERE ARTICULOS.Fabricante = FABRICANTES.Codigo
GROUP BY FABRICANTES.Nombre
HAVING AVG(Precio) >= 150

/* Con INNER JOIN */
```

```
SELECT AVG(Precio), FABRICANTES.Nombre
FROM ARTICULOS INNER JOIN FABRICANTES
ON ARTICULOS.Fabricante = FABRICANTES.Codigo
GROUP BY FABRICANTES.Nombre
HAVING AVG(Precio) >= 150
```

15. Obtener el nombre y precio del artículo más barato.

```
SELECT Nombre, Precio
FROM ARTICULOS
WHERE Precio = (SELECT MIN(Precio) FROM ARTICULOS)
```

16. Obtener una lista con el nombre y precio de los artículos más caros de cada proveedor (incluyendo el nombre del proveedor)

```
/* Sin INNER JOIN */
SELECT A.Nombre, A.Precio, F.Nombre
FROM ARTICULOS A, FABRICANTES F
WHERE A.Fabricante = F.Codigo
AND A.Precio =
(
SELECT MAX(A.Precio)
FROM ARTICULOS A
WHERE A.Fabricante = F.Codigo
)

/* Con INNER JOIN */
SELECT A.Nombre, A.Precio, F.Nombre
FROM ARTICULOS A INNER JOIN FABRICANTES F
ON A.Fabricante = F.Codigo
AND A.Precio =
(
SELECT MAX(A.Precio)
FROM ARTICULOS A
WHERE A.Fabricante = F.Codigo
```

17. Añadir un nuevo producto: Altavoces de 30 € (del fabricante 2)

```
INSERT INTO ARTICULOS( Nombre , Precio , Fabricante)
VALUES ( 'Altavoces' , 70 , 2 )
```

18. Cambiar el nombre del producto 8 a 'Impresora Laser'.

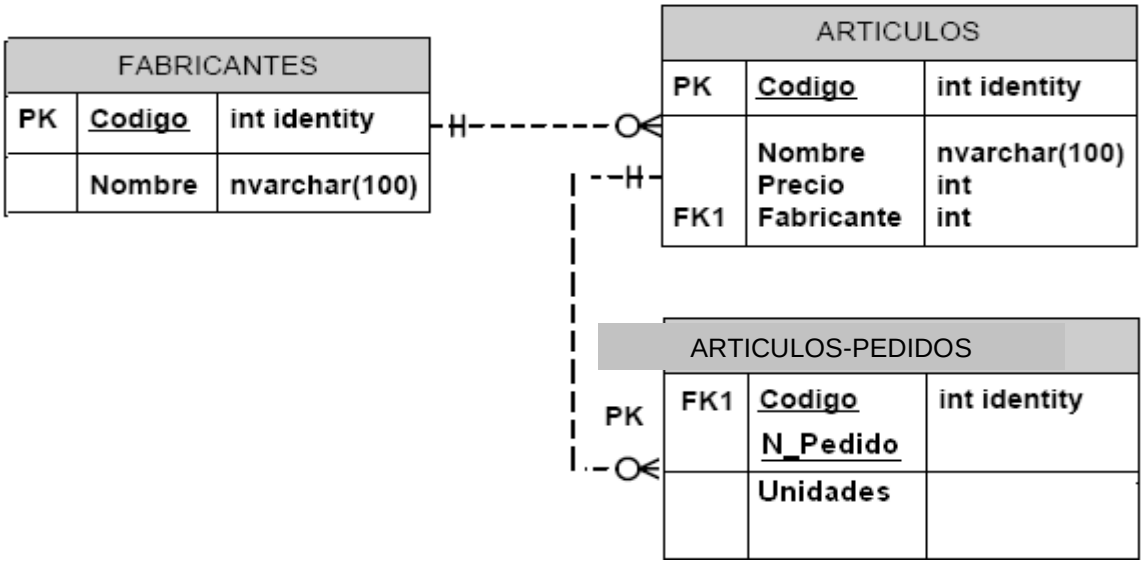
```
UPDATE ARTICULOS
SET Nombre = 'Impresora Laser'
WHERE Codigo = 8
```

19. Aplicar un descuento del 10% (multiplicar el precio por 0,9) a todos los productos.

```
UPDATE ARTICULOS
SET Precio = Precio * 0.9
```

20. Aplicar un descuento de 10 € a todos los productos cuyo precio sea mayor o igual a 120 €.

```
UPDATE ARTICULOS
SET Precio = Precio - 10
WHERE Precio >= 120
```



21. Mostrar los rgtros repetidos (nombre, precio y fabricante) de la tabla artículos y el nº de ocurrencias.

```
SELECT  Nombre, Precio, Fabricante, count(*) AS Total
FROM  Artículos
GROUP BY Nombre, Precio, Fabricante
HAVING count(*) >1
```

22. Mostrar los rgtros no repetidos (nombre, precio y fabricante) de la tabla artículos.

```
SELECT  Nombre, Precio, Fabricante, count(*) AS Total
FROM  Artículos
GROUP BY Nombre, Precio, Fabricante
HAVING count(*) =1
```

23. Mostrar los rgtros de la tabla artículos-pedidos (nombre y fabricante) que existan en la tabla artículos.

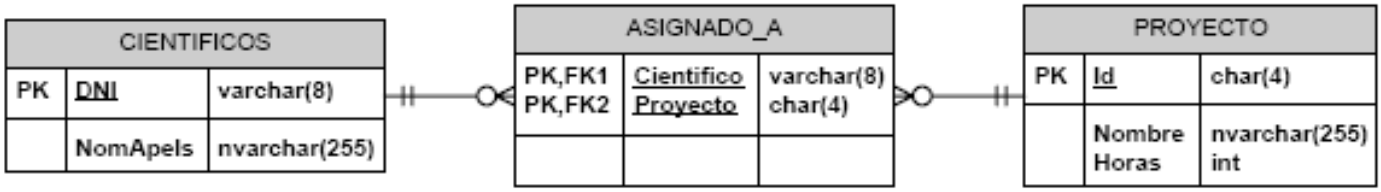
```
SELECT  DISTINCT A.Nombre, A.Fabricante
FROM  Artículos A LEFT JOIN Artículos_Pedidos AP ON A.Código = AP.Código
```

24. Mostrar los rgtros de la tabla artículos-pedidos que no existan en artículos.

```
SELECT  AP.Código, AP.N_Pedido, AP.Unidades
FROM  Artículos_Pedidos AP
WHERE NOT EXISTS        (SELECT A.Código
                          FROM Artículos A
                          WHERE A.Código = AP.Código)
SELECT  AP.Código, AP.N_Pedido, AP.Unidades
FROM  Artículos_Pedidos AP
WHERE AP.Cógigo NOT IN(SELECT Código
                          FROM Artículos)
```

25. Mostrar los rgtros de artículos que no existan en artículos pedidos.

CIENTÍFICOS



1. Sacar una relación completa de los científicos asignados a cada proyecto. Mostrar DNI, Nombre del científico, Identificador del proyecto y nombre del proyecto

```
/* Sin JOIN */
SELECT DNI, NomApels, Id, Nombre
FROM CIENTIFICOS C, ASIGNADO_A A, PROYECTO P
WHERE C.DNI = A.Cientifico AND A.Proyecto = P.Id

/* Con JOIN */
SELECT DNI, NomApels, Id, Nombre
FROM CIENTIFICOS C INNER JOIN
(ASIGNADO_A A INNER JOIN PROYECTO P ON A.Proyecto = P.Id)
ON C.DNI = A.Cientifico
```

2. Obtener el numero de proyectos al que está asignado cada científico (mostrar el DNI y el nombre).

```
SELECT DNI, NomApels, COUNT(Proyecto)
FROM CIENTIFICOS LEFT JOIN ASIGNADO_A
ON CIENTIFICOS.DNI = ASIGNADO_A.Cientifico
GROUP BY DNI, NomApels
```

3. Obtener el numero de científicos asignados a cada proyecto (mostrar el identificador del proyecto y el nombre del proyecto.

```
SELECT Id, Nombre, COUNT(Proyecto)
FROM PROYECTO LEFT JOIN ASIGNADO_A
ON PROYECTO.Id = ASIGNADO_A.Proyecto
GROUP BY Id, Nombre
```

4. Obtener el número de horas de dedicación de cada científico.

```
SELECT DNI,NomApels, SUM(Horas)
FROM CIENTIFICOS C LEFT JOIN
(ASIGNADO_A A INNER JOIN PROYECTO P ON A.Proyecto = P.Id)
ON C.DNI = A.Cientifico
GROUP BY DNI,NomApels
```

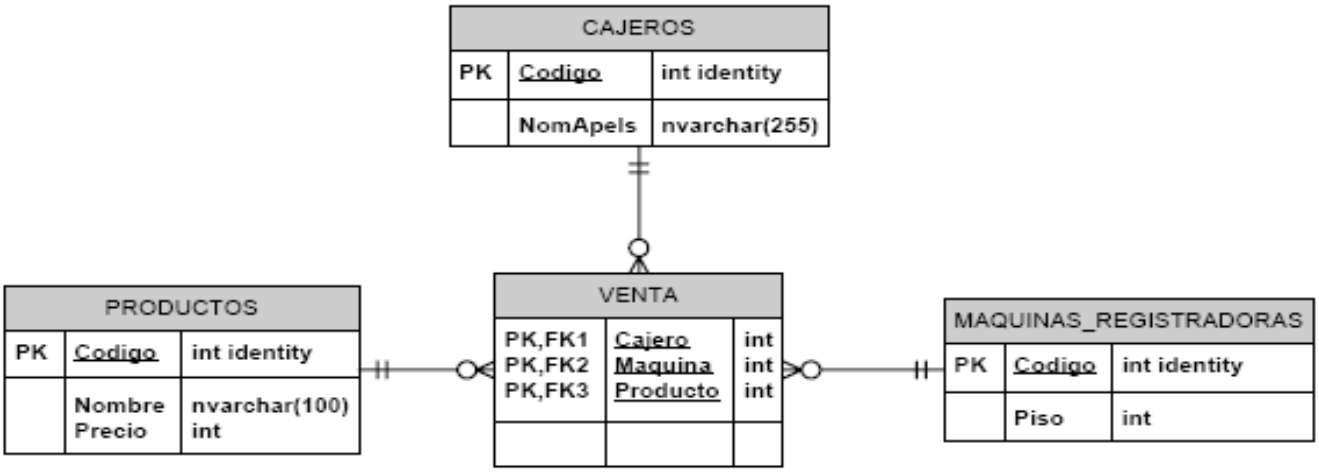
5. Obtener el DNI y el nombre de los científicos que se dedican a más de un proyecto y cuya dedicación media a cada proyecto sea superior a 80 horas.

```
/* Con dos subconsultas */
SELECT DNI, NomApels
FROM CIENTIFICOS C
WHERE 1 < ( SELECT COUNT(*) FROM ASIGNADO_A
              WHERE Cientifico = C.DNI
            )
```

```
        AND 80 < ( SELECT AVG(Horas)
                   FROM PROYECTO INNER JOIN ASIGNADO_A
                        ON PROYECTO.Id = ASIGNADO_A.Proyecto
                   WHERE Cientifico = C.DNI)

/* Juntando tablas y con HAVING */
SELECT DNI, NomApels
FROM CIENTIFICOS C, ASIGNADO_A A, PROYECTO P
WHERE C.DNI = A.Cientifico  AND P.Id = A.Proyecto
GROUP BY DNI, NomApels
HAVING COUNT(Proyecto) > 1 AND AVG(Horas) > 80
```

GRANDES ALMACENES



1. Mostrar el número de ventas de cada producto, ordenado de más a menos ventas.

```
SELECT Codigo, Nombre, COUNT(VENTA.Producto)
FROM PRODUCTOS LEFT JOIN VENTA
        ON PRODUCTOS.Codigo = VENTA.Producto
GROUP BY Codigo, Nombre
ORDER BY COUNT(VENTA.Producto) DESC
```

2. Obtener un informe completo de ventas, indicando el nombre del cajero que realizó la venta, nombre y precios de los productos vendidos, y piso en el que se encuentra la maquina registradora donde se realizó la venta.

```
/* Sin JOIN */
SELECT NomApels, Nombre, Precio, Piso
FROM VENTA V, CAJEROS C, PRODUCTOS P, MAQUINAS_REGISTRADORAS M
WHERE V.Cajero = C.Codigo AND V.Producto = P.Codigo AND V.Maquina = M.Codigo

/* Con JOIN */
SELECT NomApels, Nombre, Precio, Piso
FROM CAJEROS C INNER JOIN
(PRODUCTOS P INNER JOIN
(MAQUINAS_REGISTRADORAS M INNER JOIN VENTA V
ON V.Maquina = M.Codigo) ON V.Producto = P.Codigo)
ON V.Cajero = C.Codigo
```

3. Obtener las ventas totales realizadas en cada piso.

```
SELECT Piso, SUM(Precio)
FROM VENTA V, PRODUCTOS P, MAQUINAS_REGISTRADORAS M
WHERE V.Producto = P.Codigo AND V.Maquina = M.Codigo
```

```
GROUP BY Piso
```

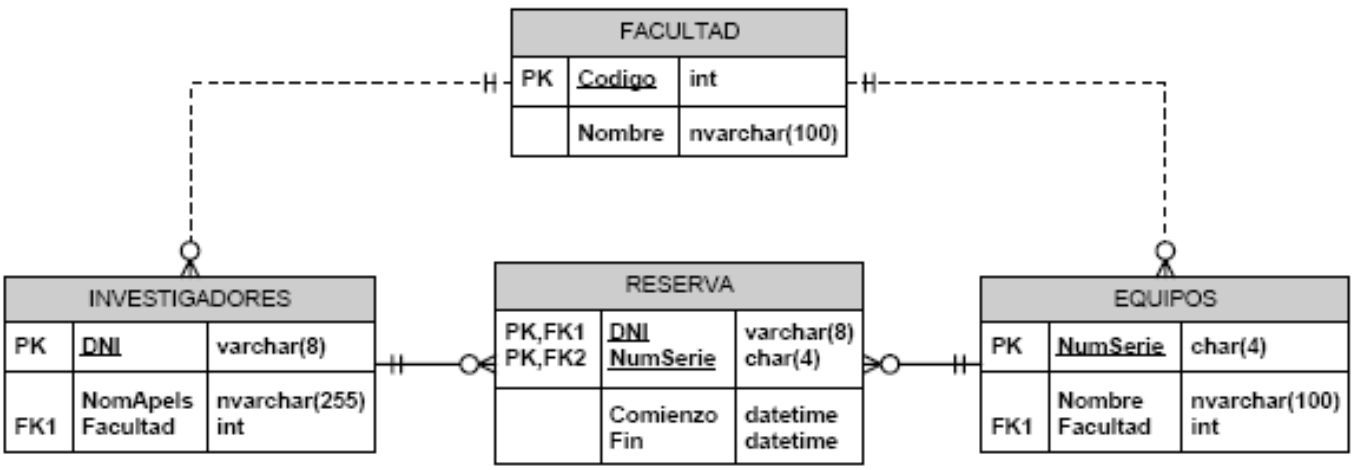
4. Obtener el código y nombre de cada empleado junto con el importe total de sus ventas.

```
SELECT C.Codigo, C.NomApels, SUM(Precio)
FROM PRODUCTOS P INNER JOIN
(CAJEROS C LEFT JOIN VENTA V ON V.Cajero = C.Codigo)
ON V.Producto = P.Codigo
GROUP BY C.Codigo, NomApels
```

5. Obtener el código y nombre de aquellos cajeros que hayan realizado ventas en pisos cuyas ventas totales sean inferiores a los 500 ¤.

```
SELECT Codigo, NomApels
FROM CAJEROS
WHERE Codigo IN
   (SELECT Cajero
    FROM VENTA
    WHERE Maquina IN
       (SELECT Codigo
        FROM MAQUINAS_REGISTRADORAS
        WHERE Piso IN
         (SELECT Piso
         FROM VENTA V, PRODUCTOS P, MAQUINAS_REGISTRADORAS M
         WHERE V.Producto = P.Codigo AND V.Maquina = M.Codigo
         GROUP BY Piso
         HAVING SUM(Precio) < 500
         )))
```

INVESTIGADORES



1. Obtener el DNI y nombre de aquellos investigadores que han realizado más de una reserva.

```
/* Juntando tablas */
SELECT I.DNI, NomApels
FROM INVESTIGADORES I
LEFT JOIN RESERVA R
ON R.DNI = I.DNI
GROUP BY I.DNI, NomApels
HAVING COUNT(R.DNI) > 1
```

```
/* Con subconsulta */
SELECT DNI, NomApels
FROM INVESTIGADORES
WHERE DNI IN
(SELECT DNI FROM
RESERVA
```

```
GROUP BY DNI                                    HAVING COUNT(*) > 1)
```

2. Obtener un listado completa de reservas, incluyendo los siguientes datos:
   a. DNI y nombre del investigador, junto con el nombre de su facultad.
   b. Numero de serie y nombre del equipo reservado, junto con el nombre de la facultad a la que pertenece.
   c. Fecha de comienzo y fin de la reserva.

```
SELECT I.DNI, NomApels, F_INV.Nombre, E.NumSerie, E.Nombre, F_EQUIP.Nombre,
Comienzo, Fin
FROM RESERVA R,INVESTIGADORES I,EQUIPOS E,FACULTAD F_INV, FACULTAD
F_EQUIP
WHERE R.DNI = I.DNI AND R.NumSerie = E.NumSerie
AND I.Facultad = F_INV.Codigo AND E.Facultad = F_EQUIP.Codigo
```

3. Obtener el DNI y el nombre de los investigadores que han reservado equipos que no son de su facultad.

```
/* Juntando tablas */
SELECT DISTINCT I.DNI,NomApels
FROM RESERVA R,INVESTIGADORES I,EQUIPOS E
WHERE R.DNI = I.DNI
AND R.NumSerie = E.NumSerie
AND I.Facultad <> E.Facultad

/* Con EXISTS */
SELECT DNI, NomApels
FROM INVESTIGADORES I
WHERE EXISTS
(SELECT *
FROM RESERVA R INNER JOIN EQUIPOS E
ON R.NumSerie = E.NumSerie
WHERE R.DNI = I.DNI
AND I.Facultad <> E.Facultad)
```

4. Obtener los nombres de las facultades en las que ningún investigador ha realizado una reserva.

```
SELECT Nombre FROM FACULTAD
WHERE Codigo IN
(SELECT Facultad
FROM INVESTIGADORES I LEFT JOIN RESERVA R ON I.DNI = R.DNI
GROUP BY Facultad
HAVING COUNT(R.DNI)=0)
```

5. Obtener los nombres de las facultades con investigadores 'ociosos' (investigadores que no han realizado ninguna reserva).

```
SELECT Nombre FROM FACULTAD
WHERE Codigo IN
(SELECT Facultad FROM INVESTIGADORES
WHERE DNI NOT IN (SELECT DNI FROM RESERVA) )
```

6. Obtener el número de serie y nombre de los equipos que nunca han sido reservados.

```
/* Juntando tablas */
SELECT E.NumSerie, Nombre
FROM EQUIPOS E LEFT JOIN RESERVA R ON R.NumSerie = E.NumSerie
GROUP BY E.NumSerie,Nombre
HAVING COUNT(R.NumSerie)=0

/* Con subconsulta IN */
SELECT NumSerie, Nombre FROM EQUIPOS
WHERE NumSerie NOT IN(SELECT NumSerie FROM RESERVA)

/* Con EXISTS */
SELECT NumSerie, Nombre
FROM EQUIPOS E
WHERE NOT EXISTS (SELECT * FROM RESERVA R
                  WHERE R.NumSerie = E.NumSerie)
```

1. Búsqueda de registros existentes y no existentes entre tablas en T-SQL

   En este artículo (que es bastante sencillo pero demasiado útil) voy a exponer de una forma fácil y práctica la forma de obtener registros que se encuentren y no se encuentren en otra tabla según nuestros criterios de búsqueda.

   Esto resulta demasiado útil cuando intentamos detectar registros duplicados dentro de una tabla o dentro de varias tablas, o cuando necesitamos ver si en una tabla faltan datos de que se encentren en otra, etc.

   Supongamos lo siguiente: Tenemos una tabla con información de Compañeros, esta cuenta con un registro repetido dos veces:

   --Registros dentro de Companeros
   CompaneroId ApellidoPat ApellidoMat Nombres
   ---------- -------------- -------------- --------------
   1 Lopez Garcia Pedro
   2 Perez Rodriguez Juan
   3 Lopez Garcia Roque
   4 Perez Garcia Jesus
   5 Gonzalez Abudes Puyul
   6 Perez Rodriguez Juan

   (6 row(s) affected)

   Como pueden ver el registro repetido es:

   --Registros repetidos dentro de Companeros y el total de las ocurrencias
   ApellidoPat ApellidoMat Nombres Total
   ------------ ------------- ----------- -----------
   Perez Rodriguez Juan 2

   (1 row(s) affected)

   Y los registros que se mantienen sin redundancia son:

   --Registros no repetidos dentro de Companeros
   ApellidoPat ApellidoMat Nombres
   --------------- --------------- ------------------------
   Gonzalez Abudes Puyul
   Lopez Garcia Pedro
   Lopez Garcia Roque
   Perez Garcia Jesus

   (4 row(s) affected)

   Esto en veces es demasiado útil y práctico y es algo que se nos presenta muy seguido (por eso lo publico hasta a mi mismo se me olvida jejeje). Aquí les dejo los querys correspondientes:

- print '--Registros dentro de Companeros'

```
select * from Companeros
```

- print '--Registros repetidos dentro de Companeros y el total de las ocurrencias'

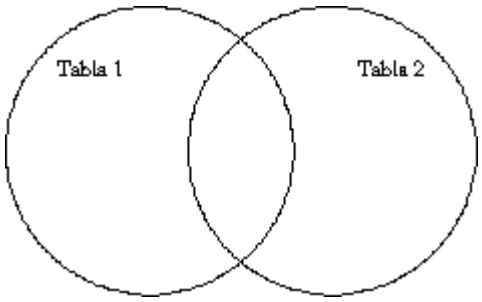
```
SELECT ApellidoPat,ApellidoMat,Nombres, count(*) Total
from Companeros
group by ApellidoPat,ApellidoMat, Nombres
having count(*) > 1
```

- print '--Registros no repetidos dentro de Companeros'

```
SELECT ApellidoPat,ApellidoMat,Nombres
from Companeros
group by ApellidoPat,ApellidoMat, Nombres
having count(*) = 1
```

Ahora, muchas de las ocasiones necesitamos determinar ocurrencias o redundancia entre datos almacenados en diversas tablas. Esto lo vamos a comenzar explicándolo de forma gráfica:

Supongamos el siguiente conjunto de datos:



Entre estos dos existe información en común, lo que requerimos es determinar esa intersección de información para determinar las ocurrencias entre los dos conjuntos de datos. Esto sería sencillo hacerlo con un simple INNER JOIN, pero hay ocasiones en las que cuando el número de tablas es mayor y estas se relacionan con más tablas resulta ser algo tedioso.

print '--Registros dentro de Amigos'

```
select * from Amigos
```

```
AmigoId ApellidoPat ApellidoMat Nombres
----------- --------------- --------------- ----------------
1 Villaneda Avila Sergio José
2 Villaneda Avila Andrea
3 Villaneda Avila Sarahi

(3 row(s) affected)
```

print '--Registros dentro de Conocidos'

```
select * from Conocidos
```

```
ConocidoId ApellidoPat ApellidoMat Nombres
----------- --------------- --------------- ---------------
1 Villaneda Avila Sergio José
2 Sanchez Gamboa Mario Alberto
3 Gozalez Cancalas Luis

(3 row(s) affected)
```

print '--Registros dentro de Amigos que existan dentro de Conocidos (INNER JOIN)'

```
SELECT dbo.Amigos.ApellidoPat, dbo.Amigos.ApellidoMat, dbo.Amigos.Nombres
FROM dbo.Amigos
INNER JOIN dbo.Conocidos
ON dbo.Amigos.ApellidoPat = dbo.Conocidos.ApellidoPat
AND dbo.Amigos.ApellidoMat = dbo.Conocidos.ApellidoMat
AND dbo.Amigos.Nombres = dbo.Conocidos.Nombres
```

```
ApellidoPat ApellidoMat Nombres
---------------- --------------- -----------
Villaneda Avila Sergio José
```

(1 row(s) affected)

Hay otra forma de hacerlo y es con la cláusula EXISTS y sería de la siguiente forma:

```
print '--Registros dentro de Amigos que existan dentro de Conocidos (EXITS)'
select t1.ApellidoPat, t1.ApellidoMat, t1.Nombres
from Amigos as t1
where exists
(select t2.ApellidoPat, t2.ApellidoMat, t2.Nombres
from Conocidos as t2
where t1.ApellidoPat=t2.ApellidoPat
and t1.ApellidoMat=t2.ApellidoMat
and t1.Nombres=t2.Nombres)

ApellidoPat ApellidoMat Nombres
---------- ---------- ----------
Villaneda Avila Sergio José

(1 row(s) affected)
```

Como ven arroja el mismo resultado?. Ahora, si queremos seleccionar los registros de la forma inversa, es decir, que estén dentro de la tabla Amigos pero no estén dentro de Conocidos, sería algo por el estilo: (también dejo la forma inversa, entre Conocidos y Amigos)

```
print '--Registros dentro de Amigos que no existan dentro de Conocidos'
select t1.ApellidoPat, t1.ApellidoMat, t1.Nombres
from Amigos as t1
where not exists
(select t2.ApellidoPat, t2.ApellidoMat, t2.Nombres
from Conocidos as t2
where t1.ApellidoPat=t2.ApellidoPat
and t1.ApellidoMat=t2.ApellidoMat
and t1.Nombres=t2.Nombres)

ApellidoPat ApellidoMat Nombres
---------- ---------- ----------
Villaneda Avila Andrea
Villaneda Avila Sarahi

(2 row(s) affected)

print '--Registros dentro de Conocidos que existan dentro de Amigos'
select t2.ApellidoPat, t2.ApellidoMat, t2.Nombres
from Conocidos as t2
where exists
(select t1.ApellidoPat, t1.ApellidoMat, t1.Nombres
from Amigos as t1
where t2.ApellidoPat=t1.ApellidoPat
and t2.ApellidoMat=t1.ApellidoMat
and t2.Nombres=t1.Nombres)

ApellidoPat ApellidoMat Nombres
---------- ---------- ----------
Villaneda Avila Sergio José

(1 row(s) affected)

print '--Registros dentro de Conocidos que no existan dentro de Amigos'
select t2.ApellidoPat, t2.ApellidoMat, t2.Nombres
```

```
from Conocidos as t2
where not exists
(select t1.ApellidoPat, t1.ApellidoMat, t1.Nombres
from Amigos as t1
where t2.ApellidoPat=t1.ApellidoPat
and t2.ApellidoMat=t1.ApellidoMat
and t2.Nombres=t1.Nombres)

ApellidoPat ApellidoMat Nombres
--------------- ----------------- ------------------------
Sanchez Gamboa Mario Alberto
Gozalez Cancalas Luis

(2 row(s) affected)
```