FBDSOLUCIONSQLCUADERNO.pdf

danielsp10

Fundamentos de Bases de Datos

2º Grado en Ingeniería Informática

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación
Universidad de Granada

```
/*CONSULTAS SQL - EJERCICIOS*/

/*EJERCICIO 3.2: Muestra el suministro de piezas realizado*/
SELECT DISTINCT codpj FROM ventas;

/*EJERCICIO 3.3: Piezas de Madrid que son grises o rojas*/
SELECT codpie FROM pieza WHERE ciudad='Madrid' and (color='Gris'
or color='Rojo');

/*EJERCICIO 3.4: Suminsitros cuya cantidad esta en [200,300]*/
SELECT * FROM ventas WHERE 200<=cantidad and cantidad<=300;

/*EJERCICIO 3.5: Mostrar las piezas que contengan la palabra tornilo
con la t en mayúscula o en minúscula*/
SELECT * FROM pieza WHERE nompie LIKE '_ornillo';

/*EJERCICIO 3.7: Encontrar los codigos de los proyectos a los que
sólo abastece S1 (usando interseccion)*/
SELECT * FROM ventas ORDER BY codpj;

/*EJERCICIO 3.8: Encontrar los códigos de los proyectos a los que
sólo abastece S1*/
SELECT DISTINCT codpj FROM ventas WHERE codpro='S1' MINUS SELECT
DISTINCT codpj FROM ventas WHERE codpro!='S1';

/*EJERCICIO 3.9: Mostrar todas las ciudades de la base de datos.
Utilizar UNION*/
SELECT ciudad FROM pieza UNION SELECT ciudad FROM proveedor UNION
SELECT ciudad FROM proyecto;

/*EJERCICIO 3.10: Mostrar todas las ciudades de la base de datos.
Utilizar UNION ALL */
SELECT ciudad FROM pieza UNION ALL SELECT ciudad FROM proveedor
UNION ALL SELECT ciudad FROM proyecto;

/*EJERCICIO 3.11: Comprueba cuantas tuplas resultan del producto
cartesiano entre ventas x proveedor*/
SELECT COUNT(*) FROM ventas,proveedor;

/*EJERCICIO 3.12: Ternas que son de la misma ciudad pero que hayan
realizdado alguna venta*/
SELECT codpro,codpie,codpj FROM ventas INTERSECT SELECT DISTINCT
codpro,codpie,codpj FROM proveedor,pieza,proyecto WHERE
proveedor.ciudad=pieza.ciudad AND
proveedor.ciudad=proyecto.ciudad AND pieza.ciudad=proyecto.ciudad;
```

/*EJERCICIO 3.13: Encontrar las parejas de proveedores que no viven en la misma ciudad*/

```
SELECT p.codpro,p.ciudad,s.codpro,s.ciudad FROM proveedor p,
proveedor s WHERE p.ciudad != s.ciudad AND (p.codpro < s.codpro);
```

/*EJERCICIO 3.14: Encuentra las piezas con máximo peso*/

```
SELECT peso FROM pieza MINUS (SELECT p1.peso FROM pieza p1,pieza
p2 WHERE p1.codpie <> p2.codpie AND p1.peso<p2.peso);
```

/*EJERCICIO 3.15: Mostrar las piezas vendidas por los proveedores de Madrid*/

```
SELECT DISTINCT codpie FROM ventas NATURAL JOIN (SELECT codpro
FROM proveedor WHERE ciudad='Madrid') ORDER BY codpie;
```

/*EJERCICIO 3.16: Encuentra la ciudad y los códigos de las piezas suministradas a cualquier proyecto por un proveedor que está en la misma ciudad donde está el proyecto*/

```
SELECT codpie,ciudad FROM pieza NATURAL JOIN (SELECT DISTINCT
codpie FROM ventas NATURAL JOIN
(SELECT codpro FROM proveedor NATURAL JOIN proyecto));
```

/*EJERCICIO 3.18: Listar las ventas ordenadas por cantidad, si algunas ventas coinciden en la cantidad se ordenan en función de la fecha de manera descendente*/

```
SELECT * FROM ventas ORDER BY cantidad, fecha DESC;
```

/*EJERCICIO 3.19: Mostrar las piezas vendidas por los proveedores de Madrid [Usando IN]*/

```
SELECT DISTINCT codpie FROM ventas WHERE codpro IN (SELECT codpro
FROM proveedor WHERE ciudad='Madrid') ORDER BY codpie;
```

/*EJERCICIO 3.20: Encuentra los proyectos que están en una ciudad donde se fabrica alguna pieza*/

```
SELECT codpj FROM proyecto NATURAL JOIN (SELECT ciudad FROM
pieza);
```

/*EJERCICIO 3.21: Encuentra los códigos de aquellos proyectos que no utilizan ninguna pieza roja que esté suministrada por un proveedor de Londres*/

```
SELECT codpj FROM proyecto WHERE NOT EXISTS(SELECT codpj FROM
ventas NATURAL JOIN
(SELECT codpie FROM pieza WHERE color='Rojo') NATURAL JOIN (SELECT
codpro FROM proveedor WHERE ciudad='Londres') WHERE
ventas.codpj=proyecto.codpj);
```

/*EJERCICIO 3.22: Muestra el código de las piezas cuyo peso es mayor que el peso de cualquier 'tornillo'*/

```
SELECT codpie FROM pieza WHERE peso > ANY (SELECT peso FROM pieza
WHERE nompie ='Tornillo');
/*EJERCICIO 3.23: Encuentra las piezas con peso máximo*/
SELECT codpie FROM pieza WHERE peso >= ALL (SELECT peso FROM
pieza);

/*EJERCICIO 3.24: Encontrar los códigos de las piezas suministradas
a todos los proyectos localizados en Londres*/
SELECT DISTINCT codpie FROM pieza WHERE NOT EXISTS(
    SELECT codpj FROM proyecto WHERE ciudad='Londres'
    MINUS
    SELECT codpj FROM ventas NATURAL JOIN (SELECT codpj FROM
proyecto WHERE ciudad='Londres')
    WHERE codpie=pieza.codpie
);

/*EJERCICIO 3.25: Encontrar aquellos proveedores que envían piezas
procedentes de todas las ciudades donde hay un proyecto*/
SELECT codpro FROM proveedor WHERE NOT EXISTS(
    SELECT codpie FROM pieza NATURAL JOIN (SELECT ciudad FROM
proyecto)
    MINUS
    SELECT codpie FROM ventas
    WHERE codpro=proveedor.codpro
);

/*EJERCICIO 3.26: Encontrar el número de envíos con más de 1000
unidades*/
SELECT COUNT(*) FROM ventas WHERE cantidad >1000;

/*EJERCICIO 3.27: Mostrar el máximo peso*/
SELECT MAX(peso) FROM pieza;

/*EJERCICIO 3.28: Mostrar el código de la pieza de máximo peso*/
SELECT codpie FROM pieza WHERE peso >= (SELECT MAX(peso) FROM
pieza);

/*EJERCICIO 3.30: Muestra los códigos de proveedores que han hecho
más de 3 envíos diferentes*/
SELECT codpro FROM ventas GROUP BY codpro HAVING COUNT(*) > 3;

/*EJERCICIO 3.31: Mostrar la media de las cantidades vendidas por
cada código de pieza junto con su nombre*/
SELECT nompie,ROUND(AVG(cantidad),4) FROM ventas NATURAL JOIN
(SELECT codpie,nompie FROM pieza) GROUP BY codpie,nompie;

/*EJERCICIO 3.32: Encontrar la cantidad media de ventas de la pieza
'P1' realizadas por cada proveedor*/
```

```
SELECT ROUND(AVG(COUNT(*)),4) FROM ventas WHERE codpie='P1' GROUP
BY codpro;
--Falta añadir a la media los proveedores que no venden P1.

/*EJERCICIO 3.33: Encontrar la cantidad total de cada pieza vendida
a cada proyecto*/
SELECT codpj,codpie,SUM(cantidad) FROM ventas GROUP BY
codpie,codpj ORDER BY codpj,codpie;

/*EJERCICIO 3.35: Mostrar los nombres de proveedores tales que el
total de sus ventas superen la cantidad de 1000 unidades*/
SELECT nompro FROM proveedor NATURAL JOIN (SELECT codpro FROM
ventas GROUP BY codpro HAVING SUM(cantidad) >= 1000);

/*EJERCICIO 3.36: Mostrar la pieza que más se ha vendido en total*/
SELECT codpie FROM ventas GROUP BY codpie HAVING SUM(cantidad) >=
(SELECT MAX(SUM(cantidad)) FROM ventas GROUP BY codpie);

/*EJERCICIO 3.38: Encontrar la cantidad media de piezas
suministradas cada mes*/
SELECT TO_CHAR(fecha,'MM/YYYY'),ROUND(AVG(cantidad),4) FROM ventas
GROUP BY TO_CHAR(fecha,'MM/YYYY');

/*EJERCICIO 3.42: Mostrar los codigos de aquellos proveeodres que
hayan superado las ventas totales realizadas por el proveedor 'S1'*/
SELECT codpro FROM ventas GROUP BY codpro HAVING SUM(cantidad) >
(SELECT SUM(cantidad) FROM ventas WHERE codpro='S1' GROUP BY
codpro);

/*EJERCICIO 3.43: Mostrar los mejores proveedores, entendiéndose
como los que tienen mayores cantidades totales.*/
SELECT codpro FROM ventas GROUP BY codpro HAVING SUM(cantidad) >=
(SELECT MAX(SUM(cantidad)) FROM ventas GROUP BY codpro);

/*-- Ejercicio 3.43.1: Mostrar los códigos de los proveedores con
las dos mayores cantidades de ventas.
1. Conseguimos la mayor cantidad de ventas por proveedor
2. Conseguimos la mayor cantidad de ventas que no sea igual a 1.
Vemos los proveedores cuya suma de cantidad es igual a cualquiera de
esas dos*/
SELECT codpro FROM ventas GROUP BY codpro HAVING SUM(cantidad)=
ANY(
SELECT MAX(SUM(cantidad)) FROM ventas GROUP BY codpro
UNION
SELECT MAX(SUM(cantidad)) FROM ventas GROUP BY codpro HAVING
SUM(cantidad) < (SELECT MAX(SUM(cantidad)) FROM ventas GROUP BY
codpro));
```

```
/*-- Ejercicio 3.43.2: Mostrar los códigos de los dos proveedores
con más ventas.
1. Cojo el codigo del proveedor con más ventas.
2. Cojo el código del proveedor con más ventas que no sea igual al
encontrado en 1.*/
SELECT codpro FROM ventas GROUP BY codpro HAVING SUM(cantidad) =
(SELECT MAX(SUM(cantidad)) FROM ventas GROUP BY codpro)

UNION
SELECT codpro FROM ventas GROUP BY codpro HAVING SUM(cantidad)=
    (SELECT MAX(SUM(cantidad)) FROM ventas WHERE codpro NOT IN
    (SELECT codpro FROM ventas GROUP BY codpro HAVING
SUM(cantidad) = (SELECT MAX(SUM(cantidad)) FROM ventas GROUP BY
codpro))
    GROUP BY codpro);

/*EJERCICIO 3.44: Mostrar los proveedores que venden piezas a todas
las ciudades de los proyectos a los que suministra 'S3', sin
incluirlo*/
SELECT codpro FROM proveedor p WHERE codpro <> 'S3' AND NOT EXISTS
(
    (SELECT DISTINCT ciudad FROM ventas NATURAL JOIN proyecto
WHERE codpro = 'S3' )
    MINUS
    (SELECT DISTINCT ciudad FROM proyecto NATURAL JOIN ventas
WHERE codpro = p.codpro));

/*EJERCICIO 3.45: Encontrar aquellos proveedores que hayan hecho al
menos 10 pedidos*/
SELECT codpro,COUNT(*) FROM ventas GROUP BY codpro HAVING COUNT(*)
>= 10;

/*EJERCICIO 3.46: Encontrar aquellos proveedores que venden todas
las piezas suministradas por S1*/
SELECT codpro FROM proveedor WHERE codpro <> 'S5' AND NOT EXISTS(
    SELECT codpie FROM ventas WHERE codpro='S5'
    MINUS
    SELECT codpie FROM ventas WHERE codpro=proveedor.codpro
);

/*EJERCICIO 3.47: Encontrar la cantidad total de piezas que ha
vendido cada proveedor que cumple la condicion de vender todas las
piezas suministradas por S1*/
SELECT codpro,SUM(cantidad) FROM ventas v1 WHERE codpro <> 'S5'
AND NOT EXISTS(
```

```
    SELECT codpie FROM ventas WHERE codpro='S5'
    MINUS
    SELECT codpie FROM ventas WHERE ventas.codpro=v1.codpro
)
GROUP BY codpro;

/*EJERCICIO 3.48: Encontrar qué proyectos están suministrados por
todos los proveedores que suministran la pieza P3*/
SELECT codpj FROM proyecto WHERE NOT EXISTS(
    SELECT DISTINCT codpro FROM ventas
    MINUS
    SELECT DISTINCT codpro FROM ventas WHERE codpie='P3' AND
codpj=proyecto.codpj
);

/*EJERCICIO 3.49: Encontrar la cantidad media de piezas suministrada
a aquellos proveedores que venden la pieza P3*/
SELECT codpro,ROUND(AVG(cantidad),4) FROM ventas NATURAL JOIN
(SELECT DISTINCT codpro FROM ventas
WHERE codpie='P3') GROUP BY codpro;

/*EJERCICIO 3.52: Mostrar para cada proveedor la media de productos
suministrados cada año*/
SELECT codpro,TO_CHAR(fecha,'YYYY'),ROUND(AVG(cantidad),4) FROM
ventas
GROUP BY codpro,TO_CHAR(fecha,'YYYY') ORDER BY
codpro,TO_CHAR(fecha,'YYYY');

/*EJERCICIO 3.53: Encontrar todos los proveedores que venden una
pieza roja*/
SELECT DISTINCT codpro FROM ventas NATURAL JOIN (SELECT codpie
FROM pieza WHERE color='Rojo');

/*EJERCICIO 3.54: Encontrar todos los proveedores que venden todas
las piezas rojas*/
SELECT codpro FROM proveedor WHERE NOT EXISTS(
    SELECT codpie FROM pieza WHERE color='Rojo'
    MINUS
    SELECT codpie FROM ventas WHERE ventas.codpro=proveedor.codpro
);

/*EJERCICIO 3.55: Encontrar todos los proveedores tales que todas
las piezas que venden son rojas*/
SELECT codpro FROM proveedor WHERE NOT EXISTS(
    SELECT DISTINCT codpie FROM ventas WHERE
ventas.codpro=proveedor.codpro
    MINUS
```

```
    SELECT DISTINCT codpie FROM ventas NATURAL JOIN (SELECT codpie
FROM pieza WHERE color='Rojo')
)
INTERSECT
SELECT codpro FROM ventas GROUP BY codpro HAVING COUNT(*) > 0;

/*EJERCICIO 3.56: Encontrar el nombre de aquellos proveedores que
venden mas de una pieza roja*/
SELECT nompro FROM proveedor WHERE codpro IN(
SELECT DISTINCT v1.codpro FROM (SELECT codpro,codpie FROM ventas
NATURAL JOIN (SELECT codpie FROM pieza WHERE color='Rojo')) v1,
(SELECT codpro,codpie FROM ventas NATURAL JOIN (SELECT codpie FROM
pieza WHERE color='Rojo')) v2
WHERE v1.codpro=v2.codpro AND v1.codpie!=v2.codpie);

/*EJERCICIO 3.57: Encontrar todos los proveedores que vendiendo
todas las piezas rojas cumplen la condicion de que todas sus ventas
son de mas de 10 unidades*/
SELECT DISTINCT codpro FROM ventas WHERE codpro IN(
    SELECT codpro FROM proveedor p WHERE NOT EXISTS(
        SELECT codpie FROM pieza WHERE color='Rojo'
        MINUS
        SELECT codpie FROM ventas v WHERE p.codpro = v.codpro
    )
)
AND codpro IN(
    SELECT DISTINCT codpro FROM ventas v1
    WHERE NOT EXISTS(
        SELECT * FROM ventas v2 WHERE v1.codpro=v2.codpro AND
cantidad<=10));

/*EJERCICIO 3.59: Encuentra, de entre las piezas que no se han
vendido en septiembre de 2009, las ciudades de aquellas que se han
vendido en mayor cantidad durante Agosto de ese mismo año.*/
SELECT ciudad FROM pieza WHERE codpie NOT IN(
    SELECT DISTINCT codpie FROM ventas WHERE
TO_CHAR(fecha,'MM/YYYY')='09/2009'
) AND codpie IN(
    SELECT codpie FROM ventas WHERE
TO_CHAR(fecha,'MM/YYYY')='08/2009'
    GROUP BY codpie HAVING SUM(cantidad) >= (SELECT
MAX(SUM(cantidad)) FROM ventas
    WHERE TO_CHAR(fecha,'MM/YYYY')='08/2009' GROUP BY codpie)
);


/*****************************************************************/
```

--EJERCICIOS ADICIONALES SQL - VENTAS

/*EJERCICIO 1: Proveedor que tiene el mayor número de ventas de la pieza P1 en el último año*/

```
SELECT codpro FROM ventas WHERE codpie='P1' AND
TO_CHAR(fecha,'YYYY')='2020' GROUP BY codpro
HAVING COUNT(*) >= (SELECT MAX(COUNT(*)) FROM ventas WHERE
codpie='P1' AND TO_CHAR(fecha,'YYYY')='2020'
GROUP BY codpro);
```

/*EJERCICIO 2: Piezas de color blanco que aparecen en, al menos, tres envíos con proveedores diferentes*/

```
SELECT DISTINCT codpie FROM ventas NATURAL JOIN (SELECT codpie
FROM pieza WHERE color='Blanco')
GROUP BY codpie HAVING COUNT(*)>=3;
```

/*EJERCICIO 3: Proyectos en los que los suministros en el año 2020 tienen una cantidad media superior a 150.*/

```
SELECT DISTINCT codpro FROM ventas WHERE
TO_CHAR(fecha,'YYYY')='2020' GROUP BY codpro HAVING
AVG(cantidad)>=150;
```

/*EJERCICIO 4: Proveedores con el número más alto de suministros a proyectos de Londres realizados durante el mes de enero de 2000.*/

```
SELECT codpro FROM ventas NATURAL JOIN (SELECT codpj FROM proyecto
WHERE ciudad='Londres') WHERE
TO_CHAR(fecha,'YYYY')='2020' GROUP BY codpro HAVING SUM(cantidad)
>= (SELECT MAX(SUM(cantidad)) FROM ventas
NATURAL JOIN (SELECT codpj FROM proyecto WHERE ciudad='Londres')
WHERE TO_CHAR(fecha,'YYYY')='2020' GROUP BY codpro);
```

/*EJERCICIO 5. Proveedores que han suministrado al menos tres piezas distintas a cada proyecto*/

```
SELECT codpro FROM ventas GROUP BY codpro,codpj HAVING COUNT(*) >=
3;
```

/*EJERCICIO 6: Piezas que aparecen en un único suministro durante el año 2010 (2013)*/

```
SELECT codpie FROM ventas WHERE TO_CHAR(fecha,'YYYY')='2013' GROUP
BY codpie HAVING COUNT(*)=1;
```

```
/*EJERCICIO 7: Piezas cuyo último suministro fue realizado en marzo
de 2010*/
SELECT codpie FROM ventas GROUP BY codpie HAVING
MAX(TO_CHAR(fecha,'MM/YYYY'))='03/2010';


/*EJERCICIO 8: Proyectos que reciben sólo tres piezas distintas de
proveedores de Londres*/
SELECT codpj FROM ventas NATURAL JOIN (SELECT codpro FROM
proveedor WHERE ciudad='Londres')
GROUP BY codpj HAVING COUNT(*)=3;

/*EJERCICIO 9: Proyectos que sólo tienen un proveedor con varios
suministros en el último año*/
SELECT codpj FROM ventas WHERE TO_CHAR(fecha,'YYYY')='2020' GROUP
BY codpj HAVING COUNT(*)>1
MINUS
SELECT A.codpj FROM ventas A,ventas B WHERE
    TO_CHAR(A.fecha,'YYYY')=TO_CHAR(B.fecha,'YYYY')
    AND
    TO_CHAR(A.fecha,'YYYY')='2020'
    AND
    A.codpro != B.codpro;

/*EJERCICIO 10: Piezas de color rojo que han sido suministradas al
menos dos veces cada año*/
SELECT codpie FROM pieza WHERE color='Rojo' AND NOT EXISTS(
    SELECT TO_CHAR(fecha,'YYYY') FROM ventas
    MINUS
    SELECT TO_CHAR(fecha,'YYYY') FROM ventas WHERE
ventas.codpie=pieza.codpie GROUP BY
    codpie,TO_CHAR(fecha,'YYYY') HAVING COUNT(*) >= 2
);

/*****************************************************************/

/*EJERCICIO 3.60: Muestra la información disponible acerca de los
encuentros de liga*/
SELECT * FROM encuentros;

/*EJERCICIO 3.61: Muestra los nombres de los equipos y de los
jugadores ordenados alfabéticamente*/
SELECT nombreE,nombreJ FROM jugadores NATURAL JOIN equipos ORDER
BY nombreE,nombreJ;

/*EJERCICIO 3.62: Muestra los jugadores que no tienen ninguna
falta*/
```

```
SELECT codj,nombrej FROM jugadores MINUS (SELECT codj,nombrej FROM
jugadores NATURAL JOIN faltas);

/*EJERCICIO 3.63; Muestra los compañeros de equipo del jugador que
tiene por codigo x (codJ='x') y donde 'x' es uno elegido por ti*/
SELECT * FROM jugadores WHERE codj LIKE 'A%';

/*EJERCICIO 3.64: Muestra los jugadores y la localidad donde juegan
(la de sus equipos)*/
SELECT nombreJ,localidad FROM jugadores NATURAL JOIN (SELECT codE,
localidad FROM equipos) ORDER BY localidad;

/*EJERCICIO 3.31: Muestra todos los encuentros posibles de la liga*/
SELECT e1.nombreE, e2.nombreE FROM equipos e1, equipos e2 WHERE
e1.codE <> e2.codE
AND e1.codE < e2.codE ORDER BY e1.nombreE,e2.nombreE;

/*EJERCICIO 3.65: Muestra los equipos que han ganado algún encuentro
jugando como local*/
SELECT DISTINCT Elocal FROM encuentros WHERE plocal > pvisitante;

/*EJERCICIO 3.66: Muestra los equipos que han ganado algún
encuentro*/
SELECT DISTINCT ELocal FROM encuentros WHERE plocal > pvisitante
UNION
SELECT DISTINCT EVisitante FROM encuentros WHERE plocal <
pvisitante;

/*EJERCICIO 3.69: Muestra los encuentros que faltan para terminar la
liga. Suponiendo que en la tabla Encuentros sólo se almacenan los
encuentros celebrados hasta la fecha.*/
SELECT e1.codE,e2.codE FROM equipos e1, equipos e2 WHERE e1.codE
<> e2.codE
MINUS SELECT ELocal,EVisitante FROM encuentros;

/*EJERCICIO 3.70: Muestra los encuentros que tienen lugar en la
misma localidad*/
SELECT ELocal,EVisitante,localidad FROM (SELECT
ELocal,EVisitante,Localidad FROM encuentros,(SELECT codE,localidad
FROM equipos) WHERE codE=ELocal),
(SELECT codE,localidad FROM equipos) WHERE codE=EVisitante;
SELECT EVisitante,ELocal,Localidad FROM encuentros,(SELECT
codE,localidad FROM equipos) WHERE codE=EVisitante;

/*EJERCICIO 3.71: Para cada equipo muestra cantidad de encuentros
que ha disputado como local*/
SELECT ELocal, COUNT(*) FROM encuentros GROUP BY elocal ORDER BY
elocal;
```

/*EJERCICIO 3.72: Muestra los encuentros en los que se alcanzó mayor diferencia*/

```
SELECT ELocal,EVisitante,ABS(MAX(PLocal) - MIN(PVisitante))
diferencia FROM encuentros GROUP BY Elocal,EVisitante
HAVING ABS(MAX(PLocal) - MIN(PVisitante)) = (SELECT
MAX(ABS(MAX(PLocal) - MIN(PVisitante))) FROM encuentros GROUP BY
ELocal,EVisitante);
```

/*EJERCICIO 3.73: Muestra los jugadores que no han superado 3 faltas acumuladas*/

```
SELECT codj,num FROM faltas WHERE num <= 3;
```

/*EJERCICIO 3.74: Muestra los equipos con mayores puntuaciones en los partidos jugados fuera de casa*/

```
SELECT EVisitante,MAX(PVisitante) FROM encuentros GROUP BY
EVisitante
HAVING MAX(PVisitante) >= (SELECT MAX(PVisitante) FROM
encuentros);
```

/*EJERCICIO 3.78: Muestra el equipo con mayor número de puntos en total de los encuentros jugados*/

```
SELECT DISTINCT * FROM (SELECT ELocal,SUM(PLocal) FROM encuentros
GROUP BY ELocal),(SELECT EVisitante,SUM(PVisitante) FROM
encuentros GROUP BY EVisitante)
WHERE ELocal=Evisitante;
```