

En la primera part del tema s'ha repassat tota la sintaxi de la sentència SELECT. Una vegada conegudes totes les clàusules, és moment de traure tota la potència de SQL accedint a Bases de Dades Relacionals, fent unes sentències "avançades", d'un nivell de complexitat major que fins ara.

Els tipus de sentències avançades que veurem seran:

- Combinacions de taules, en les quals entrarà en joc més d'una taula. Veurem les distintes possibilitats que tindrem.
- Subconsultes, on tindrem una sentència SQL dins d'una altra.
- Consultes d'unio, on unirem més d'una sentència SQL.

Vam veure en la sentència bàsica que en la clàusula FROM posàvem la taula **o taules** d'on s'agafarien les dades, però en tots els exemples posteriors només entrava en joc una única taula.

És el moment d'estudiar les diferents possibilitats que tindrem quan posem més d'una taula.

- La primera és el **producte cartesià**, que no utilitzarem mai, però hem de saber en què consisteix per poder evitar-lo.
- La segona serà la més utilitzada, la **combinació** (que de vegades anomenarem **reunió**).
- La tercera és una variant de l'anterior, la **combinació externa**, molt útil en alguns casos.

2.2.1 Producte cartesià

La manera més senzilla és posar les taules separades per comes, però segurament el resultat no és el que esperàvem.

Per exemple podem fer la següent sentència:

```
SELECT COMARQUES.nom_c, nom
FROM COMARQUES, POBLACIONS;
```

Nota

Observeu que hem posat el nom de la taula davant del camp **nom_c**, perquè les dues taules tenen un camp amb aquest nom. Aquesta operació s'anomena **qualificació**. Si no qualificàrem amb el nom de la taula davant, hi hauria ambigüitat, no sabria a quin camp es refereix, si el d'una taula o el de l'altra. Quan els noms dels camps són diferents i per tant no coincideixen en les dues taules, no cal qualificar el camp (com per exemple amb el camp **nom**)

Si executem la sentència, veurem que tindrem un nombre de files inesperadament alt, **18.428 files !!!** I si analitzem les files veurem el perquè: s'ha combinat cada comarca amb tots els pobles (siguen de la comarca o no).

The screenshot shows a web-based SQL editor interface. The top bar indicates the connection 'geo sobre geo@80.35.84.29:5432'. The 'Editor SQL' tab is active, displaying the query: `SELECT COMARQUES.nom_c, nom FROM COMARQUES, POBLACIONS;`. Below the editor, the 'Subfinestra de sortida' (Output Subwindow) is visible, showing the 'Sortida de dades' (Data Output) tab. The results are displayed in a table with two columns: 'nom_c' (character varying(50)) and 'nom' (character varying(50)). The table contains 12 rows of data, all with 'Ademuz' in the 'nom' column. At the bottom of the output window, a status bar shows 'OK.', 'Unix', 'Ln 2, Col 27, Ch 55', '18428 regis', and '11354 ms'.

	nom_c character varying(50)	nom character varying(50)
1	Safor	Ademuz
2	Horta Sud	Ademuz
3	Camp de Morvedr	Ademuz
4	Foia de Bunyol	Ademuz
5	Alacantí	Ademuz
6	Alt Maestrat	Ademuz
7	Plana Baixa	Ademuz
8	Horta Nord	Ademuz
9	Ports	Ademuz
10	Racó	Ademuz
11	Plana d'Utiel	Ademuz
12	Vinalopó Mitjà	Ademuz

Aquesta operació s'anomena **PRODUCTE CARTESIÀ (cross join)**, i es caracteritza en què cadascuna de les files d'una taula es combina amb totes les files de l'altra taula. El nombre de files resultant serà, doncs, el resultat de multiplicar el nombre de files d'una taula pel nombre de files de l'altra taula (en el nostre cas $34 \times 542 = 18428$).

Rarament voldrem fer un producte cartesià. El més normal serà combinar una miqueta millor les taules. En el nostre exemple segurament ens interessarà molt més combinar *cada comarca amb les seues poblacions*.

Combinació de dues taules: Sintaxi

Normalment el producte cartesià no ens interessarà. Més bé voldrem combinar les taules de manera que dos camps, un camp de cada taula, coincideixen. I el més habitual, si tenim la Base de Dades ben dissenyada, serà que els camps coincidents siguin una clau externa amb la clau principal a la qual apunta. Així, en l'exemple que utilitzàvem en el punt anterior, el que sí que ens serà útil és combinar cada comarca amb les seues poblacions. I justament tenim un camp en la taula POBLACIONS, **nom_c**, que és clau externa i apunta a la clau principal de **COMARQUES**.

Aquesta operació l'anomenarem **COMBINACIÓ INTERNA** o senzillament **COMBINACIÓ**, i de vegades també es diu **REUNIÓ**. La seua sintaxi és la següent:

```
SELECT ...  
FROM taula1 INNER JOIN taula2 ON condició
```

i on la condició de la reunió consistirà en comparar un camp de cada taula. Els dos camps hauran de ser del mateix tipus, però no caldrà que tinguen el mateix nom. Les files que eixiran al resultat seran les que acompliran la condició.

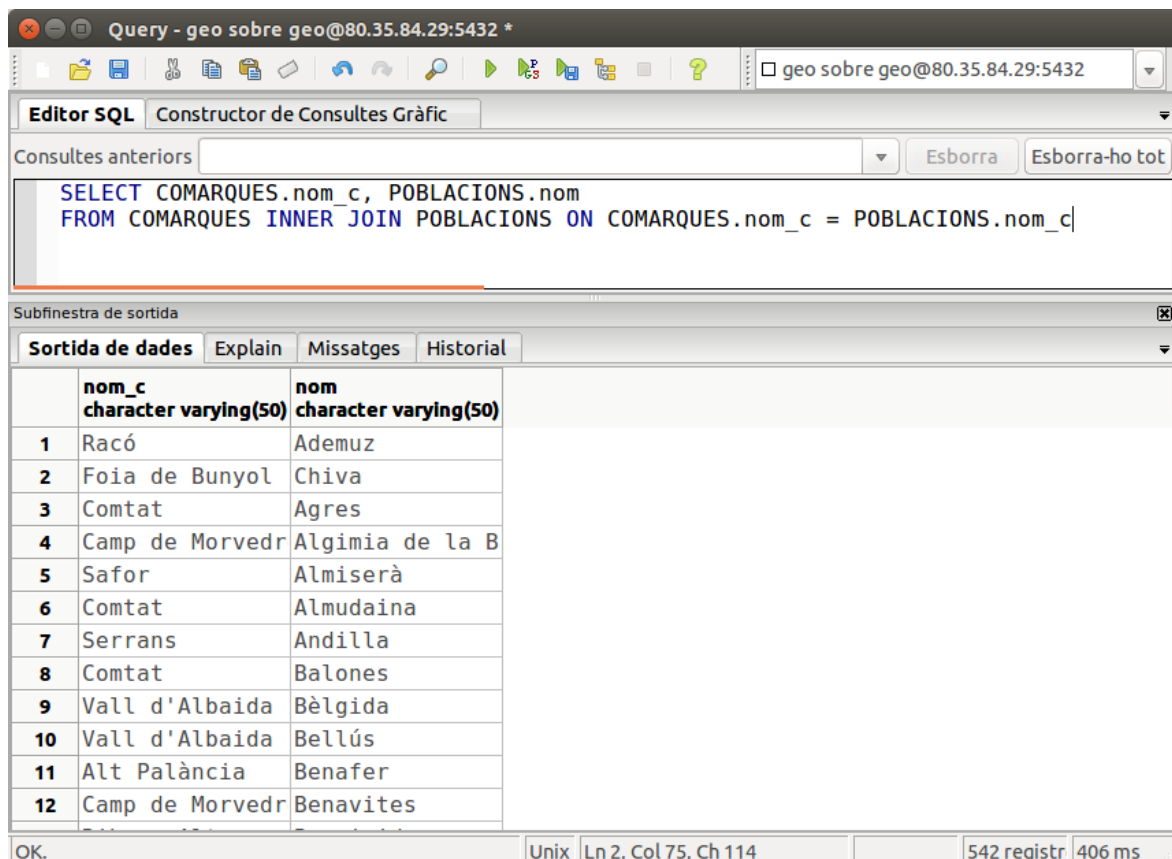
Encara que els operadors que es poden utilitzar són tots els de comparació, en la pràctica SEMPRE utilitzarem el d'igualar. Per tant podem refinar millor la combinació de 2 taules

```
SELECT ...  
FROM taula1 INNER JOIN taula2 ON taula1.camp1 = taula2.camp2
```

L'exemple de les comarques i les seues poblacions quedarà així

```
SELECT COMARQUES.nom_c, POBLACIONS.nom  
FROM COMARQUES INNER JOIN POBLACIONS ON COMARQUES.nom_c = POBLACIONS.nom_c
```

I aquest seria el resultat



The screenshot shows a database query tool window titled "Query - geo sobre geo@80.35.84.29:5432 *". The "Editor SQL" tab is active, displaying the query: `SELECT COMARQUES.nom_c, POBLACIONS.nom FROM COMARQUES INNER JOIN POBLACIONS ON COMARQUES.nom_c = POBLACIONS.nom_c`. Below the editor, the "Subfinestra de sortida" (Output Subwindow) shows the results in a table format. The table has two columns: "nom_c" (character varying(50)) and "nom" (character varying(50)). The results are listed in 12 rows, numbered 1 to 12. At the bottom of the window, it shows "OK.", "Unix Ln 2, Col 75, Ch 114", "542 registr" (records), and "406 ms" (execution time).

	nom_c character varying(50)	nom character varying(50)
1	Racó	Ademuz
2	Foia de Bunyol	Chiva
3	Comtat	Agres
4	Camp de Morvedr	Algimia de la B
5	Safor	Almiserà
6	Comtat	Almudaina
7	Serrans	Andilla
8	Comtat	Balones
9	Vall d'Albaida	Bèlgida
10	Vall d'Albaida	Bellús
11	Alt Palància	Benafer
12	Camp de Morvedr	Benavites

que com veiem torna 542 files (tantes com pobles)

Alternativament, podríem posar la mateixa reunió d'una altra forma:

```
SELECT COMARQUES.nom_c, POBLACIONS.nom
```

```
FROM COMARQUES, POBLACIONS
WHERE COMARQUES.nom_c = POBLACIONS.nom_c
```

on estrictament el que estem fent és, del producte cartesià de les 2 taules, seleccionar únicament quan coincideix el **nom_c** (és a dir la comarca amb les seues poblacions), i per tant el resultat seria el mateix. Potser fóra més eficient utilitzar la primera manera, però de vegades la comoditat ens farà utilitzar la segona (sobretot quan s'hagen de combinar moltes taules).

Les dues maneres de posar la combinació de dues taules són les més habituals, i que funcionen en qualsevol Sistema Gestor de Bases de Dades.

Tanmateix en PostgreSQL (i en altres SGBD com Oracle) hi ha més maneres de fer una combinació. No les veurem tan a fons perquè les anteriors ens basten i sobren:

- **INNER JOIN amb USING:** En el cas que els camps a reunir de les dues taules es diguen exactament igual, podem substituir la condició posada en **ON** per l'expressió **USING**, amb el camp de la reunió entre parèntesis

```
SELECT COMARQUES.nom_c, POBLACIONS.nom
FROM COMARQUES INNER JOIN POBLACIONS USING (nom_c)
```

- **NATURAL JOIN:** També per al cas anterior, en què el camp en les dues taules es diu igual, podem fer-lo de forma encara més abreviada. Farà una reunió, igualant tots els camps que es diguen igual de les dues taules. Hem d'anar en compte, per si de cas hi ha algun altre camp en les dues taules que es diga igual.

```
SELECT COMARQUES.nom_c, POBLACIONS.nom
FROM COMARQUES NATURAL JOIN POBLACIONS
```

Exemples

1. Traure el noms de les Poblacions i els noms dels Instituts que hi ha en elles.

Haurem de combinar les taules per la clau externa d'INSTITUTS a POBLACIONS (és a dir la que apunta de cod_m en INSTITUTS fins a la clau principal de COMARQUES, que és justament cod_m).

```
SELECT POBLACIONS.nom, INSTITUTS.nom
FROM POBLACIONS INNER JOIN INSTITUTS ON POBLACIONS.cod_m=INSTITUTS.cod_m;
```

Utilitzant l'altra sintaxi, que posem la condició en el **WHERE**, ens quedaria:

```
SELECT POBLACIONS.nom, INSTITUTS.nom
FROM POBLACIONS , INSTITUTS
WHERE POBLACIONS.cod_m=INSTITUTS.cod_m;
```

Com en aquest cas el camp que hem d'igualar té el mateix nom en les dues taules, utilitzant la sintaxi del **USING** ens quedari més fàcil:

```
SELECT POBLACIONS.nom, INSTITUTS.nom
FROM POBLACIONS INNER JOIN INSTITUTS USING (cod_m) ;
```

En canvi, hem d'anar amb molt de compte amb la sintaxi del **NATURAL JOIN**, perquè intentarà igualar tots els camps que es diuen igual, i en aquest cas tenim dos camps coincidents: cod_m i nom. cod_m és el que volem, però nom ens fastidiarà, i evidentment no coincideix mai el nom de l'Institut i el de la població, i per tant no tornarà cap fila.

```
SELECT POBLACIONS.nom, INSTITUTS.nom
FROM POBLACIONS NATURAL JOIN INSTITUTS;
```

2. Traure els noms de les comarques i la província, amb el nombre de poblacions que té cada comarca.

Ens fan falta dues taules, COMARQUES per a poder traure el nom de la comarca i la província, i POBLACIONS per a poder comptar els pobles de cada comarca. Les haurem de combinar, agrupar per comarca (i província també, perquè volem que aparega el nom de la província) i comptar les poblacions. A l'hora de comptar podem comptar files (COUNT(*)), però potser siga millor comptar algun camp de la taula POBLACIONS, per exemple cod_m, que és la clau principal (recordem que els valors nuls no es comptaran, i cod_m per ser clau principal no pot ser nul).

```
SELECT COMARQUES.nom_c, provincia, COUNT(cod_m) AS Quants
FROM COMARQUES INNER JOIN POBLACIONS ON COMARQUES.nom_c=POBLACIONS.nom_c
GROUP BY COMARQUES.nom_c, provincia;
```

Tres o més taules

Si tenim més de 2 taules, haurem de procedir de la mateixa manera, ja que si deixem de combinar alguna taula, tindrem el producte cartesià. Com en la immensa majoria de casos, la reunió la farem per les claus externes que tenim definides. Únicament haurem de cuidar els parèntesis, per a marcar primer una condició de combinació i després l'altra. En un exemple ho veurem perfectament il·lustrat.

Intentem traure el nom d'una comarca i la província, el nom dels seus pobles i el nom dels instituts d'aquests pobles. Ens fan falta les taules COMARQUES (pera poder traure el nom de la comarca i província), POBLACIONS (per a traure el nom de la població) i INSTITUTS (per al nom d'aquests). Ordenarem per nom de comarca, i dins d'aquest per població, per a una millor lectura del resultat

```
SELECT COMARQUES.nom_c, provincia, POBLACIONS.nom, INSTITUTS.nom
FROM (COMARQUES INNER JOIN POBLACIONS ON COMARQUES.nom_c=POBLACIONS.nom_c)
INNER JOIN INSTITUTS ON POBLACIONS.cod_m=INSTITUTS.cod_m
ORDER BY 1,3;
```

Podríem posar la consulta de la forma alternativa, en què les condicions de reunió van en el WHERE. Òbviament aquestes condicions han d'anar unides per l'operador AND.

```
SELECT COMARQUES.nom_c, provincia, POBLACIONS.nom, INSTITUTS.nom
FROM COMARQUES , POBLACIONS , INSTITUTS
WHERE COMARQUES.nom_c=POBLACIONS.nom_c AND POBLACIONS.cod_m=INSTITUTS.cod_m
ORDER BY 1,3;
```

Anem a plantejar un altre exemple. Es tracta de traure el nom i la província de les comarques, amb el número d'Instituts que hi ha en elles. En principi podríem pensar que les úniques taules que ens fan falta són COMARQUES (per a traure el nom i província de la comarca) i INSTITUTS (per a poder comptar els INSTITUTS). Si intentem fer aquesta consulta, **NO** obtindrem el resultat desitjat.

```
SELECT nom_c, provincia, COUNT(codi)
FROM COMARQUES , INSTITUTS
GROUP BY nom_c, provincia
```

Evidentment hi haurà un producte cartesià, ja que no hem combinat les taules, i ens eixirà per a cada comarca 375 instituts, que és el número total d'instituts, ja que s'ha combinat cada comarca amb tots els instituts.

Però aleshores, per quin camp combinem? Si intentem unir les claus principals, **nom_c** amb **codi** (el codi d'Institut) no poden combinar bé per raons evidents. Ens hem de fixar en el disseny de la Base de Dades. Observarem que el problema és que no hi ha una clau externa entre INSTITUTS i COMARQUES. Però també ens dóna la solució: **haurem de posar també la taula POBLACIONS** encara que no vulguem visualitzar cap camp d'aquesta taula, ja que si estan relacionades les taules INSTITUTS i COMARQUES és a través d'aquesta taula. Per tant la consulta correcta serà:

```
SELECT COMARQUES.nom_c, provincia, COUNT(codi)
FROM (COMARQUES INNER JOIN POBLACIONS ON COMARQUES.nom_c=POBLACIONS.nom_c)
INNER JOIN INSTITUTS ON POBLACIONS.cod_m=INSTITUTS.cod_m
GROUP BY COMARQUES.nom_c, provincia
```

La forma alternativa sembla més curta. Està clar que si són 3 taules, hauran d'haver 2 condicions de combinació unides per AND.

```
SELECT COMARQUES.nom_c, provincia, COUNT(codi)
FROM COMARQUES , POBLACIONS , INSTITUTS
WHERE COMARQUES.nom_c=POBLACIONS.nom_c AND POBLACIONS.cod_m=INSTITUTS.cod_m
GROUP BY COMARQUES.nom_c, provincia;
```

De forma general, si tenim *n* taules en una consulta, ens faran falta *n-1* condicions de combinació unides per AND. Per exemple, si en una consulta entren 5 taules, per a no tenir cap producte cartesià ens faran falta 4 condicions unides per AND.

Una taula més d'una vegada.

Anem a plantejar un altre exemple interessant: traure el nom de les poblacions, amb el nom de la capital de comarca. Lamentablement amb les dades que tenim en la Base de Dades d'exemple no podrem provar-lo, així que anem a fer una suposició, una Base de Dades lleugerament modificada per a il·lustrar aquest exemple.

Suposem que la nostra taula de POBLACIONS fóra lleugerament diferent, i que incorporara un camp nou amb el codi del municipi que és capital de comarca de la població:

```
POBLACIONS
(
  cod_m numeric(5,0) CONSTRAINT cp_pobl PRIMARY KEY,
  nom character varying(50) NOT NULL,
  poblacio numeric(6,0),
  extensio numeric(6,2),
  altura numeric(4,0),
  longitud character varying(50),
  latitud character varying(50),
  llengua character(1),
  nom_c character varying(50),
  cod_capital numeric(5,0) CONSTRAINT ce_capital REFERENCES POBLACIONS (cod_m)
)
```

Per a poder traure al mateix temps el nom de les poblacions i el nom de la seua capital de comarca no tenim prou amb posar la taula POBLACIONS una vegada: només trauríem el nom de la població i ens quedaríem amb el codi de municipi de la capital. La solució serà reunir-la amb la taula POBLACIONS, posant-la una segona vegada per a tenir dues instàncies de la taula, una instància per al població normal i una altra per a la capital. Però com distingirem entre les dues instàncies? Doncs posant un nom a cadascuna. En general podem posar un nom en la sentència a qualsevol taula que aparega, posant aquest nom a continuació de la taula (opcionalment podríem posar AS enmig):

```
SELECT ...
FROM taula T
```

En la resta de la consulta haurem d'utilitzar aquest nom. L'exemple quedarà de la següent manera:

```
SELECT P1.nom AS "Nom població" , P2.nom as "Nom capital"
FROM POBLACIONS P1 INNER JOIN POBLACIONS P2 ON P1.cod_capital=P2.cod_m
```

o de la forma alternativa:

```
SELECT P1.nom AS "Nom població" , P2.nom as "Nom capital"
FROM POBLACIONS P1, POBLACIONS P2
WHERE P1.cod_capital=P2.cod_m
```

Nota

Recordeu que aquestes instruccions no les podem provar, perquè no tenim el camp **cod_capital**.

Clau externa formada per més d'un camp

Per últim anem a considerar el cas que la clau externa estiga formada per més d'un camp. Ho basarem en l'exemple dels **Bancs**, on la taula COMPTE CORRENT depèn en identificació de SUCURSAL. Com la clau principal de SUCURSAL està formada per 2 camps, la clau externa de COMPTE CORRENT, que apunta a la primera també estarà formada per 2 camps. Si volem traure el número de compte corrent, el nom de la sucursal d'on és el compte, i el saldo, ens faran falta les dues taules. Aquesta seria la manera de combinar-les:

```
SELECT C.n_ent , C.n_suc , n_cc , S.nom , C.saldo
FROM SUCURSAL S INNER JOIN COMPTE_CORRENT C ON S.n_ent=C.n_ent AND S.n_suc=C.n_suc
```

o de la forma alternativa:

```
SELECT C.n_ent , C.n_suc , n_cc , S.nom , C.saldo
FROM SUCURSAL S, COMPTE_CORRENT C
WHERE S.n_ent=C.n_ent AND S.n_suc=C.n_suc
```

En ambdós casos s'ha optat per posar nom a les taules (S i C respectivament) per comoditat, per a que no quedara tan llarga la consulta.



Exercicis apartat 2.2.2

- 6.51** Traure el nom dels clients amb el número de factura i la data (individuals, sense agrupar res) que té cada client. Trau el resultat ordenat per client, i dins d'aquest per data de la factura
- 6.52** Traure el nom del soci, amb el codi i la descripció de cada article que ha demanat. Ordena per nom del soci i codi de l'article.
- 6.53** Modificar l'anterior per a que no es repetesquen els resultats.
- 6.54** Traure el nom dels clients amb la quantitat de factures que tenen, ordenades per aquest número de major a menor
- 6.55** Traure el número de factura, data, codi de client, total de la factura (amb l'àlies `IMPORT`) i total de la factura aplicant descomptes d'article (amb àlies `DESCOMPTE_1`). Tindrem el problema que el valor `NULL` és especial, i en operar amb qualsevol altre valor donarà `NULL`. En aquest cas clarament l'hem de considerar com un descompte 0. Podeu utilitzar una funció que substitueix els valors nuls trobats en el primer paràmetre, pel segon paràmetre d'aquesta manera: **`COALESCE(dte,0)`**. Ordena per número de factura.
- 6.56** Modificar l'anterior per a aplicar també el descompte de la factura (amb l'àlies `DESCOMPTE_2`)
- 6.57** Traure el codi i nom d'aquells venedors que supervisen algú (consten com a cap). Traure també el número de supervisats de cadascun d'aquests supervisors.
- 6.58** Traure el codi i descripció dels articles juntament amb el número de vegades que s'ha venut, el total d'unitats venudes i la mitjana d'unitats venudes per factura.
- 6.59** Traure el codi i la descripció de les categories, amb la quantitat d'articles venuts de cada categoria, d'aquelles categories de les quals se n'han venut més de 100 unitats. Ordenar per aquest número de forma decendent.

En ocasions ens interessarà fer una combinació diferent. Com quasi sempre ens basarem en un exemple. Quan en un exemple del punt anterior traïem els nom de les poblacions amb el nom dels instituts, no podien eixir les poblacions que no tenen instituts. Ara ens plantejarem la possibilitat de traure totes les poblacions, fins i tot aquelles que no tenen instituts, però d'aquelles que sí que en tinguen traure també el nom dels instituts. Aquesta operació s'anomena **COMBINACIÓ EXTERNA**.

Sintaxi

Tindrem dues possibilitats: traure totes les de l'esquerra o traure totes les de la dreta.

Per a traure TOTES les files de la taula de l'esquerra, i aquelles que estiguen relacionades de la de la dreta:

```
SELECT ...  
FROM taula1 LEFT [OUTER] JOIN taula2 ON condició
```

Així traurem TOTES les files de taula1, i aquelles que estiguen relacionades de taula2.

Per a fer-ho al revès, és a dir, totes les files de la taula de la dreta i aquelles files que estiguen relacionades de l'esquerra:

```
SELECT ...  
FROM taula1 RIGHT [OUTER] JOIN taula2 ON condició
```

D'aquesta manera traurem TOTES les files de taula2, i aquelles que estiguen relacionades de taula1.

En el nostre exemple:

```
SELECT POBLACIONS.nom, INSTITUTS.nom  
FROM POBLACIONS LEFT JOIN INSTITUTS ON POBLACIONS.cod_m=INSTITUTS.cod_m  
ORDER BY 1
```

on hem ordenat pel nom de la població per a una millor lectura, i ens donarà el següent resultat:

Query - geo on geo@172.16.1.2:5432 *

File Edit Query Favourites Macros View Help

Previous queries Delete Delete All

```
SELECT POBLACIONS.nom, INSTITUTS.nom
FROM POBLACIONS LEFT JOIN INSTITUTS ON POBLACIONS.cod_m=INSTITUTS.cod_m
ORDER BY 1
```

Output pane

Data Output Explain Messages History

	nom character varying(50)	nom character varying(60)
1	Ademuz	IES de ADEMUZ
2	Ador	
3	Agost	SECCIÓ de L'IES SAN VICENTE A AGOS
4	Agres	
5	Agullent	
6	Aielo de Malferit	IES PORÇONS
7	Aielo de Rugat	
8	Aigües	
9	Aín	
10	Alacant	IES LEONARDO DA VINCI
11	Alacant	IES LAS LOMAS
12	Alacant	IES JORGE JUAN
13	Alacant	IES JAIME II

OK. Unix Ln 3, Col 11, Ch 119 734 rows. 51 ms

Podem observar que ens trau fins i tot els pobles que no tenen instituts, i que en el camp nom de l'institut tenen el valor NULL.

Fem una variant interessant. Anem a traure els pobles amb el nombre d'instituts que té cadascun. Ens farà falta la taula POBLACIONS per a poder traure el nom de la població i la taula INSTITUTS per a traure el nombre d'instituts, i agruparem pel nom de la població. Les dues taules les hem de combinar (per evitar el producte cartesià). Si fem una combinació normal (interna), *els que no tenen instituts no entren*. Però si fem una **combinació externa sí que entraran**.

Només ens queda comptar per un camp que en el cas dels que no tenen instituts tinga el valor nul, és a dir, per un camp de la taula INSTITUTS, i el que millor se'ns acopla és algun que forma part de la clau principal, ja que com no pot agafar un valor nul en la taula INSTITUTS, l'única possibilitat que agafe el valor nul en la combinació externa és que la població no tinga institut, i aleshores en el moment de comptar ens donarà el valor 0.

Aquesta serà la consulta, on hem tornat a ordenar pel nom de la població

```
SELECT POBLACIONS.nom, COUNT(INSTITUTS.codi)
FROM POBLACIONS LEFT JOIN INSTITUTS ON POBLACIONS.cod_m=INSTITUTS.cod_m
GROUP BY POBLACIONS.nom
ORDER BY 1
```

I aquest serà el resultat

Query - geo on geo@172.16.1.2:5432 *

File Edit Query Favurites Macros View Help

Previous queries Delete Delete All

```

SELECT POBLACIONS.nom, COUNT(INSTITUTS.codi)
FROM POBLACIONS LEFT JOIN INSTITUTS ON POBLACIONS.cod_m=INSTITUTS.cod_m
GROUP BY POBLACIONS.nom
ORDER BY 1

```

Output pane

Data Output Explain Messages History

	nom character varying(50)	count bigint
1	Ademuz	1
2	Ador	0
3	Agost	1
4	Agres	0
5	Agullent	0
6	Aielo de Malferit	1
7	Aielo de Rugat	0
8	Aigües	0
9	Aín	0
10	Alacant	20
11	Alaquàs	2
12	Albaida	1

OK. Unix Ln 4, Col 11, Ch 151 542 rows. 20 ms

Una altra variant també interessant és fer una consulta similar per a traure els pobles que no tenen institut. Haurem de fer una combinació externa, i en la condició posar justament que un dels camps de la taula INSTITUTS siga nul (per exemple, la clau principal):

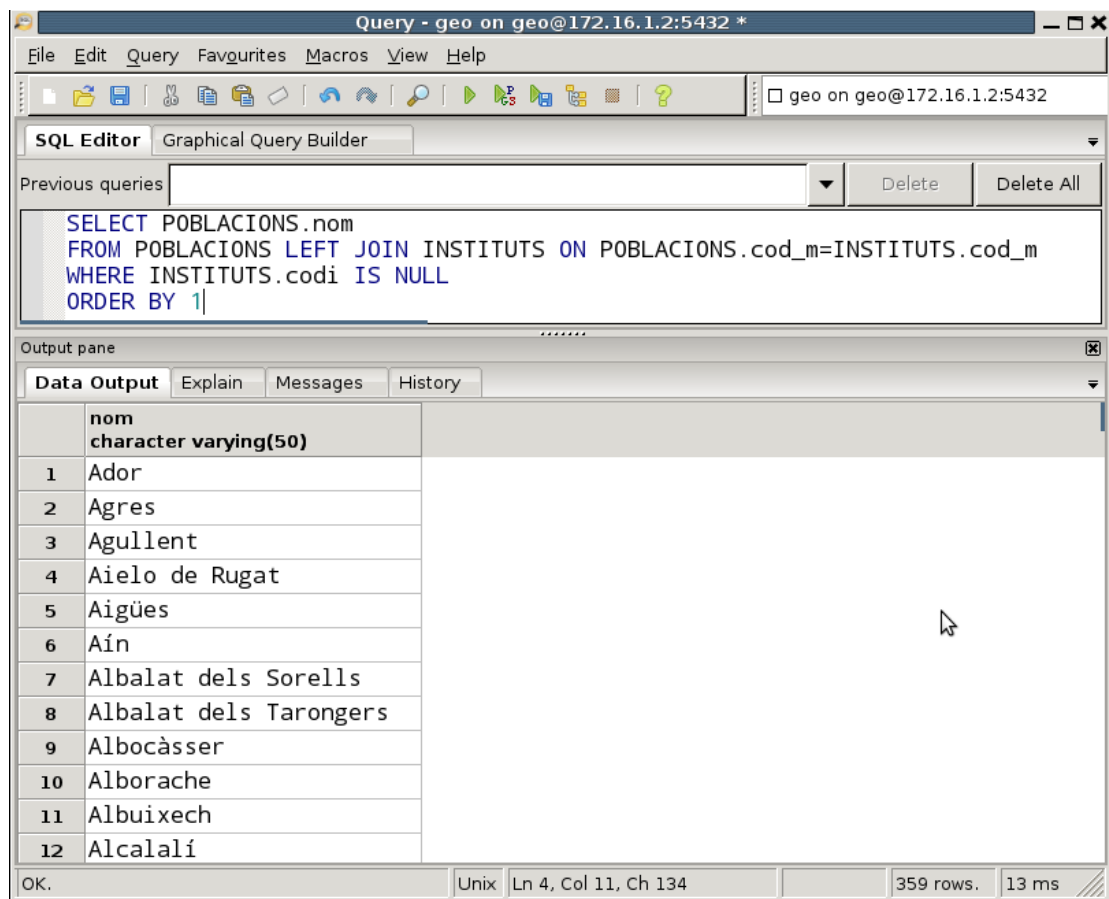
Aquesta serà la consulta, on hem tornat a ordenar pel nom de la població:

```

SELECT POBLACIONS.nom
FROM POBLACIONS LEFT JOIN INSTITUTS ON POBLACIONS.cod_m=INSTITUTS.cod_m
WHERE INSTITUTS.codi IS NULL
ORDER BY 1

```

I aquest serà el resultat:



Exemples

1. Traure totes les comarques amb el número de pobles que té cadascuna, fins i tot aquelles comarques que no tinguin cap poble.

Aquest exemple és poc il·lustratiu, perquè no tenim en principi cap comarca que no tinga pobles. De tota manera, la manera seria fent un LEFT JOIN entre COMARQUES i POBLACIONS, per a després agrupar per comarca i comptar les poblacions. Observeu com també podem utilitzar la sintaxi del **USING** en el LEFT JOIN.

```
SELECT COMARQUES.nom_c, COUNT(cod_m)
FROM COMARQUES LEFT JOIN POBLACIONS USING(nom_c)
GROUP BY COMARQUES.nom_c
ORDER BY 1;
```



Exercicis Apartat 2.2.3

6.60 Traure el codi i el nom dels clients que no tenen cap factura.

6.61 Traure el codi, descripció i total d'unitats venudes de tots els articles, fins i tot d'aquells que no s'ha venut res.

Nota

Per a deixar-lo més bonic, com que la suma de valors nuls no és 0 sinó nul, per a que ens aparegue el valor 0 podem utilitzar la funció `COALESCE(valor,0)`, que si el valor és nul torna un 0.

6.62 Traure el nom de tots els pobles i el número de clients en cas de que en tinguin. Ordena per número de clients de forma descendent.

6.63 Traure el codi i la descripció de les categories, amb el número d'articles de cada categoria i el número total d'unitats venudes de cada categoria, d'aquelles categories de les quals tenim més de 15 articles, i ordenat per número d'articles de forma descendent. Aquesta sentència ja és prou complicada. Concretament haureu de tenir en compte que:

- Voldrem traure el número d'articles de cada categoria, però potser alguns articles no s'han venut, i per tant no apareixeran en la taula `LINIA_FAC`.
- I també tenim el problema que, com ens fa falta la taula `LINIA_FAC`, un article venut en més d'una factura apareixerà més d'una vegada. Si comptem de forma normal, el comptaríem més d'una vegada cada article. Per tant voldrem comptar els diferents articles de cada categoria.

Una subconsulta és una consulta dins d'una altra consulta. Aquesta subconsulta pot tenir tots els elements que hem vist fins ara.

El lloc on posar una subconsulta dins de la consulta principal pot ser en la clàusula WHERE o en la clàusula HAVING (formant part d'una condició) o en el FROM, i ha d'anar entre parèntesis. Fins i tot es pot posar en el mateix SELECT, és a dir, en les columnes que van després del SELECT.

- Si va en el **FROM**, la subconsulta serà l'origen de les dades, i per tant s'executarà abans i proporcionarà les dades per a la consulta principal.
- Si va en el **WHERE** o el **HAVING** formarà part d'una condició, i així podrem comparar en la consulta principal un camp amb el que torne la subconsulta, per exemple. A banda de les comparacions normals que ja hem vist en el WHERE o el HAVING, podrem posar alguns operadors i predicats especials, com veurem més avant.
- Si va en el mateix **SELECT** normalment serà per a traure un resultat global que no afecta a la resta de la consulta

Sintaxi en el FROM

```
SELECT ...  
FROM ( Subconsulta ) AS Nom_Subconsulta
```

Anem a posar un exemple per entendre-ho. Volem traure la mitjana de pobles per comarca. Ho podem fer de la següent manera: primer comptem quants pobles hi ha en cada comarca, i una vegada calculat això traem la mitjana.

```
SELECT AVG(quants)  
FROM (SELECT COUNT(*) AS quants  
      FROM POBLACIONS  
      GROUP BY nom_c) AS S ;
```

Observeu que és necessari posar-li un àlies a la columna de la subconsulta (quants) per a poder fer referència a ella en la consulta principal. I per una altra banda, en PostgreSQL les subconsultes que van al FROM han de tenir obligatòriament un àlies. Si no posàrem ... **AS S** (o qualsevol altre nom) ens donaria error.

Com veieu, ja té un nivell de complexitat més que acceptable. Sempre s'executa primer la subconsulta i amb les dades que proporciona, s'executa la consulta principal. Pel grau de complexitat és molt recomanable anar de dins cap a fora, és a dir, pensar bé la subconsulta, fins i tot executar-la per veure si trau el que necessitem (en l'exemple veure si trau el nombre de poblacions de cadascuna de les 34 comarques), i quan estiguem segurs que funciona bé crear la consulta principal.

Sintaxi en el WHERE o el HAVING

```
SELECT ...  
FROM Taula  
WHERE camp operador ( Subconsulta )
```

Podem observar que el que farem serà comparar algun camp de la taula (o una expressió amb alguna funció) amb el resultat que ve de la subconsulta.

Abans de veure què podem posar com a operador o com a camp o fins i tot veure uns predicats que podrem utilitzar, posarem un exemple, per clarificar les coses. Intentarem traure les comarques amb una altura superior a la mitjana. Calcular la mitjana de les altures és fàcil, i serà la subconsulta. El que farem serà comparar l'altura de cada població amb aquesta mitjana.

```
SELECT *  
FROM POBLACIONS  
WHERE altura > (SELECT AVG(altura)  
                FROM POBLACIONS)
```

Si executeu la consulta, veureu que l'altura de tots els pobles és superior a 300,44 que és l'altura mitjana (aproximadament, perquè ho calcula amb molta precisió)

No hi ha cap problema en posar dues vegades la mateixa taula. Els camps es refereixen a la taula més propera. I funciona perfectament perquè la subconsulta ens torna un únic valor, la mitjana d'altures, i en la consulta principal es compara cada altura amb aquest valor.

Posteriorment veurem com solucionar el problema de que la subconsulta torne més d'un valor.

Els operadors de la condició poden ser de 3 tipus:

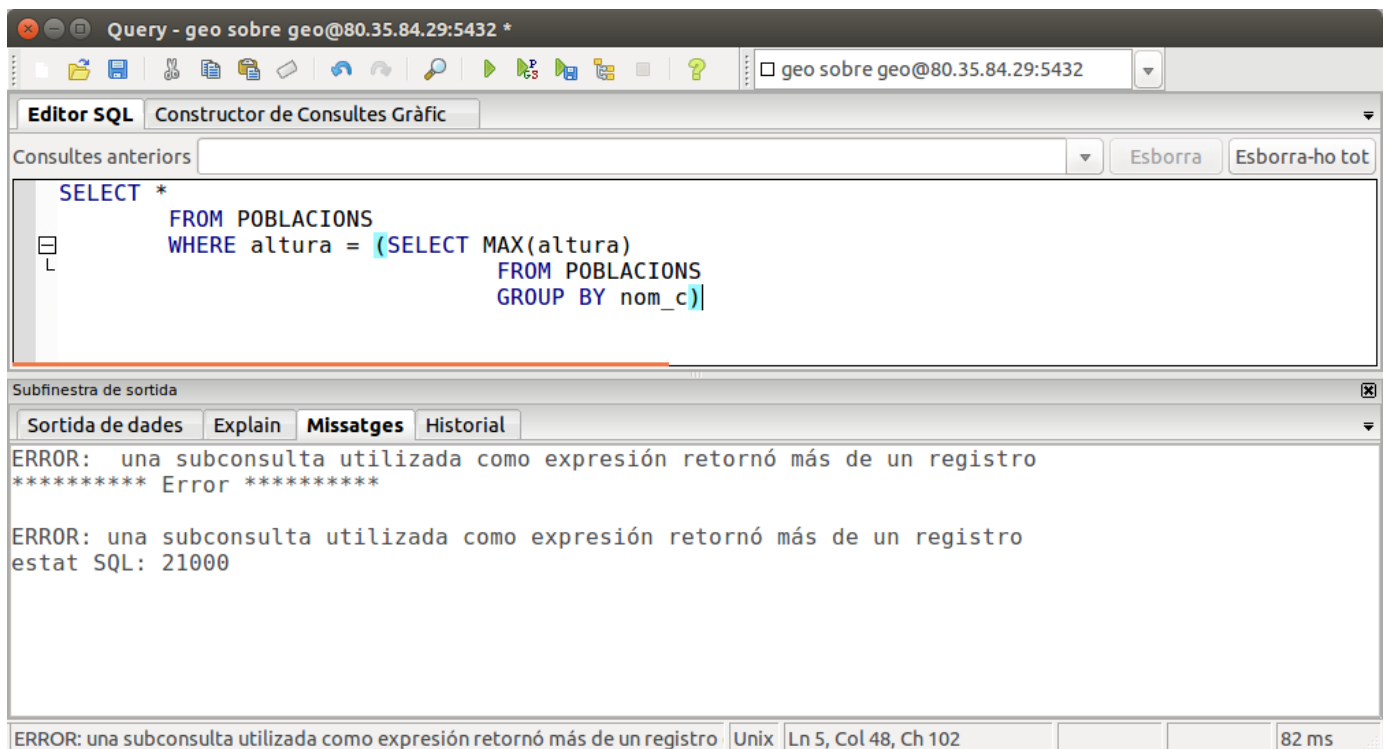
- **De comparació.** És com l'exemple de dalt, però amb qualsevol operador de comparació. Es compara el camp (o l'expressió) amb el resultat de la subconsulta. Si la subconsulta només torna un valor, no hi ha més problema, però si torna més d'un valor (més d'una fila) de moment seria incorrecte (no es pot comparar un camp amb uns quants valors). Posem un altre exemple per il·lustrar. Traure la població més alta es podria fer d'aquesta manera.

```
SELECT *
FROM POBLACIONS
WHERE altura = (SELECT MAX(altura)
                FROM POBLACIONS)
```

No hi ha problema perquè la subconsulta torna un valor. Però anem a complicar-la anem a veure les poblacions més altes de cada comarca. Podríem intentar-lo d'aquesta manera:

```
SELECT *
FROM POBLACIONS
WHERE altura = (SELECT MAX(altura)
                FROM POBLACIONS
                GROUP BY nom_c)
```

Però ens donaria el següent error:



I és que la subconsulta torna 34 valors (un per cada comarca), i d'aquesta manera no es pot comparar el valor de l'esquerra del igual amb els 34 valors de la dreta. Per a solucionar el problema de quan torna més d'un valor podem utilitzar els predicats **ALL**, **ANY**, **SOME**.

- Si utilitzem **ALL** el resultat serà cert si la comparació és certa amb **TOTS** els valors que torna la subconsulta.
- Si utilitzem **ANY** o **SOME** (que són sinònims) el resultat serà cert si la comparació és certa amb **ALGUN** valor de la subconsulta.

En el nostre exemple, segurament ens convindria **ANY**

```
SELECT *
FROM POBLACIONS
WHERE altura = ANY (SELECT MAX(altura)
                   FROM POBLACIONS
                   GROUP BY nom_c)
```

Aquesta consulta no funcionarà bé del tot, ja que seleccionarà totes les poblacions que coincideixen amb alguna de les altures màximes, siguin de la seua comarca o no. Així per exemple, l'altura màxima de la comarca de la **Plana Alta** és la **Serratella**, amb

781 metres, que efectivament apareix al llistat. Però també apareix **Castelló**, que té una altura de 30 metres. I apareix perquè l'altura màxima de la comarca **Ribera Baixa** dona la casualitat que és 30 metres (Almussafes). Ja dependrem a fer bé aquesta consulta en els exemples posteriors, però per al fet de comparar amb molts valors ens va bé.

- **L'operador IN.** No serà problema que la subconsulta torne un valor o molts. La condició serà certa si el valor del camp (o de l'expressió) està entre la llista de valors que torna la subconsulta. També poden utilitzar NOT IN, i la condició serà certa quan el valor del camp no està entre la llista. Per exemple, una altra manera de traure les poblacions que no tenen institut, que la vista en les combinacions externes. En la subconsulta traem els codis de municipi de la taula INSTITUTS, i per tant són els pobles que tenen institut, i en la consulta principal volem els que no estan en aquesta llista

```
SELECT *
FROM POBLACIONS
WHERE cod_m NOT IN (SELECT cod_m
                    FROM INSTITUTS)
```

- **L'operador EXISTS.** És segurament el més incòmode. No es compara un camp (o expressió) amb la subconsulta, sinó únicament es posa **[NOT] EXISTS (subconsulta)**. La condició serà certa si la subconsulta torna **alguna fila**, i no serà certa si no torna cap fila. Intentem fer el mateix exemple d'abans, el dels pobles sense institut. Hem d'aconseguir que la subconsulta no tinga cap fila en el cas dels que no tenen institut. De paraula ho podem dir així: volem els pobles per als quals no existeix cap fila en INSTITUTS amb el mateix codi de municipi. Ara ja es pot intuir per on van els tirs:

```
SELECT *
FROM POBLACIONS
WHERE NOT EXISTS (SELECT *
                  FROM INSTITUTS
                  WHERE cod_m= POBLACIONS.cod_m)
```

mireu com si en la subconsulta posem un camp (en l'exemple cod_m), si el camp és de la taula (o taules) de la subconsulta, es referirà a ell, per això si volem fer referència a un camp de la taula o taules de la consulta principal hem de posar el nom de la taula davant.

Sintaxi en el SELECT

```
SELECT ... ( Subconsulta )
FROM Taula
```

Anem a posar també un exemple per entendre-ho. Anem a calcular la diferència de l'altura de cada població amb la mitjana. La mitjana és un resultat global independent de la resta de la consulta, que en aquest cas és molt senzilla perquè hem d'agafar informació simple de les poblacions. La subconsulta també és molt senzilla, perquè només hem de calcular la mitjana d'altures).

```
SELECT nom, altura, altura - (SELECT AVG(altura) FROM POBLACIONS)
FROM POBLACIONS
```

Exemples

1. Traure l'altura mitjana de comarca més gran i la més menuda.

Ens fa falta prèviament l'altura mitjana de cada comarca, i això serà la subconsulta. No oblidem posar un àlies al camp de la subconsulta, per poder fer referència en la consulta principal. I no oblidem tampoc que les subconsultes en el FROM han de tenir àlies.

```
SELECT MAX(mitjana),MIN(mitjana)
FROM (SELECT AVG(altura) AS mitjana
      FROM POBLACIONS
      GROUP BY nom_c) AS S;
```

2. Traure tota la informació de les poblacions que tenen més de 5 instituts.

Podem pensar en una subconsulta on estiguen els codis de municipi de les poblacions que tenen més de 5 instituts (es consulta en la teula INSTITUTS agrupant per codi_m i comptant el número de files per a que siga major que 5).

```
SELECT *
FROM POBLACIONS
WHERE cod_m IN (SELECT cod_m
                FROM INSTITUTS
```

```
GROUP BY cod_m
HAVING count(*) > 5)
```

3. Traure tota la informació de la població més alta i de la més baixa.

Ens plantegem 2 subconsultes, la que trau l'altura màxima i la que trau l'altura mínima (en una única subconsulta ens tornaria valors en 2 columnes, i estaria més complicat). Senzillament serà traure tota la informació de les poblacions que tenen una altura igual al que torna una subconsulta o al que torna l'altra.

```
SELECT *
FROM POBLACIONS
WHERE altura = (SELECT MAX(altura)
                FROM POBLACIONS)
OR altura = (SELECT MIN(altura)
            FROM POBLACIONS);
```

4. Traure la població més alta de cada comarca.

La dificultat està en que ha de ser la màxima de les altures de la seua comarca. Per tant, en la subconsulta hem de fer referència a la comarca en qüestió. Com sempre estem tractant la taula POBLACIONS, tant en la consulta com en la subconsulta, haurem de posar un nom a la de la consulta principal, per poder fer referència a ella des de la subconsulta.

```
SELECT *
FROM POBLACIONS T1
WHERE altura = (SELECT MAX(altura)
                FROM POBLACIONS
                WHERE nom_c= T1.nom_c);
```

En el resultat obtenim 35 poblacions, quan només hi ha 34 comarques. La raó és que a l'Alcoià, hi ha 2 poblacions amb l'altura màxima (816 metres), per tant el resultat és correcte

5. Obtenir el nom de la comarca i la província de les comarques que tenen una altura mitjana més alta que la mitjana de totes les poblacions.

La subconsulta serà prou senzilla: la mitjana d'altures de les poblacions. En la consulta principal haurem d'agrupar per cada comarca i calcular la mitjana d'altures de les poblacions, i comparar-la amb la mitjana que ens ve de la subconsulta. Com que demana també la província, ens fa falta també la taula COMARQUES, i per tant l'haurem de reunir amb POBLACIONS; en aquesta ocasió ho hem fet posant la condició en el WHERE.

```
SELECT COMARQUES.nom_c, provincia, AVG(altura)
FROM COMARQUES , POBLACIONS
WHERE COMARQUES.nom_c=POBLACIONS.nom_c
GROUP BY COMARQUES.nom_c, provincia
HAVING AVG(altura) > (SELECT AVG(altura)
                     FROM POBLACIONS)
```

6. Traure el nom de les comarques, el número de municipis i el percentatge que suposa respecte al total de municipis.

Tota la informació la podem traure d'una reunió entre les taules COMARQUES i POBLACIONS (per a comptar quants pobles hi ha en cada comarca), però per a poder calcular el percentatge necessitem el número total de poblacions, que el podem calcular amb una senzilla subconsulta. El lloc més còmode és en el SELECT. La reunió l'hem feta en aquesta ocasió amb el USING.

```
SELECT COMARQUES.nom_c, provincia, COUNT(cod_m), COUNT (cod_m)*100.0/(SELECT COUNT(*) FROM
POBLACIONS)
FROM COMARQUES INNER JOIN POBLACIONS USING(nom_c)
GROUP BY 1,2;
```



Exercicis apartat 2.3

6.64 Traure el número màxim de factures fetes a un client

6.65 Traure el l'import que suposa la factura més cara i l'import que suposa la més barata (sense considerar ni descomptes ni IVA)

6.66 Traure el número de factures més alt que s'ha venut per venedor en cada trimestre (no traurem qui és el venedor, que seria encara més complicat). Per a poder agrupar per trimestre, ens farà falta la funció **TO_CHAR(data,'Q')**, que trau el número de trimestre. El pas previ és calcular el número de factures de cada venedor i en cada trimestre. Després, amb la informació anterior, voldrem calcular el màxim de cada trimestre.

6.67 Traure els articles més cars que la mitjana. Tragueu-los ordenats per la categoria, i després per codi d'article.

6.68 Modificar l'anterior per a traure els articles més cars que la mitjana de la seua categoria. Tragueu-los ordenats per la categoria

6.69 Traure els pobles on tenim clients però no tenim venedors. Ha de ser per mig de subconsultes (en plural). Ordeneu per codi del poble.

2.4 Consultes d'operacions de conjunts

Agruparem en aquest apartat les consultes que tracten conjunts de files per a fer operacions d'àlgebra de conjunts: **unió**, **intersecció** i **diferència** de conjunts

Toters aquestes consultes ajunten els resultats de dues o més consultes.

Sintaxi de la UNIÓ

```
[TABLE] consulta1
UNION [ALL]
[TABLE] consulta2 ...
```

Cadascuna de les consultes pot ser una taula (posant la paraula **TABLE** davant) o el nom d'una consulta ja guardada, encara que el més habitual serà posar directament la **sentència SQL**.

Els requisits són que les dues (o més) consultes tornen el mateix nombre de camps, i que siguin sinó del mateix tipus, sí de tipus compatibles

Igual que en la unió de conjunts, el resultat seran totes les files de les dues (o més) consultes individuals, però sense repetir files, és a dir, si de les dues consultes s'obtenen files iguals, aquestes només eixiran una vegada. L'anterior es pot evitar si posem el predicat **ALL**, i aleshores sí que eixiran les files repetides.

Els noms dels camps vindran donats per la primera consulta.

Si volem ordenar per algun camp, ho haurem de posar al final de l'última consulta, però referint-se en tot cas als camps de la primera consulta (ho podem evitar posant el número d'ordre en el **ORDER BY**)

Exemple

1. Volem veure en un únic resultat tant el nom de les comarques com el nom de les poblacions, sempre amb el nom de la província al costat

```
SELECT nom_c, provincia
  FROM CÒMARQUES
UNION
SELECT nom, provincia
  FROM COMARQUES INNER JOIN POBLACIONS USING (nom_c)
ORDER BY nom_c;
```

Com a curiositat, eixiran 575 files, però si posàrem **UNION ALL** ens eixirien 576. Això és perquè la comarca de la ciutat de València es diu València i està a la província de València. Per tant és una fila que apareixerà tant en la primera com en la segona consulta. Si fem **UNION** no es repetirà, però si fem **UNION ALL** sí que es repetirà.

Sintaxi de la INTERSECCIÓ

És idèntica a la unió, però posant la paraula **INTERSECT**, i servirà per a traure únicament les files que estan en les dues consultes.

```
[TABLE] consulta1
INTERSECT [ALL]
[TABLE] consulta2 ...
```

Igual que abans, cadascuna de les consultes pot ser una taula (posant la paraula **TABLE** davant), i tenim el requisit que les dues (o més) consultes tornen el mateix nombre de camps, i de tipus compatibles.

En principi no eixiran files repetides, a no ser que posem **ALL**

Exemple

1. Com que en l'exemple de la unió havíem vist que la fila València València eixia en les 2 consultes, anem a comprovar que apareix en la intersecció:

```
SELECT nom_c, provincia
  FROM CÒMARQUES
INTERSECT
SELECT nom, provincia
  FROM COMARQUES INNER JOIN POBLACIONS USING (nom_c)
ORDER BY nom_c;
```

Sintaxi de la DIFERÈNCIA

És idèntica a les anteriors, però posant la paraula **EXCEPT**, i servirà per a traure les files que estan en la primera consulta però que no estan en la segona.

```
[TABLE] consulta1  
EXCEPT [ALL]  
[TABLE] consulta2 ...
```

Igual que abans, cadascuna de les consultes pot ser una taula (posant la paraula **TABLE** davant), i tenim el requisit que les dues (o més) consultes tornen el mateix nombre de camps, i de tipus compatibles.

En principi no eixiran files repetides, a no ser que posem **ALL**

Exemple

1. Aprofitem el mateix exemple d'abans per a comprovar que amb **EXCEPT** no eixirà la comarca València, ja que hi ha una fila idèntica en la segona consulta:

```
SELECT nom_c, provincia  
FROM CÒMARQUES  
EXCEPT  
SELECT nom, provincia  
FROM CÒMARQUES INNER JOIN POBLACIONS USING (nom_c)  
ORDER BY nom_c;
```



Exercicis apartat 2.4

6.70 Traure el nom de tots els clients i venedors implicats en alguna venda del primer trimestre de 2015.

6.71a Traure per mig de sentències d'operacions de conjunts els pobles on tenim algun venedor o algun client. No volem resultats repetits, i ho volem ordenat pel nom del poble.

6.71b Modificar l'anterior per a traure els pobles on tenim al mateix temps venedors i clients

6.71c Modificar l'anterior per a traure els pobles on tenim venedors però no tenim clients

Exercicis de tot el tema, amb els resultats

6.51 Traure el nom dels clients amb el número de factura i la data (individuals, sense agrupar res) que té cada client. Trau el resultat ordenat per client, i dins d'aquest per data de la factura

	nom character varying(100)	num_f numeric(5,0)	data date
1	ADELL GALMES, MERCEDES ROSARIO	6675	2015-09-04
2	ADELL VILLALONGA, LUIS JOSE	6586	2015-04-10
3	ADELL VILLALONGA, LUIS JOSE	6688	2015-09-21
4	ADELL VILLALONGA, LUIS JOSE	6736	2015-12-05
5	AMO MONTSN, RAMON FERNANDO	6550	2015-02-03
6	AMO MONTSN, RAMON FERNANDO	6678	2015-09-07
7	BADENES CEPRIA, ANDRES RICARDO	6584	2015-04-07
8	BELTRAN MENEU, CRISTINA	6604	2015-05-04
9	BELTRAN MENEU, CRISTINA	6633	2015-06-30
10	BELTRAN MUNYOZ, JAIME VICENTE	6674	2015-09-02
11	BELTRAN MUNYOZ, JAIME VICENTE	6670	2015-09-07

Un total de **105** files

6.52 Traure el nom del soci, amb el codi i la descripció de cada article que ha demanat. Ordena per nom del soci i codi de l'article.

	nom character varying(100)	cod_a character varying(10)	descrip character varying(50)
1	ADELL GALMES, MERCEDES ROSARIO	L85457	Marco Legrand 2
2	ADELL GALMES, MERCEDES ROSARIO	L85685	Tecla Base Legra
3	ADELL GALMES, MERCEDES ROSARIO	LAPC	Reactancia 40 W
4	ADELL GALMES, MERCEDES ROSARIO	LAR40125	Bloque Emergenci
5	ADELL GALMES, MERCEDES ROSARIO	TC23	Tubo Fercondur
6	ADELL GALMES, MERCEDES ROSARIO	TNF21	Tubo Sapa 11
7	ADELL GALMES, MERCEDES ROSARIO	ZN5108B	Base Plastimetal
8	ADELL VILLALONGA, LUIS JOSE	CN1X10	Paralelo 2 X 1
9	ADELL VILLALONGA, LUIS JOSE	IM3P15V	Interruptor Magn
10	ADELL VILLALONGA, LUIS JOSE	IMFN25L	Cartucho Fusible
11	ADELL VILLALONGA, LUIS JOSE	IMFN25L	Cartucho Fusible
12	ADELL VILLALONGA, LUIS JOSE	L58068	Base Enchufe T.t

Un total de **541** files

6.53 Modificar l'anterior per a que no es repetesquen els resultats.

	nom character varying(100)	cod_a character varying(10)	descrip character varying(50)
1	ADELL GALMES, MERCEDES ROSARIO	L85457	Marco Legrand 2
2	ADELL GALMES, MERCEDES ROSARIO	L85685	Tecla Base Legra
3	ADELL GALMES, MERCEDES ROSARIO	LAPC	Reactancia 40 W
4	ADELL GALMES, MERCEDES ROSARIO	LAR40125	Bloque Emergenci
5	ADELL GALMES, MERCEDES ROSARIO	TC23	Tubo Fercondur
6	ADELL GALMES, MERCEDES ROSARIO	TNF21	Tubo Sapa 11
7	ADELL GALMES, MERCEDES ROSARIO	ZN5108B	Base Plastimetal
8	ADELL VILLALONGA, LUIS JOSE	CN1X10	Paralelo 2 X 1
9	ADELL VILLALONGA, LUIS JOSE	IM3P15V	Interruptor Magn
10	ADELL VILLALONGA, LUIS JOSE	IMFN25L	Cartucho Fusible
11	ADELL VILLALONGA, LUIS JOSE	L58068	Base Enchufe T.t
12	ADELL VILLALONGA, LUIS JOSE	L76179	Placa 1 F. Legra

Un total de **532** files

Observa com ara no es repeteix la fila 10 i 11, i abans sí

6.54 Traure el nom dels clients amb la quantitat de factures que tenen, ordenades per aquest número de major a menor

	nom character varying(100)	Número de factures bigint
1	LOPEZ RINCON, LUIS MIGUEL	5
2	BOTELLA CATALA, JUAN	5
3	PINEL HUERTA, VICENTE	4
4	HERRERA SALA, ANA	4
5	MIGUEL ARCHILES, OSCAR RAMON	4
6	VILLALONGA RAMIREZ, DIEGO SERGIO	4
7	NAVARRO BARBERO, MARIA LLEDO	4
8	HUGUET PERIS, JUAN ANGEL	4
9	CANCELAS MORA, MARIA	3
10	GALLEN HUERTA, OLGA	3
11	LOPEZ GUITART, YAVIER	3

Un total de 40 files

6.55 Traure el número de factura, data, codi de client, total de la factura (amb l'àlies IMPORT) i total de la factura aplicant descomptes d'article (amb àlies DESCOMPTE_1), com en la consulta **6.33**, però sense el límit de les 10 línies de factura. Ordena per número de factura.

	num_f numeric(5,0)	data date	cod_cli numeric(5,0)	import numeric	descompte_1 numeric
1	6535	2015-01-01	306	11.78	8.83500000
2	6538	2015-01-11	345	50.30	37.17000000
3	6539	2015-01-14	357	234.59	234.59000000
4	6542	2015-01-18	375	209.82	209.82000000
5	6547	2015-01-26	72	220.32	176.25600000
6	6549	2015-02-01	315	972.43	972.43000000
7	6550	2015-02-03	261	669.87	502.40250000
8	6552	2015-02-10	78	102.07	93.14700000
9	6553	2015-02-10	387	501.11	501.11000000
10	6554	2015-02-12	306	0.90	0.67500000
11	6557	2015-02-15	318	49.76	39.80800000

Un total de 105 files

6.56 Modificar l'anterior per a aplicar també el descompte de la factura (amb l'àlies DESCOMPTE_2)

	num_f numeric(5,0)	data date	cod_cli numeric(5,0)	import numeric	descompte_1 numeric	descompte_2 numeric
1	6535	2015-01-01	306	11.78	8.83500000	7.95150000
2	6538	2015-01-11	345	50.30	37.17000000	33.45300000
3	6539	2015-01-14	357	234.59	234.59000000	175.94250000
4	6542	2015-01-18	375	209.82	209.82000000	188.83800000
5	6547	2015-01-26	72	220.32	176.25600000	176.25600000
6	6549	2015-02-01	315	972.43	972.43000000	777.94400000
7	6550	2015-02-03	261	669.87	502.40250000	401.92200000
8	6552	2015-02-10	78	102.07	93.14700000	93.14700000
9	6553	2015-02-10	387	501.11	501.11000000	400.88800000
10	6554	2015-02-12	306	0.90	0.67500000	0.60750000
11	6557	2015-02-15	318	49.76	39.80800000	19.90400000

Un total de 105 files

6.57 Traure el codi i nom d'aquells venedors que supervisen algú (consten com a cap). Traure també el número de supervisats de cadascun d'aquests supervisors.

	cod_ven numeric(5,0)	nom character varying(100)	count bigint
1	5	GUILLEN VILLALONG	2
2	105	POY OMELLA, PALOM	3
3	255	DANIEL MIRALLES,	2
4	405	ROCA FAURA, ANTON	2

6.58 Traure el codi i descripció dels articles juntament amb el número de vegades que s'ha venut, el total d'unitats venudes i la mitjana d'unitats venudes per factura. Ordenar pel número total d'unitats venudes de forma descendent, i dins d'aquesta per codi d'article de forma ascendent.

	cod_a character varying(10)	descrip character varying(50)	count bigint	sum numeric	avg numeric
1	L76104	Pulsador Pus Leg	4	36	9.0000
2	L16500	Cartucho Fusible	3	26	8.6666
3	LA2760EC	Reactancia Vapor	3	26	8.6666
4	L76138	Pieza 4 E. Legra	3	25	8.3333
5	L85393	Toma Desplazada	3	24	8.0000
6	B10010B	Interruptor Bjc	2	23	11.500
7	L85510	Marco Legrand 2	2	23	11.500
8	N5088	Cortacircuitos N	2	23	11.500
9	ZN5104B	Base Enchufe Tt	3	23	7.6666
10	L55812	Interruptor 2 M.	3	22	7.3333
11	L85685	Tecla Base Legra	3	22	7.3333

Un total de **399 files**

6.59 Traure el codi i la descripció de les categories, amb la quantitat d'articles venuts de cada categoria, d'aquelles categories de les quals se n'han venut més de 100 unitats. Ordenar per aquest número de forma descendent.

	cod_cat character varying(15)	descripcio character varying(50)	sum numeric
1	Legrand	Components marca	887
2	Niessen	Components Niese	201
3	IntMagn	Interruptor Magn	182
4	Simon	Components marca	148
5	Ticino	Components marca	117

6.60 Traure el codi i el nom dels clients que no tenen cap factura.

	cod_cli numeric(5,0)	nom character varying(100)
1	3	DAMBORENEA CORBATO, ALICIA
2	15	GUAL SALES, MARIA
3	120	CASTELLO DAMBORENEA, ENRIQUE JAVIER
4	282	MORA RIBES, ENRIQUE MIGUEL
5	330	MARTI MOLTO, CONCHITA
6	372	LOPEZ LLORENS, SANCHEZ MARCOS
7	378	GUIMERA AGOST, LUIS
8	381	GUILLOT BELDA, FRANCISCO JOSE
9	390	AZNAR MONFERRER, ADRIAN

6.61 Traure el codi, descripció i total d'unitats venudes de tots els articles, fins i tot d'aquells que no s'ha venut res.

Nota

Per a deixar-lo més bonic, com que la suma de valors nuls no és 0 sinó nul, per a que ens aparegui el valor 0 podem utilitzar la funció COALESCE(valor,0), que si el valor és nul torna un 0.

	cod_a character	descrip character varying(50)	sum numeri
1	1967346	Prolongador Doble 2.5 M	8
2	2309987	Prolongador Doble 5.0 M	4
3	3204209	Prolongador Cuaduple 2.5 M	0
4	3987348	Prolongador Cuaduple 20.0 M	4
5	4283200	Prolongador Doble 10.0 M	6
6	4565467	Prolongador Sencillo 10.0 M	0
7	6579878	Prolongador Sencillo 5.0 M	0
8	7897999	Prolongador Sencillo 2.5 M	0
9	8340098	Prolongador Cuaduple 5.0 M	8
10	8394800	Prolongador Cuaduple 10.0 M	2
11	A03755	Tubo Fluorescente 15 W	0

Un total de **812 files**

6.62 Traure el nom de tots els pobles i el número de clients en cas que en tinguin. Ordena per número de clients de forma descendent.

	cod_pob numeric(5,0)	nom character varying(50)	count bigint
1	53596	VILLARREAL	10
2	12309	CASTELLON	9
3	7766	BURRIANA	7
4	32093	NULES	4
5	48037	TORO (EL)	1
6	2050	ALTERO DE MOMPOY	1
7	5495	BASANOVA (LA)	1
8	37953	PUEBLA DE SAN MIGUE	1
9	33246	PAIPORTA	1
10	32101	NURTAL -CDA. FARANDO	1
11	29149	MINANA	1
12	7625	BUGARRA	1
13	46332	SONEJA	1
14	31982	NOVELE	1
15	11024	CARRASCA (LA)	1
16	2814	ARAYA	1
17	39063	RAIGUERO (EL)	1
18	28097	MAS DEL SECO	1
19	1651	ALGAR (EL)	1
20	49180	TURIS	1
21	45004	SAX	1
22	48192	TORRE-BELTRAN	1
23	17859	ENRAMONA	1
24	2539	ANNA	0
25	54092	VISTABELLA DEL MAES	0

Un total de **1663 files**

6.63 Traure el codi i la descripció de les categories, amb el número d'articles de cada categoria i el número total d'unitats venudes de cada categoria, d'aquelles categories de les quals tenim més de 15 articles, i ordenat per número d'articles de forma descendent. Aquesta sentència ja és prou complicada. Concretament haureu de tenir en compte que:

- Voldrem traure el número d'articles de cada categoria, però potser alguns articles no s'han venut, i per tant no apareixeran en la taula LINIA_FAC.
- I també tenim el problema que, com ens fa falta la taula LINIA_FAC, un article venut en més d'una factura apareixerà més d'una vegada. Si comptem de forma normal, el comptaríem més d'una vegada cada article. Per tant voldrem comptar els diferents articles de cada categoria.

	cod_cat character varying(15)	descripcio character varying(50)	count bigint	sum numeric
1	Legrand	Components marca Le	189	887
2	IntMagn	Interruptor Magneto	55	182
3	Niessen	Components Niesen S	43	201
4	Ticino	Components marca Ti	34	117
5	Simon	Components marca Si	27	148

6.64 Traure el número màxim de factures fetes a un client

	Número màxim de factures bigint
1	5

6.65 Traure el l'import que suposa la factura més cara i l'import que suposa la més barata (sense considerar ni descomptes ni IVA)

	Màxim iport numeric	Mínim import numeric
1	1542.00	0.51

6.66 Traure el número de factures més alt que s'ha venut per venedor en cada trimestre (no traurem qui és el venedor, que seria encara més complicat). Per a poder agrupar per trimestre, ens farà falta la funció **TO_CHAR(data,'Q')**, que trau el número de trimestre. El pas previ és calcular el número de factures de cada venedor i en cada trimestre. Després, amb la informació anterior, voldrem calcular el màxim de cada trimestre.

	trim text	max bigint
1	1	5
2	2	6
3	3	7
4	4	4

6.67 Traure els articles més cars que la mitjana. Tragueu-los ordenats per la categoria, i després per codi d'article.

	cod_a character varying(10)	descrip character varying(50)	preu numeric(6,2)	stock numeric(4,0)	stock_min numeric(4,0)	cod_cat character varying(15)
1	B10007B	Placa 2 E. Bjc	68.29	1	1	BjcOlimpia
2	F05/16	Interruptor Dif	27.65	1	1	IntDif
3	F05/17	Interruptor Dif	20.73	1	1	IntDif
4	F05/50	Interruptor Dif	25.24	1	1	IntDif
5	HICH36	Interruptor Mag	28.58	1	1	IntMagn
6	ID22530	Interruptor Mag	47.60	1	1	IntMagn
7	ID24030	Interruptor Mag	50.39	1	1	IntMagn
8	ID42530	Interruptor Mag	87.36	1	1	IntMagn
9	ID425300	Interruptor Mag	73.83	1	1	IntMagn
10	ID44030	Interruptor Mag	94.18	1	1	IntMagn
11	ID440300	Interruptor Mag	79.63	1	1	IntMagn
12	ID46330	Interruptor Mag	204.34	1	1	IntMagn
13	ID463300	Interruptor Mag	105.06	1	1	IntMagn
14	IM2F15	Interruptor Mag	16.41	1	1	IntMagn
15	IM2F25	Interruptor Mag	16.41	1	1	IntMagn
16	IM2F30	Interruptor Mag	17.20	1	1	IntMagn

Un total de **164** files

6.68 Modificar l'anterior per a traure els articles més cars que la mitjana de la seua categoria. Tragueu-los ordenats per la categoria

	cod_a character varying(10)	descrip character varying(50)	preu numeric(6,2)	stock numeric(4,0)	stock_min numeric(4,0)	cod_cat character varying(15)
1	B10007B	Placa 2 E. Bjc	68.29	1	1	BjcOlimpia
2	F05/16	Interruptor Dif	27.65	1	1	IntDif
3	F05/17	Interruptor Dif	20.73	1	1	IntDif
4	F05/50	Interruptor Dif	25.24	1	1	IntDif
5	ID22530	Interruptor Mag	47.60	1	1	IntMagn
6	ID24030	Interruptor Mag	50.39	1	1	IntMagn
7	ID42530	Interruptor Mag	87.36	1	1	IntMagn
8	ID425300	Interruptor Mag	73.83	1	1	IntMagn
9	ID44030	Interruptor Mag	94.18	1	1	IntMagn
10	ID440300	Interruptor Mag	79.63	1	1	IntMagn
11	ID46330	Interruptor Mag	204.34	1	1	IntMagn
12	ID463300	Interruptor Mag	105.06	1	1	IntMagn
13	IM3P50U	Interruptor Mag	51.39	1	1	IntMagn
14	im4P10L	Interruptor Mag	32.60	1	1	IntMagn
15	im4P10V	Interruptor Mag	39.15	1	1	IntMagn
16	im4P25L	Interruptor Mag	32.60	1	1	IntMagn

Un total de **75** files

És un resultat molt similar a l'anterior, però observeu que ara no estan els productes de les files 5, 14, 15, ...

6.69 Traure els pobles on tenim clients però no tenim venedors. Ha de ser per mig de subconsultes (en plural). Ordeneu per codi del poble.

	cod_pob numeric(5,0)	nom character varying(50)	cod_pro numeric(2,0)
1	1651	ALGAR (EL)	3
2	2050	ALTERO DE MOMPO	46
3	2814	ARAYA	12
4	5495	BASANOVA (LA)	12
5	7625	BUGARRA	46
6	7766	BURRIANA	12
7	11024	CARRASCA (LA)	46
8	17859	ENRAMONA	12
9	28097	MAS DEL SECO	12
10	29149	MIÑANA	3
11	31982	NOVELE	46
12	32093	NULES	12
13	32101	NURTAL - CDA. FAR	46
14	33246	PAIPORTA	46
15	37953	PUEBLA DE SAN M	46
16	39063	RAIGUERO (EL)	3
17	45004	SAX	3
18	46332	SONEJA	12
19	48037	TORO (EL)	12
20	48192	TORRE - BELTRAN	12
21	49180	TURIS	46
22	53596	VILLARREAL	12

6.70 Traure el nom de tots els clients i venedors implicats en alguna venda del primer trimestre de 2015. Intentar traure en una segona columna el text **Venedor** per als venedors, i **Client** per als clients. Ordenat pel nom.

	nom character varying(100)	?column? text
1	AGOST TIRADO, JORGE VICTOR	Venedor
2	AMO MONTSN, RAMON FERNANDO	Client
3	CANCELAS MORA, MARIA	Client
4	CORBATO CARUANA, JOSE JUSTO	Venedor
5	DANIEL MIRALLES, SERGIO	Venedor
6	GUILLEN VILLALONGA, NATALIA	Venedor
7	HUGUET PERIS, JUAN ANGEL	Client
8	IGLESIAS NAVARRO, IGNACIO	Client
9	LOPEZ BOTELLA, MAURO	Client
10	LOPEZ GUITART, XAVIER	Client
11	LOPEZ RINCON, LUIS MIGUEL	Client
12	MATEU MARTI, MARIA DOLORES	Client
13	NAVARRO BARBERO, MARIA LLEDO	Client
14	PEREZ CEBRIA, IGNACIO DIEGO	Venedor
15	PINEL HUERTA, VICENTE	Client
16	PITARCH MONSONIS, MARIA CARMEN	Client
17	POY OMELLA, PALOMA	Venedor
18	ROCA FAURA, ANTONIO DIEGO	Venedor
19	RUBERT CANO, DIEGO GUILLERMO	Venedor
20	SAMPEDRO SIMO, MARIA MERCEDES	Client
21	TICHELL MONLLEO, MARIA ANGELES	Client
22	TUR MARTIN, MANUEL FRANCISCO	Client
23	VIDAL DIEZ, JOSE	Venedor
24	VILLALONGA RAMIREZ, DIEGO SERGIO	Client
25	VILLALONGA SANCHIS, MILAGROS	Client

6.71a Traure per mig de sentències d'operacions de conjunts els pobles on tenim algun venedor o algun client. No volem resultats repetits, i ho volem ordenat pel nom del poble.

	nom character varying(50)
1	ALGAR (EL)
2	ALTERO DE MOMPO
3	ARAYA
4	BASANOVA (LA)
5	BENICARLO
6	BUGARRA
7	BURRIANA
8	CARRASCA (LA)
9	CASTELLON
10	ENRAMONA
11	LUCENA DEL CID

Un total de **31** files

6.71b Modificar l'anterior per a traure els pobles on tenim al mateix temps venedors i clients

	nom character varying(50)
1	CASTELLON

6.71c Modificar l'anterior per a traure els pobles on tenim venedors però no tenim clients

	nom character varying(50)
1	BENICARLO
2	LUCENA DEL CID
3	PILAR DE LA HORADADA
4	SAN JUAN DE ALICANTE
5	USERAS
6	VALENCIA
7	VISTABELLA
8	VIVER

