**Sistemes Gestors de Bases de Dades**

**Tema 6: SQL - Consulta bàsica (part I-2)**

**12. Funcions d'agregat**

Les **funcions d'agrega**t, o funcions de domini agregat, són aquelles que trauen un resultat a partir dels valors d'un determinat camp en un conjunt de files. Així tindrem una funció per a **sumar** els valors d'una columna, o **comptar**-los, o traure la **mitjana**, o el **màxim**, ...

Actuaran sobre un conjunt de files determinat, que en principi suposarem que és tota la taula (totes les files de la taula). En la següent pregunta, veurem que el conjunt de files sobre el qual es calcula una funció d'agregat, el podrem canviar amb la clàusula **GROUP BY**.

Sintaxi

**COUNT (\* | <expressió> )** : compta el número de files; si es posa una columna o expressió, no es comptaran els valors nuls.

**SUM ( <expressió> )** : torna la suma de la columna o expressió especificada. Ignora els valors nuls.

**AVG ( <expressió> )** : calcula la mitjana aritmètica de la columna o expressió especificada. Ignora els valors nuls.

**VAR\_SAMP ( <expressió> )** : calcula la variància d'una mostra a partir de la columna o expressió especificada.

**STDDEV ( <expressió> )** : desviació típica d'una mostra.

**MAX ( <expressió> )** : calcula el màxim.

**MIN ( <expressió> )** : calcula el mínim.

Per exemple, si volem saber el nombre d'Instituts:

**SELECT COUNT(\*) AS "Nombre d'Instituts"**

**FROM INSTITUTS;**

Nota

És interessant la utilització d'àlias, per a que no apareguen capçaleres com ***count***

Exemples

1. Comptar el nombre total de pobles.

**SELECT Count(\*)**

**FROM POBLACIONS;**

2. Comptar el nombre de poblacions de la **Plana Alta**.

**SELECT Count(\*)**

**FROM POBLACIONS WHERE nom\_c = 'Plana Alta';**

3. Calcular la mitjana d'habitants dels pobles de la **Plana Alta** i **Plana Baixa**.

**SELECT AVG(poblacio)**

**FROM POBLACIONS WHERE nom\_c = 'Plana Alta' OR nom\_c = 'Plana Baixa'**

4. Calcular la mitjana de densitat de les poblacions. La densitat es calcula com el número d'habitants dividit per l'extensió.

**SELECT AVG(poblacio/extensio)**

**FROM POBLACIONS;**

5. Calcular l'altura màxima i mínima de tots els pobles.

**SELECT MAX(altura), MIN(altura)**

**FROM POBLACIONS**

**Exercicis apartat 12**

**6.15** Comptar el nombre de **clients** que tenen el **codi postal nul**.

**6.16** Comptar el número de vegades que l'article **L76104** entra en les línies de factura, i el número total d'unitats venudes d'aquest article. Només us fa falta la taula LINIA\_FAC.

**6.17a** Traure la **mitjana** del **stock** dels articles.

**6.17b** Modificar l'anterior per a tenir en compte els valors nuls, com si foren 0. Us vindrà bé la funció COALESCE que converteix els nuls del primer paràmetre al valor donat com a segon paràmetre (si és diferent de nul, deixa igual el valor). Per tant l'heu d'utilitzar d'aquesta manera: COALESCE(stock,0)

**6.18** Comptar **quantes factures** té el client **375**

**6.19** Calcular el **descompte màxim**, el **mínim** i el descompte **mitjà** de les **factures**.

**Exercicis**

**13. La clàusula GROUP BY**

Agrupa totes les files amb valors iguals d'una o d'unes columnes

Sintaxi

**SELECT <columnes> FROM <taules> GROUP BY <columnes>**

Per cada fila amb valors iguals de les columnes de la clàusula **GROUP BY**, en trau només una, és a dir, les agrupa.

Les ***funcions d'agregat*** agafen tot el seu sentit i potència combinades amb el **GROUP BY**: tornaran un valor per cada grup. Així, per exemple, aquesta sentència traurà quantes poblacions hi ha en cada comarca:

**SELECT COUNT(\*)**

**FROM POBLACIONS GROUP BY nom\_c**

Si volem excloure files per a que no entren en les agrupacions, ho farem per mig de la clàusula **WHERE**. En aquest exemple ens aprofitem del codi de municipi, que en el cas dels municipis de la província de Castelló és 12000 i pico.

**SELECT COUNT(\*)**

**FROM POBLACIONS WHERE cod\_m >= 12000 and cod\_m < 13000 GROUP BY nom\_c**

Quan tenim agrupacions de files, bé perquè utilitzem la clàusula GROUP BY, bé perquè entre les columnes que es trien en el SELECT hi ha alguna funció d'agregat, o les dues coses a l'hora, es poden cometre errors amb una relativa facilitat. Quan hi ha una agrupació **totes les columnes que seleccionem amb el SELECT hauran d'estar en el GROUP BY, o bé estar dins d'una funció d'agregat**. En cas contrari ens donarà error. Per exemple, aquesta consulta funciona bé, i de fet és millor consulta que les d'abans, ja que ens diu la comarca i el numero de pobles que té cadascuna:

**SELECT nom\_c, COUNT(\*)**

**FROM POBLACIONS GROUP BY nom\_c**

Però aquesta no:

**SELECT nom\_c, COUNT(\*), cod\_m**

**FROM POBLACIONS GROUP BY nom\_c**

És sintàcticament incorrecta, i a banda no té sentit: si agrupem tots els pobles de la mateixa comarca ¿com hem de poder traure després el codi del municipi de cada poble?.

Com és relativament fàcil cometre aquest error, haurem d'identificar aquest error, per a poder solucionar-lo.

Per exemple, aquesta consulta ens dóna la població més alta de cada comarca:

**SELECT nom\_c, MAX(altura)**

**FROM POBLACIONS GROUP BY nom\_c**

Però si intentàrem traure també el nom de la població més alta de cada comarca:

**SELECT nom\_c, MAX(altura), nom**

**FROM POBLACIONS GROUP BY nom\_c**

ens donaria el següent error:

Hem d'aprendre a identificar aquest error, i solucionar-lo. En aquest cas el solucionarem llevant el camp **Nom**. La sentència per a poder traure el nom de la població més alta de cada comarca és complicada, i encara no podem fer-la.

Exemples

1. Comptar el nombre d'instituts de cada població.

**SELECT cod\_m, COUNT(\*) FROM INSTITUTS GROUP BY cod\_m**

2. Comptar el nombre de comarques de cada província.

**SELECT provincia, COUNT(\*)**

**FROM COMARQUES GROUP BY provincia**

3. Calcular l'altura màxima, mínima i l'altura mitjana de cada comarca.

**SELECT nom\_c, MAX(altura), MIN(altura), AVG(altura)**

**FROM POBLACIONS GROUP BY nom\_c**

4. Comptar el nombre d'instituts de cada població i cada codi postal.

**SELECT cod\_m, codpostal, COUNT(\*)**

**FROM INSTITUTS GROUP BY cod\_m, codpostal**

**Exercicis apartat 13**

**6.20** Comptar el número de pobles de cada província (és suficient traure el codi de la província i el número de pobles).

**6.21** Comptar el número de factures de cada venedor a cada client.

**6.22** Comptar el número de factures de cada trimestre. Per a poder traure el trimestre i agrupar per ell (ens val el número de trimestre, que va del 1 al 4), podem utilitzar la funció **TO\_CHAR(data,'Q')**.

**6.23** Calcular el total de cada factura, sense aplicar descomptes ni IVA. Només ens farà falta la taula **LINIES\_FAC**, i consistirà en agrupar per cada **num\_f** per a calcular la suma del **preu** multiplicat per la **quantitat**.

**Exercicis**

**14. La clàusula HAVING**

Aquesta clàusula acompanya normalment a la de **GROUP BY**, i servirà pera poder triar alguns grups que acomplesquen una determinada condició.

Pot anar sense el GROUP BY, però aleshores el seu funcionament és com el WHERE, i per tant no val la pena. És a dir, en la pràctica sempre que trobem un HAVING hi haurà també el GROUP BY i servirà per seleccionar alguns grups, els que acomplesquen la condició del HAVING.

També podríem dir que el HAVING és al GROUP BY, el que el WHERE és al SELECT

Sintaxi

**SELECT <columnes> FROM <taules> [GROUP BY <columnes>] HAVING <condició>**

Únicament comentarem el cas en què acompanya al GROUP BY. I com hem dit, el que fa és filtrar els grups: dels grups resultants del GROUP BY, només eixiran els que acomplesquen la condició.

Aquesta condició contindrà alguna funció d'agregat o contindrà columnes incloses en el GROUP BY. Fixeu-vos que és lògic, ja que serveix per a triar grups una vegada fets, i aleshores ja no es podrà anar a un element del grup.

Per exemple, aquesta sentència servirà per traure les comarques on hi ha més de 20 pobles, i el número que hi ha:

**SELECT nom\_c, COUNT(\*)**

**FROM POBLACIONS GROUP BY nom\_c HAVING COUNT(\*) > 20;**

Exemples

1. Traure aquelles poblacions que tenen més d'un de Centres Integrats de Formació Professional. La manera de saber que és un Centre Integrat és perquè el seu nom comença per CIPFP. De moment només traurem el codi de la població, i així només ens fa falta la taula INSTITUTS. Més avant aprendrem a agafar les dades de més d'una taula, i aleshores traurem també el nom de la població

**SELECT cod\_m, COUNT(\*) FROM INSTITUTS WHERE nom LIKE 'CIPFP%' GROUP BY cod\_m HAVING COUNT(\*) > 1**

El que fem en aquesta sentència és, de la taula INSTITUTS seleccionar únicament els Centres Integrats (utilitzant l'operador LIKE per a que comencen per CIPFP)i i després agrupar pel codi de municipi. Una vegada fets els grups, eliminarem els grups que no acompleixen la condició de HAVING, és a dir, els que tenen 0 o 1 Centre Integrat. I d'aquestos grups seleccionats traurem el codi del municipi i el número de Centres Integrats (que sempre serà igual o major que 2).

2. Calcular el número d'habitants màxim, el mínim i el número d'habitants mitjà de les poblacions de les comarques amb més de 20 pobles.

**SELECT nom\_c , COUNT(cod\_m) AS "Número de pobles" , Max(poblacio) AS Màxim , Min(poblacio) AS Mínim**

**, Avg(poblacio) AS Mitjana FROM POBLACIONS GROUP BY nom\_c HAVING COUNT(cod\_m) > 20;**

3. Traure l'altura mitjana, total de població i població mitjana, d'aquelles comarques que tenen una altura mitjana superior a 800 metres.

**SELECT nom\_c , AVG(altura) AS "Altura mitjana" , SUM(poblacio) AS "Total població" , Avg(poblacio)**

**AS "Població mitjana" FROM POBLACIONS GROUP BY nom\_c HAVING AVG(altura) > 800;**

**Exercicis apartat 14**

**6.24** Calcular la mitjana de quantitats demanades d'aquells articles que s'han demanat més de dues vegades. Observeu que la taula que ens fa falta és LINIA\_FAC, i que la condició (en el HAVING) és sobre el número de vegades que entra l'article en una linia de factura, però el resultat que s'ha de mostrar és la mitjana de la quantitat.

**6.25** Traure els pobles que tenen entre 3 i 7 clients. Traure només el codi del poble i aquest número

**6.26** Traure les categories que tenen més d'un article "car" (de més de 100 €). Observeu que també ens eixirà la categoria NULL, és a dir, apareixerà com una categoria aquells articles que no estan catalogats.

**Exercicis**

**15. La clàusula ORDER BY**

Ordena les files del resultat respecte l'ordre especificat

Sintaxi

**SELECT <columnes> FROM <taules> ORDER BY camp1 [ASC | DESC] { , camp2 [ASC | DESC] }**

Ací tenim un exemple:

**SELECT nom\_c, provincia**

**FROM COMARQUES ORDER BY nom\_c**

S'ordenaran les files pel camp, en l'ordre marcat: ascendent o descendent (per defecte, ascendent). Si hi haguera un segon camp d'ordenació, aquest entraria en joc en cas de valors iguals del primer. Aquest segon podrà ser en ordre ascendent o descendent, independentment de l'ordre del primer camp.

**SELECT nom\_c, provincia**

**FROM COMARQUES ORDER BY provincia DESC, nom\_c**

Es podrà ordenar per qualsevol camp de la taula, estiga indexada per aquest camp o no. L'avantatge d'estar indexada respecte a un camp és la rapidesa en l'ordenació. Així, si tenim una taula gran i ordenem per un determinat camp, es perdrà temps en aquesta ordenació. Si contínuament estem ordenant per aquest camp, perdrem aquest temps moltes vegades. Aleshores seria convenient crear un índex. Però hem de recordar que la creació d'un índex ocupa espai. Per tant no és bona solució indexar per tots els camps. Únicament, en tot cas, per aquells que més s'ordene. Veurem la creació d'índex en la tercera part d'aquest tema.

I es pot especificar en l'ordenació, una expressió que agafe un o més d'un camp amb operadors i funcions. Es pot posar un camp o una expressió que no estiga en la llista de camps o expressions que es volen visualitzar (al costat del SELECT), encara que normalment sí que ho visualitzarem, per a poder comprovar que realment està ordenat pel que s'ha especificat. Per exemple, si volem ordenar per la densitat d'habitants de les poblacions, que es calcula dividint el número d'habitants per l'extensió:

**SELECT nom, poblacio, extensio**

**FROM POBLACIONS ORDER BY poblacio/extensio DESC**

Observeu que estem ordenant per un camp (**poblacio / extensio**) que no estem visualitzant, encara que seria molt més il·lustratiu mostrar-lo

Opcionalment, en el moment d'especificar el camp o l'expressió per la qual volem ordenar, si aquesta apareix en la llista de columnes a visualitzar, podrem posar senzillament el número d'ordre de la columna. Així, per exemple, podem fer el següent:

**SELECT nom, poblacio, extensio, poblacio/extensio**

**FROM POBLACIONS ORDER BY 4 DESC**

On estem indicant que s'ordene de forma descendent per la quarta columna que va a visualitzar-se, és a dir, per **poblacio / extensio**

Hem d'observar que la clàusula ORDER BY és l'última, i que en cas d'haver clàusula GROUP BY, s'intentarà ordenar després d'haver agrupat. Per tant en cas de que la sentència continga un GROUP BY o s'haja utilitzat alguna funció d'agregat (que implica fer grups), només podrem posar en el ORDER BY camps que estiguen en el GROUP BY o que formen part d'una funció d'agregat. El raonament és el mateix que el fet en la clàusula GROUP BY o HAVING i l'error en cas de no respectar açò seria el matiex que el vist en aquell apartat.

Exemples

1. Traure tota la informació de les poblacions ordenades pel nom de la població.

**SELECT \***

**FROM POBLACIONS ORDER BY nom**

2. Traure tota la informació de les poblacions, ordenades pel nom de la comarca, i dins d'aquesta per l'altura (de forma descendent).

**SELECT \***

**FROM POBLACIONS ORDER BY nom\_c, altura DESC**

3. Traure les comarques amb el número de pobles i total d'habitants, d'aquelles que tenen més de 10 pobles, ordenades pel número de

pobles, i dins d'aquest pel total de població de forma descendent.

**SELECT nom\_c, COUNT(\*), SUM(poblacio)**

**FROM POBLACIONS GROUP BY nom\_c HAVING COUNT(\*) > 10 order by COUNT(\*) , SUM(poblacio) DESC**

**SELECT nom\_c, COUNT(\*), SUM(poblacio)**

**FROM POBLACIONS GROUP BY nom\_c HAVING COUNT(\*) > 10 order by 2 , 3 DESC**

**Exercicis apartat 15**

**6.27** Traure tots els clients ordenats per codi de població, i dins d'aquestos per codi postal.

**6.28** Traure tots els articles ordenats per la categoria, dins d'aquest pel stock, i dins d'aquest per preu (de forma descendent)

**6.29** Traure els codis de venedor amb el número de factures venudes en el segon semestre de 2015, ordenades per aquest número de forma descendent

**Exercicis**

**16. El predicat DISTINCT**

Per defecte, si no especifiquem el contrari, eixiran *totes* les files de la taula o taules que acomplesquen les condicions. Així, per exemple, si de la taula d'instituts volguérem treure únicament el codi de la població, ens eixiria per exemple 12040 (el codi de Castelló) en tantes files com instituts de Castelló tinguem (concretament 14). És el mateix resultat que si haguérem posat el predicat **ALL** davant de les columnes, ja que aquest és el predicat per defecte:

**SELECT ALL cod\_m**

**FROM INSTITUTS**

Aquest resultat no és el correcte, si volem consultes com "Poblacions on hi ha instituts". Seria millor si isquera 12040 només una vegada. Açò ho aconseguirem amb el predicat DISTINCT.

En definitiva el que farà el predicat DISTINCT serà traure les files diferents del resultat que demanem. Si només demanem un camp, traurà els valors diferents d'aquest camp. Si demanem més d'un camp, traurà els valors diferents per al conjunt dels camps (és a dir les files diferents, que en un camp poden coincidir, però no en el conjunt de tots els camps)

Sintaxi

**SELECT DISTINCT <columnes >**

**FROM <taules>**

Així, en l'exemple anterior:

**SELECT DISTINCT cod\_m**

**FROM INSTITUTS**

Hi ha una variant del DISTINCT que suparta PostgreSQL, però que no suporten altres SGBD més senzills, com Access. És posar el DISTINCT dins d'una funció d'agregat, com per exemple COUNT. El resultat és que comptarà (o la funció implicada: sumarà calcularà la mitjana, ...) els valors diferents del camp que vaja a continuació.

Així per exemple, si volem comptar en quantes poblacions diferents tenim instituts, la consulta és tan senzilla com aquesta:

**SELECT COUNT(DISTINCT cod\_m)**

**FROM INSTITUTS**

Exemples

1. Traure les diferents provincies.

**SELECT DISTINCT provincia**

**FROM COMARQUES**

2. Traure els distints districtes (codis postals) de Castelló (codi de municipi 12040) on hi ha instituts

**SELECT DISTINCT codpostal**

**FROM INSTITUTS WHERE cod\_m=12040**

3. Traure les distintes comarques i llengües que es parlen en elles

**SELECT DISTINCT nom\_c , llengua**

**FROM POBLACIONS ORDER BY 1**

**Exercicis apartat 16**

**6.30** Traure els venedors que han venut alguna cosa el mes de gener de 2015.

**6.31** Traure els diferents caps de venedors (eviteu que aparega el valor nul)

**6.32** Comptar en quantes poblacions tenim clients

**Exercicis**

**17. La clàusula LIMIT .. OFFSET**

Per mig de la clàusula **LIMIT - OFFSET** podrem fer que no apareguen totes les files que torna la sentència, sinó unes quantes.

**LIMIT número**: especificarem quantes files volem que es tornen **OFFSET número**: especificarem a partir de quina posició volem que es tornen les files

Si no posem la part del OFFSET, apareixeran les primeres, i si especifiquem OFFSET, se saltaran les primeres, tantes com s'indica en OFFSET. Per a que realment tinga sentit aquesta clàusula, és conveninet ordenar la informació, ja que al dir les primeres ja s'està assumint que seran les primeres respecte algun ordre. Així, per exemple ens podrem plantejar traure coses com les 10 poblacions més altes, o les més habitades.

L'ordre implícit que acabem de comentar s'haurà de fer per mig de la clàusula ORDER BY. Així, si volem traure els clients més joves haurem d'ordenar per la data de naixement en ordre descendent, per a després traure els primenrs. Per tant és molt difícil veure una clàusula LIMIT si no tenim una clàusula ORDER BY.

**Nota**

En el SQL d'Access no existeix la clàusula LIMIT. Per a fer coses similar disposa del predicat **TOP**, que es posa immediatament després del SELECT, però sempre traurà les primeres, no té possibilitat d'OFFSET.

Sintaxi

**SELECT <columnes> FROM <taules> [LIMIT *n*] [OFFSET *m*]**

El número ***n*** ha de ser un enter, i se seleccionaran únicament les ***n*** primeres files (les de dalt). En cas de posat OFFSET se saltaran ***m*** files. En cas de no posar **LIMIT**, se saltaran ***m*** files i es trauran totes les altres fins el final.

Per exemple, si volem traure les 10 poblacions més altes, haurem d'agafar les 10 primeres, ordenant per altura en forma descendent:

**SELECT nom , altura FROM POBLACIONS ORDER BY altura DESC LIMIT 10**

L'exemple anterior sembla correcte, però no funciona del tot bé perquè amb les dades que tenim, hi ha 3 poblacions que no tenen altura, i el valor nul el posa al final de tots els altres valors, en ordre ascendent, i per tant al principi en ordre descendent.

Ho podem arreglar senzillament saltant les 3 primeres (que són les del nul)

**SELECT nom , altura FROM POBLACIONS ORDER BY altura DESC LIMIT 10 OFFSET 3**

Encara que sembla millor llevar les d'altura nula, així no estem obligats a saber quantes poblacions amb altura nula hi ha

**SELECT nom , altura FROM POBLACIONS WHERE altura IS NOT NULL ORDER BY altura DESC LIMIT 10**

Si vulguérem traure totes les poblacions i altures, excepte les que tenen nul, i sabem que aquestes en són 3, podem posar OFFSET sense LIMIT, per a saltar les 3 primeres, i traure-les totes fins el final

**SELECT nom , altura FROM POBLACIONS**

**ORDER BY altura DESC OFFSET 3**

Exemples

1. Traure les 5 poblacions més poblades

**SELECT nom , poblacio**

**FROM POBLACIONS ORDER BY poblacio DESC LIMIT 5**

2. Traure les 4 comarques amb més pobles.

**SELECT nom\_c , COUNT(\*) FROM POBLACIONS GROUP BY nom\_c ORDER BY 2 DESC LIMIT 4**

3. Traure les 10 poblacions amb més instituts, saltant-nos les 3 primeres.

**SELECT cod\_m , COUNT(\*)**

**FROM INSTITUTS GROUP BY cod\_m ORDER BY 2 DESC LIMIT 10 OFFSET 3**

**Exercicis apartat 17**

**6.33** Traure tota la informació dels dos articles més cars.

**6.34** Traure el codi de les tres ciutats amb més clients

**6.35** Traure el venedor que ha venut menys factures

**Exercicis**

**18. Consulta de creació de taules**

A banda de poder consultar informació de una o més d'una taula, la sentència SELECT pot servir per a crear una nova taula, amb estructura i dades (les que venen de la pròpia sentència SELECT). Això sí, no podrem definir d'aquesta manera ni clau principal, ni claus externes, ni cap altra restricció de les conegudes.

A més, aquesta característica escapa del estàndard ANSI SQL, per la qual cosa no li donarem excessiva importància.

Sintaxi

**SELECT <columnes> INTO nova\_taula**

**FROM <taules>**

La sentència pot dur qualsevol clàusula o predicat dels vistos fins ara, i el resultat que done aquesta sentència, es guardarà en una nova taula, amb el nom especificat.

El nom dels camps de la nova taula seran els especificats en l'apartat <columnes>. Per tant és especialment recomanable la utilització d'àlies, ja que si en posem seran el noms dels camps de la nova taula.

Els tipus de dades dels camps seran els heretats de la consulta SELECT.

En cas d'existir ja una taula amb el nom especificat ens avisarà d'aquest fet, donant-nos la possibilitat d'esborrar la taula anterior i crear la nova o cancelar.

**Nota**

És molt recomanable, com d'altres sentències de manipulació de dades que veurem més endavant, executar primer la sentència sense el **INTO**, per a no crear la taula encara. Quan estiguem segurs que el resultat és el que desitgem, afegim el INTO, i la taula es crearà a més garanties.

Exemples

1. Crear una còpia de la taula comarques anomenada **COMARQUES\_COPIA**.

**SELECT \* INTO COMARQUES\_COPIA**

**FROM COMARQUES**

Per a no "embrutar" la Base de Dades, podem esborrar-la després d'haver vist la seua creació amb la sentència

**DROP TABLE COMARQUES\_COPIA**

2. Crear una taula anomenada **RESUM\_COMARQUES** que continga el nom de la comarca, el número de pobles, el total de població i

l'altura mitjana

**SELECT nom\_c, COUNT(\*) AS num\_pobles, SUM(poblacio) AS poblacio , SUM(extensio) AS extensio ,**

**AVG(altura) AS altura\_mitjana INTO RESUM\_COMARQUES FROM POBLACIONS GROUP BY nom\_c**

Igual que en l'anterior, després d'haver vist la seua creació i contingut, podem esborrar-la amb la sentència

**DROP TABLE RESUM\_COMARQUES**

**Exercicis apartat 18**

**6.36** Crear una taula anomenada **ARTICLE\_36**, que siga una còpia de la taula ARTICLE, però substituint els valors nuls de **stock** i **stock\_min** per zeros.

**6.37** Utilitzar la taula anterior per a traure el stock màxim, el mínim i la mitjana de stocks. Observeu que si utilitzàrem la taula ARTICLE, els resultats no serien els mateixos (excepte el màxim), sobretot la mitjana, ja que els valors nuls no entrarien en els càlculs d'aquesta mitjana.

**Exercicis**

**19. Ordre amb què s'executa una sentència SQL**

Com hem vist, i com veurem en la Part II d'aquest tema, la sentència **SELECT** és molt completa i molt potent. Pot fer moltes coses.

Potser ens convinga saber en quin ordre s'executen les clàusules de què es compon, perquè això ens pot prevenir de possibles errors en el moment de construir una sentència d'una certa envergadura. L'ordre d'execució és el següent:

Primer s'agafen les dades des de la taula o les taules especificades en el **FROM**. No podrem tractar informació que no tinguem en aquest origen de dades.

Després s'eliminen les files que no acompleixen la condició del **WHERE**, en cas que tinguem aquesta clàusula.

Després les files resultants s'agrupen pel o pels camps especificats en el **GROUP BY**, en cas que tinguem aquesta clàusula.

Una vegada fets els grups, s'eliminen els que no acomplesquen la condició del **HAVING**, en cas que tinguem aquesta clàusula.

Després se selecciona la informació especificada en les columnes, que en cas d'haver alguna funció d'agregat actuarà sobre els grups que resten (si teníem clàusula GROUP BY) o sobre el total de l'origen de dades.

Posteriorment s'ordena pels camps especificats en el **ORDER BY**, en cas que tinguem aquesta clàusula.

Després s'aplica els predicat **DISTINCT** en cas de tenir-lo especificat.

Per últim s'agafen tantes files com indica la clàusula **LIMIT**, desplaçades tantes com indique **OFFSET**, si és que tenim aquesta clàusula especificada..

Si tenim clàusula **INTO** es procedirà a crear una taula nova amb el resultat anterior (la vorem més endavant)

Tenir clar aquest ordre ens pot clarificar alguna cosa, i poder evitar alguns errors. L'error de la següent sentència ja s'havia explicat en la pregunta 13.

**SELECT nom\_c, COUNT(\*), cod\_m**

**FROM POBLACIONS GROUP BY nom\_c**

ens donarà el següent error:

Però si analitzem l'ordre en què s'executen és lògic: quan arribem a mostrar els camps (entre ells **cod\_m**) els grups ja s'han fet, i per a valors iguals de **nom\_c**. En aquest moment no puc traure una cosa individual de cada grup com és el codi de municipi, perquè ja s'ha agrupat. En aquest moment només es pot intentar traure el nom de la comarca (ja que té el mateix valor per a tot el grup, és el camp pel qual hem agrupat), o alguna funció d'agregat, que calcula sobre el grup. I d'això ens intenta avisar PostgreSQL.

Per a solucionar-lo podem incloure el **cod\_m** en el **GROUP BY**, i aleshores farem un grup per cada comarca i població diferent, però segurament això no ens valdrà de res en aquest exemple, perquè cada grup només contindrà un element (un municipi), encara que en altres exemples sí que pot tenir sentit. O si no era això el que preteníem, senzillament llevem el camp cod\_m de la sentència, i ens funcionarà bé.

Un altre exemple il·lustratiu (que ja el vam posar molt paregut en la pregunta 13) pot ser el següent: podríem intentar traure l'altura màxima de tots els pobles, i la població que té aquesta altura. Podríem estar temptats de fer-lo d'aquesta manera:

**SELECT MAX(altura), nom**

**FROM POBLACIONS;**

Ens donarà el mateix error que abans, ja que com tenim una funció d'agregat intentarà fer grups, i com no tenim clàusula GROUP BY tota la taula serà un grup. I podrà calcular el màxim sense problemes, però no podrà traure una cosa individual del grup, com és el nom. I en aquest cas no es pot solucionar incloent el nom en el GROUP BY, perquè aleshores farem un grup per cada població. De moment, abans de veure les subconsultes, només podem resoldre aquest exemple ordenant per l'altura de forma descendent, i fer **LIMIT 1**.

Podem observar que si tenim clàusula **GROUP BY**, a partir d'aquest moment tots els camps que posem han d'estar en el GROUP BY o en una funció d'agregat, tant en la condició del **HAVING**, com en les **columnes** com en el **ORDER BY**. En canvi no caldrà per a la clàusula **WHERE**, ja que aquesta es realitza abans que el **GROUP BY**

**Exercicis de tot el tema, amb els resultats**

En la BD **factura**, connectant com a usuari **factura**:

**6.1** Traure tota la informació dels pobles (anomeneu-la **Ex\_6\_1.sql**).

Un total de **1663 files**

**6.2** Traure el codi postal, el nom i l'adreça, per aquest ordre, de tots els venedors (anomeneu-la **Ex\_6\_2.sql**).

**6.3** Traure el codi d'article, la descripció, preu i preu incrementat en un 5%, de tots els articles.

Un total de **812 files**

**6.4** Traure la informació dels clients amb el següent format (ha d'anar tot en una columna):

**Damborenea Corbato, Alicia. CALLE MADRID, 83 (12425)**

Fixeu-vos que està tot en una columna, i per tant haureu de concatenar de la forma adequada. Fixeu-vos també que en en el nom només les inicials estan en majúscules

Un total de **49 files**

**6.5** Traure el num\_f, data i cod\_ven de les factures amb les següents capçaleres respectivament: **Número Factura**, **data** i **Codi Venedor** (anomeneu-lo **Ex\_6\_5**)

Un total de **105 files**

**6.6** Donar àlias als camps que ho necessiten de la taula ARTICLE (anomeneu-lo **Ex\_6\_6**)

Un total de **812 files**

**6.7** Traure els **clients** de la **ciutat** amb codi **12309**.

**6.8** Traure totes les **factures** del mes de **març** de **2015**.

**6.9** Traure tots els articles de la **categoria BjcOlimpia** amb un **stock** entre **2** i **7** unitats.

**6.10** Traure tots els **clients** que **no** tenen introduït el **codi postal**.

**6.11** Traure tots els **articles** amb el **stock** introduït però que **no** tenen introduït el **stock mínim**.

**6.12** Traure tots els **clients**, el **primer cognom** dels quals és **VILLALONGA**.

**6.13.a** Modificar l'anterior per a traure tots els que són **VILLALONGA** de **primer** o de **segon** cognom.

**6.13.b** Modificar l'anterior per a traure tots els que **no** són **VILLALONGA** ni de primer ni de segon cognom.

Un total de **46 files**

**6.14** Traure els **articles** "**Pulsador**" (la descripció conté aquesta paraula), el **preu** dels quals oscila entre **2 i 4 €** i dels quals tenim un **stock** estrictament **major** que el **stock mínim**.

**6.15** Comptar el nombre de **clients** que tenen el **codi postal nul**.

**6.16** Comptar el número de vegades que l'article **L76104** entra en les línies de factura, i el número total d'unitats venudes d'aquest article. Només us fa falta la taula LINIA\_FAC.

**6.17a** Traure la **mitjana** del **stock** dels articles.

**6.17b** Modificar l'anterior per a tenir en compte els valors nuls, com si foren 0. Us vindrà bé la funció COALESCE que converteix els nuls del primer paràmetre al valor donat com a segon paràmetre (si és diferent de nul, deixa igual el valor). Per tant l'heu d'utilitzar d'aquesta manera: COALESCE(stock,0)

**6.18** Comptar **quantes factures** té el client **375**

**6.19** Calcular el **descompte màxim**, el **mínim** i el descompte **mitjà** de les **factures**.

**6.20** Comptar el número de pobles de cada província (és suficient traure el codi de la província i el número de pobles).

**6.21** Comptar el número de factures de cada venedor a cada client.

Un total de **96 files**

D'aquestes 96 files, relativament poque tenen un valor diferent de 1 en el número de factures: la fila 29 (455, 30, 2) o la fila 34 (5, 342, 3)

**6.22** Comptar el número de factures de cada trimestre. Per a poder traure el trimestre i agrupar per ell (ens val el número de trimestre, que va del 1 al 4), podem utilitzar la funció **TO\_CHAR(data,'Q')**.

No apareix ordenat, i vol dir que en el trimestre 2 hi ha 25 factures, en el trimestre 4 hi ha 35, en el trimestre 3 hi ha 33 i en el trimestre 1 hi ha 22

**6.23** Calcular el total de cada factura, sense aplicar descomptes ni IVA. Només ens farà falta la taula **LINIES\_FAC**, i consistirà en agrupar per cada **num\_f** per a calcular la suma del **preu** multiplicat per la **quantitat**.

Un total de **105 files**

**6.24** Calcular la mitjana de quantitats demanades d'aquells articles que s'han demanat més de dues vegades. Observeu que la taula que ens fa falta és LINIA\_FAC, i que la condició (en el HAVING) és sobre el número de vegades que entra l'article en una linia de factura, però el resultat que s'ha de mostrar és la mitjana de la quantitat.

**6.25** Traure els pobles que tenen entre 3 i 7 clients. Traure només el codi del poble i aquest número

**6.26** Traure les categories que tenen més d'un article "car" (de més de 100 €). Observeu que també ens eixirà la categoria NULL, és a dir, apareixerà com una categoria aquells articles que no estan catalogats.

**6.27** Traure tots els clients ordenats per codi de població, i dins d'aquestos per codi postal.

Un total de **49 files**

**6.28** Traure tots els articles ordenats per la categoria, dins d'aquest pel stock, i dins d'aquest per preu (de forma descendent)

Un total de **812 files**

**6.29**Traure els codis de venedor amb el número de factures venudes en el segon semestre de 2015, ordenades per aquest número de forma descendent

**6.30** Traure els venedors que han venut alguna cosa el mes de gener de 2015.

**6.31** Traure els diferents caps de venedors (eviteu que aparega el valor nul)

**6.32** Comptar en quantes poblacions tenim clients

**6.33** Traure tota la informació dels dos articles més cars.

**6.34** Traure el codi de les tres ciutats amb més clients

**6.35** Traure el venedor que ha venut menys factures

**6.36** Crear una taula anomenada **ARTICLE\_36**, que siga una còpia de la taula ARTICLE, però substituint els valors nuls de **stock** i **stock\_min** per zeros.

El resultat ha de ser la creació de la taula. Si consulteu el seu contingut ha de ser el següent:

Un total de **812 files**

**6.37** Utilitzar la taula anterior per a traure el stock màxim, el mínim i la mitjana de stocks. Observeu que si utilitzàrem la taula ARTICLE, els resultats no serien els mateixos (excepte el màxim), sobretot la mitjana, ja que els valors nuls no entrarien en els càlculs d'aquesta mitjana.

Llicenciat sota la Llicència Creative Commons Reconeixement NoComercial CompartirIgual 3.0