SENTENCIES SELECT AVANÇADES

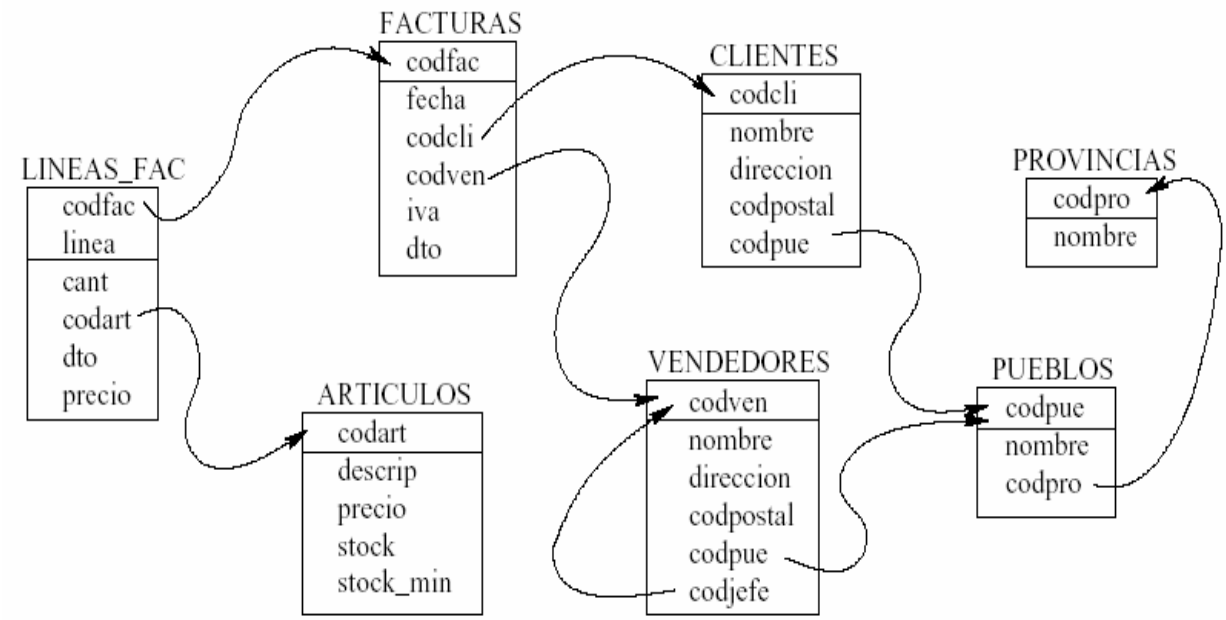
SQL PLUS d'ORACLE

SQL Postgres

A. Moll

Sobre la Base de Dades de FACTURES

La siguiente figura muestra el esquema de la base de datos gráficamente.



El preu de la taula ARTÍCULOS és el preu actual mentre que el preu de LINEAS_FAC és el preu en el que es va fer la venda en el seu moment.

EXERCICIS

1. Código, fecha e iva de las facturas sin iva (iva nulo o cero).

SQL*PLUS	Estàndart	Postgres
select codfac,fecha,iva from facturas where nvl(iva,0)=0;	select codfac,fecha,iva from facturas where iva is null or iva =0;	**select codfac,fecha,iva from facturas where coalesce(iva,0)=0;**

2. Nombre de las provincias cuya segunda letra es una “O” (bien mayúscula o minúscula).

3. Mostrar los nombres de las provincias cuya segunda letra sea una 'o'. Estas provincias deben aparecer ordenadas alfabéticamente, con la inicial en mayúscula y el resto en minúsculas.

SQL*PLUS	PostgreSQL
Initcap també hi és en ORACLE	**SELECT INITCAP(nombre) FROM provincias WHERE LOWER(SUBSTRING(nombre FROM 2 FOR 1)) = 'o' ORDER BY INITCAP(nombre);**

4. Nombre del mes actual.

SQL*PLUS	PostgreSQL
Select to_char(sysdate,'month') from dual;	select to_char(now(),'month') ;

5. Código y fecha de las facturas del año pasado para aquellos clientes cuyo código se encuentra entre 50 y 60.

SQL*PLUS
```
select codfac, fecha
from   facturas
where  codcli between 50 and 60
and    to_number( to_char( fecha, ‘yyyy’ ) ) =
       to_number( to_char( sysdate, ‘yyyy’ ) ) – 1;
```

PostgreSQL
```
select codfac, fecha
from   facturas
where  codcli between 50 and 60
and
to_number(to_char(fecha,'yyyy'),'9999')=to_number(to_char(current_date,'yyyy'),'9999')-1;
```

6. Nombre de las provincias que terminan con la letra “s” (bien mayúscula o minúscula).

7. Obtener el código y descripción de los artículos que no tienen ninguna letra en su código.

8. Descuento medio aplicado en las facturas sin considerar los valores nulos.

```
select avg( dto )
from   facturas ;  /* AVG ignora els valors nuls */
```

9. Descuento medio aplicado en las facturas considerando los valores nulos como cero.

SQL*PLUS	PostgreSQL
select avg(nvl(dto, 0)) from facturas ;	select sum(dto) / count(*) from facturas ;
	select avg(coalesce(dto,0)) from facturas ;

10. Meses completos transcurridos entre la primera y la última factura del cliente con código 210.

SQL*PLUS	PostgreSQL
select trunc(months_between(max(fecha), min(fecha))) from facturas where codcli = 210 ;	select 12*extract(year from age(max(fecha),min(fecha))) + extract(month from age(max(fecha),min(fecha))) from facturas where codcli=210;

age(DataPosterior, DataAnterior)

11. Número de facturas para cada año.

12. Número de clientes del pueblo con mayor número de clientes.

SQL*PLUS	PostgreSQL
select max(count(*)) from clientes group by codpue ;	**select count(*)** **from clientes** **group by codpue** **order by 1 desc** **limit 1;**
	Select max(T.Num) **From (select count(*) as Num** **from clientes** **group by codpue) as T;**

EN POSTGRES NO PODEM ANIDAR (*NESTED*) FUNCIONS D'AGRUPACIÓ.!!

13. Código de aquellos artículos de los que se ha facturado más de 6000 euros.

14. Iva mínimo para cada mes -de los facturados- del año pasado aplicado en las facturas realizadas por los vendedores cuyo código se halla entre 100 y 200 (incluidos).

15. De los artículos cuyo código termina con la letra "X" más un dígito numérico, mostrar el código y la cantidad total pedida en las líneas de factura.

16. Código de los artículos cuyo código comienza por 'U' y que han sido vendidos siempre con el mismo precio.

	Select codart From (select codart From lineas_fac Where codart like 'U%' Group by codart, precio) Group by codart Having count(*); Caldria analitzar qué passa amb els nuls en el camp *precio*.

17. Para cada vendedor de la provincia de Castellón, mostrar su nombre y el nombre de su jefe inmediato.

18. Fecha de cada factura del mes de diciembre del año pasado junto con todos los detalles de su primera línea.

SQL*PLUS

```
select f.fecha, l.*
from   facturas f, lineas_fac l
where  f.codfac = l.codfac
and    l.linea = 1
and    to_char( f.fecha, 'mm' ) = '12'
and to_number( to_char( f.fecha, 'yyyy' ) )
=
 to_number( to_char( sysdate, 'yyyy' ) ) - 1;
```

PostgreSQL

```
select f.fecha, l.*
from   facturas f, lineas_fac l
where  f.codfac = l.codfac and    l.linea = 1
and    to_char(f.fecha,'mm') = '12'
and  to_number(to_char(f.fecha, 'yyyy'),'9999')
=to_number(to_char(current_date,'yyyy'),'9999') - 1;
```

19. Para cada factura del mes de diciembre del año pasado se desea obtener el código, fecha y nombre del vendedor, ordenado por código de factura.

Ajuda: Uso de una concatenación externa para que no se pierda ninguna factura sin vendedor asignado. En altres paraules, han de sortir tbé les factures amb venedor desconegut.

SQL*PLUS	PostgreSQL
select f.codfac, f.fecha, v.nombre	Select f.codfac, f.fecha, v.nombre

A LEFT JOIN B ≡ A.x (+) = B.x

from facturas f, vendedores v where v.codven (+) = f.codven and to_char(f.fecha, 'mm') = '12' and to_number(to_char(f.fecha, 'yyyy')) = to_number(to_char(sysdate, 'yyyy')) - 1 order by 1 ;	From Facturas F LEFT JOIN Vendedores V on F.codven=V.codven

20. Nombre de los pueblos de Castellón que comienzan por "AR", mostrando el número de facturas realizadas por los clientes de dicho pueblo. Si un pueblo no tiene clientes o sus clientes no tienen facturas, debe aparecer también en el listado.

Ayuda: Uso de dos concatenaciones externas, pues puede haber pueblos sin clientes y clientes sin facturas. Agrupación de las facturas por pueblos.

SQL*PLUS
```
select p.nombre, count( f.codfac )
from   pueblos p, clientes c, facturas f
where  p.codpue = c.codpue (+)
and    c.codcli = f.codcli (+)
and    upper( p.nombre ) like 'AR%'
and    p.codpro = '12'
group  by p.codpue, p.nombre;
```

PostgreSQL
```
From
((Pueblos P  Left Join  Clientes C on
P.codpue=C.Codpue)  Left Join  Facturas F on
C.Codcli=F.codcli)
```

21. Códigos de los clientes de la provincia de Castellón (código de provincia '12') que no tienen facturas.

SQL*PLUS
```
select c.codcli
from   pueblos p, clientes c
where  p.codpue = c.codpue
and    p.codpro = '12'
minus
select f.codcli
from   facturas f;
```

PostgreSQL
```
select c.codcli
from   pueblos p, clientes c
where  p.codpue = c.codpue
and    p.codpro = '12'
except
select f.codcli
from   facturas f;
```

22. Nombre de las provincias con más de 500 pueblos y en las que hay más de 5 pueblos con clientes.

23. Número de clientes que no tienen ninguna factura.

Ayuda: Consulta y subconsulta unidas mediante negación simple. Si se realiza con un "not in" hay que tener cuidado con aquellas facturas cuyo código de clientes es nulo.

```
select count( * )
from   clientes c
where  c.codcli not in ( select f.codcli
                         from   facturas f
                         where  f.codcli is not null ) ;

select count( * )
from   clientes c
where  not exists( select '*'
                   from   facturas f
                   where  f.codcli = c.codcli ) ;
```

Semàntica del IN / NOT IN en Postgres

Ens basem en el cas, per exemple, de

Select ------
From Clientes
Where codcli IN o bé NOT IN (subquery) ➔ Codcli és clau ppal i no té nuls.

- El subquery queda MUT ➔ IN genera un FALSE i NOT IN genera TRUE
- El subquery contesta i no hi ha cap valor NULL en la resposta ➔ Comportament normal d'ambdos predicats.
- El subquery contesta i hi ha algun NULL en la resposta. Distinguerem dos subcasos:
  - El codcli està entre els valors ➔ IN genera TRUE i NOT IN genera FALSE.
  - Cocdli no està ➔ Es genera INDEFINIT (NULL lògic de Postgres) en ambdos casos (IN i NOT IN)

24. Número de clientes que en todas sus facturas tienen un 16% de IVA (los clientes deben tener al menos una factura).

    Ayuda: Consulta con operación "para todos". Solución con doble negación.

SQL*PLUS	PostgreSQL
select count(*) from clientes c where not exists(select '*' from facturas f where f.codcli = c.codcli and nvl(f.iva, 0) <> 16) and exists(select '*' from facturas f where f.codcli = c.codcli) ;	select count(*) from clientes c where not exists(select * from facturas f where f.codcli = c.codcli and coalesce(f.iva, 0) <> 16) and exists(select * from facturas f where f.codcli = c.codcli) ;
Select count(*) from clientes c where 16 = ALL (select nvl(iva,0) from facturas f where f.codcli=c.codcli) and exists(select '*' from facturas f where f.codcli = c.codcli);	Select count(*) from clientes c where 16 = ALL(select coalesce(iva,0) from facturas f where f.codcli=c.codcli) and exists(select * from facturas f where f.codcli = c.codcli);

25. Nombre del vendedor con más facturas.

26. Código y nombre de aquellos vendedores que han relizado más de 15 facturas.

27. Código, descripción y precio de los diez artículos más caros.

28. Nombre del cliente con mayor facturación.

29. Código y descripción de aquellos artículos con un precio superior a la media y que hayan sido comprados por más de 12 clientes.

30. Mostrar, ordenadamente, los clientes que nunca han realizado más de una compra en una fecha determinada.

31. Mostrar, ordenadamente, los clientes que han comprado un mismo artículo en más de una factura.

---

32. Què fa la consulta següent:

```
SELECT art.descrip, TO_CHAR (fac.fecha,'YYYY')
FROM facturas fac, articulos art, lineas_fac lin
WHERE fac.codfac = lin.codfac AND lin.codart = art.codart
GROUP BY TO_CHAR(fac.fecha,'YYYY'),art.codart,art.descrip
```

```
HAVING TO_NUMBER(TO_CHAR(TO_DATE ('3112'||TO_CHAR(fac.fecha,'YYYY'),
'DDMMYYYY'),'WW'))=COUNT (DISTINCT TO_CHAR(fac.fecha,'WW'))
ORDER BY 1;
```

33. Què fa la consulta següent escrita en Postgres?:

```
SELECT codart, precio,
       CASE WHEN precio > 150 THEN ROUND(precio * 0.90, 2)
                             ELSE ROUND(precio * 0.85, 2)
       END AS promocion,
       CASE WHEN precio > 150 THEN '10%'
                             ELSE '15%'
       END AS dto
FROM articulos
WHERE (precio>150 AND stock*precio>=300)
       OR (precio<=150 AND stock*precio>= 150)
ORDER BY codart;
```

34. Què fa la consulta següent. Escriu una consulta alternativa que faça el mateix i no faça servir referència externa.

```
SELECT f.codcli
FROM facturas AS f
WHERE 0 = ALL ( SELECT COALESCE(l.dto,0)
                FROM lineas_fac AS l
                WHERE l.codfac = f.codfac )
GROUP BY f.codcli
HAVING MAX(f.iva) = MIN(f.iva);
```

Esta sentencia obtiene los códigos de los clientes que en las facturas que no tienen ningún descuento de artículo, han pagado siempre el mismo iva.
Una sentencia equivalente, sin referencia externa, es la siguiente:

```
SELECT f.codcli
FROM facturas AS f
WHERE f.codfac IN ( SELECT l.codfac
                    FROM lineas_fac AS l
                    GROUP BY l.codfac
                    HAVING MAX(COALESCE(l.dto,0))=0)
GROUP BY f.codcli
HAVING MAX(f.iva) = MIN(f.iva);
```