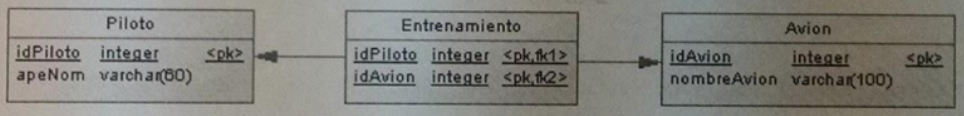
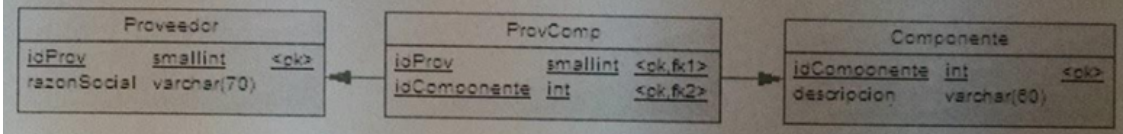
4. Dado el siguiente modelo de datos:



Obtener apellido y nombres de los pilotos que están entrenados para pilotear todos los aviones de la tabla `Avion`. Resolver sobre SQL Server sin utilizar ninguna extensión PSM (solo sentencias ANSI estándar).

```sql
select P.apeNom from Piloto P
inner join Entrenamiento E on P.idPiloto=E.idPiloto
inner join Avion A on E.idAvion=A.idAvion
where
(select count(*) from avion)
=
(select count(*) from entrenamiento where idpiloto = P.idpiloto)
```

2. Dado el siguiente modelo de datos, que representa componentes electrónicos y proveedores que los proveen:

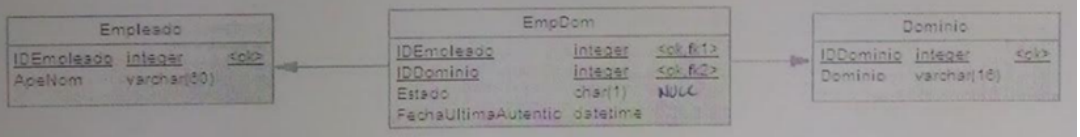


...obtener la razón social de los proveedores que, como mínimo, proveen todos los componentes que provee el proveedor con `idProv` con valor `200`.

Resolver sobre cualquier motor de base de datos con una única sentencia `SELECT`, sin utilizar tablas auxiliares ni extensiones PSM.

```sql
-- ejercicio 3 sql directo
/* obtener provedores que como minomo proveen todos los componentes que provee el proveedor id = 200*/
select P.razonSocial from proveedor P
where
(select count(*) from proveedor P1 -- cantidad de componentes vendidos por prov 200
inner join provcomp PC1 on P1.idprov = PC1.idprov
inner join componente C1 on C1.idcomponente = PC1.idcomponente
where P1.idprov = P.idprov and P1.idprov <> 200)
>=
(select count(*) from proveedor P2 -- cantidad de componentes vendidos por prov 200
inner join provcomp PC2 on P2.idprov = PC2.idprov
inner join componente C2 on C2.idcomponente = PC2.idcomponente
where P2.idprov = 200)
```

2015 **2.** En una empresa, cada empleado posee una serie de dominios de red ante los cuales posee credenciales para autenticarse. Sin embargo, por políticas de seguridad de la empresa, cada empleado, en un momento dado, posee sus cuentas de dominio desactivadas excepto una, que se denomina cuenta activa.
En un modelo de datos, esto está representado de la siguiente manera:



La columna Estado puede tener un valor "A" o "I", de *Activa* e *Inactiva* respectivamente.
Solo puede haber **una** ocurrencia empleado/dominio con estado "A" en un momento dado.
El sistema considera que, por omisión, el dominio activo es el utilizado más recientemente, pero puede haber un dominio establecido como activo sin ser el usado más recientemente

Se acaban de importar datos a la tabla EmpDom y existen ocurrencias que poseen un valor NULL para la columna Estado.
Ejecute una sentencia SQL que establezca dominios activos para estos empleados.

Resolver sin utilizar vistas, tablas auxiliares ni extensiones T-SQL

```
1  declare micursor cursor for select * from EmpDom for update;
2  declare @newstate char(1)
3  declare @idempleado integer
4  declare @iddominio INTEGER
5  declare @fecha_autent DATETIME
6  declare @idmayoractivo integer
7
8
9  open micursor
10 fetch next from micursor into @idempleado,@iddominio,@newstate,@fecha_autent
11 while @@FETCH_STATUS=0
12 begin
13
14 if(@newstate is NULL)
15         update EmpDom set state='A' where current of micursor
16         -- obtenemos el id del de fecha mas reciente
17         set @idmayoractivo = (select top 1 D.idDominio from EmpDom D
18                                                 where D.Estado = 'A'
19                                                 order by D.fechaUltimaAutentic desc )
20         update EmpDom set Estado = 'I' where @idmayoractivo <> idDominio
21
22 end if
23
24 fetch next from micursor into @newstate
25 end --end while
26
27 close micursor
28 deallocate micursor
```

Los consumidores son almacenados en la siguiente tabla:

Consumidor2		
IdConsumidor	integer	<pk>
Apellido	varchar(50)	
Nombres	varchar(50)	
Domicilio	varchar(100)	
Email	varchar(30)	
FamiliarDe	integer	
FechaAlta	datetime	

En esta lista, la columna FamiliarDe asocia consumidores con el mismo domicilio.
Este dato se necesita ya que una regla de negocios de la empresa expresa que se debe enviar un email de publicidad a un único miembro de una familia.
Se considera que se trata de una familia cuando el domicilio es el mismo.

Dada una tupla cualquiera, la columna FamiliarDe apunta al ID de la primer persona en la lista (de existir) que posee el mismo domicilio que esa tupla.

Por ejemplo, el siguiente es un lote de datos existente en la tabla:

Results Messages

	IdConsumidor	Apellido	Nombres	Domicilio	Email	FamiliarDe	FechaAlta
1	1	Heredia	Javier	San Martín 85	jheredia@gmail.com	NULL	2021-02-14 09:00:00.000
2	2	Barbiero	Mateo	La Paz 800	mlbarbiero@hotmail.com	NULL	2021-02-23 07:16:00.000
3	3	Wilder	Alan	San Luis 800	awilder@gmail.com	NULL	2021-03-18 07:40:00.000
4	4	Sotelo	Analía	San Martín 85	ana_sotel@gmail.com	1	2021-03-21 17:05:00.000
5	5	Lopez	Javier	San Luis 800	lopez_123@gmail.com	3	2021-04-11 22:00:00.000
6	6	Fletcher	Andrés	Belgrano 54	fletcher3@hotmail.com	NULL	2021-05-03 07:09:33.000

...el consumidor Sotelo se considera familiar del consumidor Heredia, y el consumidor López se considera familiar del consumidor Wilder.

Se necesita implementar un trigger T-SQL que, ante la inserción de un nuevo consumidor, su columna FamiliarDe apunte -de existir- al consumidor más antiguo (cargado más antiguamente) con el mismo domicilio.
Si no existe consumidor más antiguo con el mismo domicilio, la columna FamiliarDe se deja con un valor NULL.

```
1   -- agregar id de familiar(mas viejo) a familiarde(insertado)
2   create trigger finalbd009
3   on consumidor2 for insert
4   as
5   declare @domicilio varchar(100) = (select domicilio from inserted);
6   declare @id integer = (select idconsumidor from inserted);
7   declare @idfamiliarde integer;
8   declare @idfamiliar integer;
9
10  begin
11  set @idfamiliar = (select top 1 c.idconsumidor from consumidor2 c
12                                      where c.domicilio = @domicilio and c.idconsumidor <> @id
13                                      order by c.fechaalta);
14  if(@idfamiliar is not null)
15  begin
16          print 'Tiene familiar';
17          update consumidor2 set familiarde = @idfamiliar where idconsumidor = @id
18  end
19  else
20          print 'No tiene familiar'
21  end
```

3. Una empresa realiza promociones a través de listas de correo de consumidores.

Los consumidores son almacenados en la siguiente tabla:

Consumidor		
IdConsumidor	integer	<pk>
Apellido	varchar(50)	
Nombres	varchar(50)	
Domicilio	varchar(100)	
Email	varchar(30)	

Hay consumidores que poseen un mismo domicilio.
Cuando esto sucede, se considera que se trata de una familia.

Importante: Tenga en cuenta que esta tabla posee una estructura diferente a la del Ejercicio 1. Carece de las columnas *FamiliarDe* y *FechaAlta*.

El siguiente es un lote de datos existente en la tabla:

Data Output | Explain | Messages | History

	idconsumidor integer	apellido character varying(50)	nombres character varying(50)	domicilio character varying(100)	email character varying(30)
1	1	Heredia	Javier	San Martín 85	jheredia@gmail.com
2	2	Barbiero	Mateo	La Paz 800	mlbarbiero@hotmail.com
3	3	Wilder	Alan	San Luis 800	awilder@gmail.com
4	4	Sotelo	Analía	San Martín 85	ana sotel@gmail.com
5	6	Fletcher	Andrés	Belgrano 54	fletcher3@hotmail.com
6	5	Lopez	Javier	San Luis 800	lopez 123@gmail.com

...el consumidor Sotelo se considera familiar del consumidor Heredia, y el consumidor López se considera familiar del consumidor Wilder.

Implemente una function PL/pgsql que retorne una salida como la siguiente:

Data Output | Explain | Messages | History

	apellido character varying(50)	nombres character varying(50)	posee text	cantidad bigint
1	Heredia	Javier	Si	1
2	Barbiero	Mateo	No	
3	Wilder	Alan	Si	1
4	Sotelo	Analía	Si	1
5	Fletcher	Andrés	No	
6	Lopez	Javier	Si	1

La columna posee indica si el consumidor posee o no otro conviviente además del de la tupla en cuestión.

```sql
create type datosper as ( -- Creo el struct que voy a retornar
    apellido character varying(50),
    nombres character varying(50),
    posee text,
    cantidad bigint
);
create or replace function finalbd009() returns setof datosper
language plpgsql
as
$$
declare retorno datosper; -- esto es lo que voy a retornar
declare familiares integer; -- variable que me va a indicar cantidad de familiares
declare mconsumidor record; -- variable consumidor para el cursor
begin
for mconsumidor in (select * from consumidor) loop -- cursor implicito

        retorno.apellido := mconsumidor.apellido;
        retorno.nombres := mconsumidor.nombres;
        familiares := (select count(*) from consumidor
                       where domicilio = mconsumidor.domicilio
                       and idconsumidor <> mconsumidor.idconsumidor); -- calculo los familiares del consumidor n
        if(familiares > 0) then -- entra si tengo familiares
            retorno.posee := 'Si';
            retorno.cantidad := familiares;
        else -- entra si no tengo familiares
            retorno.posee := 'No';
            retorno.cantidad := null;
        end if;
            return next retorno; -- retorno la salida
    end loop;
end;
$$
```

```
CREATE TABLE titles
(
    title_id        varchar(6)
        CONSTRAINT UPKCL_titleidind PRIMARY KEY CLUSTERED,
    title           varchar(80)     NOT NULL,
    type            char(12)        NOT NULL
        DEFAULT ('UNDECIDED'),
    pub_id          char(4)             NULL
        REFERENCES publishers(pub_id),
    price           money               NULL,
    advance         money               NULL,
    royalty         int                 NULL,
    ytd_sales       int                 NULL,
    notes           varchar(200)        NULL,
    pubdate         datetime        NOT NULL
        DEFAULT (getdate())
)


CREATE TABLE sales
(
    stor_id         char(4)         NOT NULL
        REFERENCES stores(stor_id),
    ord_num         varchar(20)     NOT NULL,
    ord_date        datetime        NOT NULL,
    qty             smallint        NOT NULL,
    payterms        varchar(12)     NOT NULL,
    title_id        tid
        REFERENCES titles(title_id),
    CONSTRAINT UPKCL_sales PRIMARY KEY CLUSTERED (stor_id, ord_num, title_id)
)

GO
```

Se desea generar una tabla (MesMaximasVentas) con la siguiente estructura, que liste los diferentes años en los que se produjeron ventas junto al mes en que se vendió la mayor cantidad de publicaciones:

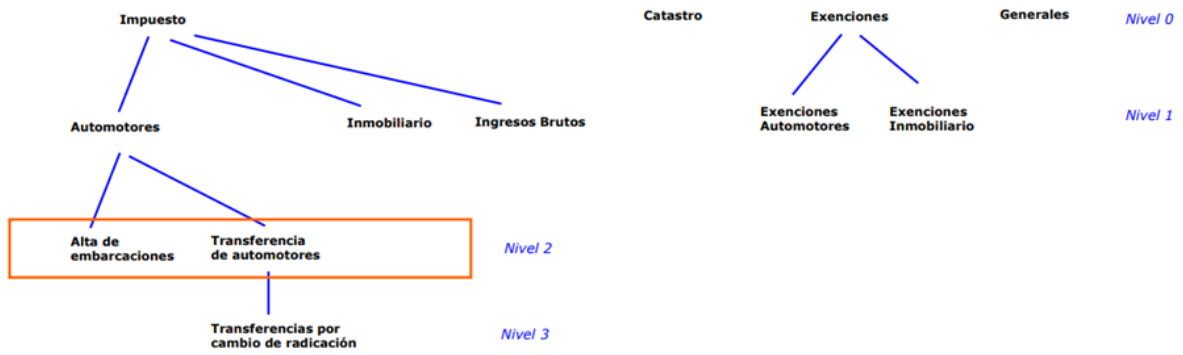
anio	mes
1992	6
1993	5
1994	9

La columna anio posee el año (extraído de la columna ord_date) y es de tipo INTEGER, y la columna mes posee el mes en que más ventas se registraron y también es de tipo INTEGER.

Resolver utilizando T-SQL o PL/pgSQL.

```sql
-- obtener los meses en los que se genero mayores ventas de las publicaciones por año
SELECT YEAR(SM.ord_date),
(SELECT TOP 1 month(S.ord_date) FROM sales S
        inner join titles T ON S.title_id = T.title_id
        GROUP BY YEAR(S.ord_date),MONTH(S.ord_date)
        HAVING YEAR(S.ord_date) = YEAR(SM.ord_date)
        ORDER BY SUM(S.qty*T.price) DESC)
FROM sales SM
GROUP BY YEAR(SM.ord_date)
```

Se solicita listar las categorías del Nivel 2, indicando además si las mismas poseen o no elementos hijo.



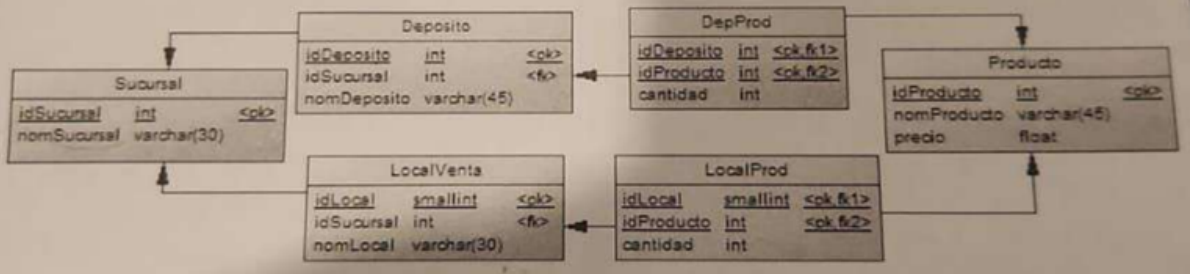
Para nuestro ejemplo deberíamos obtener:

Results | Messages

	Categoria	TieneHijos?
1	Alta de embarcaciones	No tiene hijos
2	Transferencia de automotores	TieneHijos

```
1  --select * from categoria where idcategpadre is null; -- padres
2  --select * from categoria where idcategpadre in (select idcateg from categoria where idcategpadre is null); -- primer nivel
3  select C.descrip as categoria,
4   case   when (select count(*) from categoria where idcategpadre = C.idcateg) > 0 then 'tiene hijos'
5          else 'No tiene hijos'
6   end
7  from categoria C where idcategpadre in
8                 (select idcateg from categoria where idcategpadre in
9                 (select idcateg from categoria where idcategpadre is null)); -- segundo nivel
10
```

El siguiente es un modelo de datos que representa Depósitos y Locales de venta de una empresa distribuidora:



Obtener el nombre de todas las sucursales que posean más cantidad de artículos en sus depósitos que en sus locales de venta.

Resolver utilizando SQL directo, sin utilizar vistas, tablas auxiliares ni extensiones T-SQL o PL/pgSQL.
No utilizar sentencias o comandos no vistos en las Guías de Estudio.

```sql
-- ejercicio 2 sql directo
-- obtener sucursales que tengan mas productos en sus depositos que en sus locales de venta
select * from sucursal SM
where
(select sum(cantidad) from sucursal S -- aca retorno la cantidad de productos que tiene en todos sus depositos
inner join deposito D on S.idsucursal = D.idsucursal
inner join depprod DP on DP.iddeposito = D.iddeposito
inner join producto P on P.idproducto = DP.idproducto
where S.idsucursal = SM.idsucursal)
>
(select sum(LP.cantidad) from sucursal S -- aca retorno la cantidad de productos que tiene en todos sus locales
inner join localventa L on S.idsucursal = L.idsucursal
inner join localprod LP on LP.idlocal = L.idlocal
inner join producto P on P.idproducto = LP.idproducto
where S.idsucursal = SM.idsucursal)
```

La tabla debería quedar -luego de realizado el proceso- como muestra la siguiente captura:

Results | Messages

IDTurno	IDPuestoAtencion	FechaHoraAtencion	Contribuyente	IDTramite
1	1	2021-11-29 07:30:00.000	Villaverde Felipe	1
2	2	2021-11-29 07:30:00.000	Giraudo Lucrecia	3
3	3	2021-11-29 07:30:00.000	Bernarducci Pablo	7
6	1	2021-11-29 07:45:00.000	Gómez Esteban	5
9	4	2021-11-29 07:45:00.000	NULL	NULL
10	5	2021-11-29 07:45:00.000	NULL	NULL
13	3	2021-11-29 08:00:00.000	Ruiz María Luisa	2
14	4	2021-11-29 08:00:00.000	NULL	NULL
15	5	2021-11-29 08:00:00.000	NULL	NULL
18	3	2021-11-29 09:00:00.000	Gervasoni Verón...	6
19	4	2021-11-29 09:00:00.000	NULL	NULL
20	5	2021-11-29 09:00:00.000	NULL	NULL

De las filas que corresponda eliminar, se debe tener en cuenta que se deben eliminar todas o no se debe eliminar ninguna.
En nuestro ejemplo correspondía eliminar 8 filas. O se eliminan las ocho o no se elimina ninguna.

Resolver utilizando T-SQL. No utilizar sentencias no vistas en las Guías de estudio.

```
begin transaction
declare mcursor cursor for (select fechahoraatencion from turno
                            group by fechahoraatencion);
declare @idturno1 integer;
declare @idturno2 integer;
declare @hora datetime;
declare @nulos integer;

open mcursor;
fetch next from mcursor into @hora;
while(@@FETCH_STATUS=0)begin

    set @nulos = (select COUNT(*) from turno
                  where contribuyente is null and idtramite is null
                  and fechahoraatencion = @hora);
    if(@nulos < 2) rollback transaction;
    else
    begin
    set @idturno1 = (select top 1 idturno from turno
                     where contribuyente is null and idtramite is null
                     order by idturno desc)
    set @idturno2 = (select top 1 idturno from turno
                     where contribuyente is null and idtramite is null
                     and idturno <> @idturno1
                     order by idturno desc)
    delete from turno where idturno = @idturno1;
    delete from turno where idturno = @idturno2;
    end

fetch next from mcursor into @hora;
end -- end while
commit transaction
close mcursor;
deallocate mcursor;
```

coloquio 2022

```sql
create table resultado(
nombre varchar(60),
trimestre1 varchar(60),
trimestre2 varchar(60),
bool varchar(2)
);
declare mcursor cursor for (select pub_id,pub_name from publishers);
declare @pubid integer;
declare @pubname varchar(60);
declare @trimestre1 varchar(60);
declare @bool varchar(2);
declare @trimestre2 varchar(60);
declare @idpubmasvendio integer = (select top 1 p.pub_id from publishers p
                                   inner join titles t on t.pub_id=p.pub_id
                                   inner join sales s on s.title_id=t.title_id
                                   group by p.pub_id
                                   order by sum(s.qty) desc);
open mcursor;
fetch next from mcursor into @pubid,@pubname;
while @@FETCH_STATUS=0 begin
    set @trimestre1 = (select top 1 t.title_id from titles t
                       inner join sales s on s.title_id=t.title_id
                       where t.pub_id = @pubid and MONTH(s.ord_date) between 1 and 3
                       group by t.title_id
                       order by SUM(s.qty * t.price) desc)
    set @trimestre2 = (select top 1 t.title_id from titles t
                       inner join sales s on s.title_id=t.title_id
                       where t.pub_id = @pubid and MONTH(s.ord_date) between 3 and 6
                       group by t.title_id
                       order by SUM(s.qty * t.price) desc)
    if(@pubid = @idpubmasvendio) set @bool = 'S'
    else set @bool = 'N'

    insert into resultado
    values
    (@pubname, @trimestre1,@trimestre2,@bool);


fetch next from mcursor into @pubid,@pubname;
end
close mcursor
deallocate mcursor
select * from resultado
```