76 - Restricciones (foreign key)

Primer problema:

Una empresa tiene registrados sus clientes en una tabla llamada "clientes", también tiene una tabla
"provincias" donde registra los nombres de las provincias.
1- Elimine las tablas "clientes" y "provincias", si existen y créelas:

```
 if object_id('clientes') is not null
  drop table clientes;
 if object_id('provincias') is not null
  drop table provincias;

 create table clientes (
  codigo int identity,
  nombre varchar(30),
  domicilio varchar(30),
  ciudad varchar(20),
  codigoprovincia tinyint
 );

 create table provincias(
  codigo tinyint not null,
  nombre varchar(20)
 );
```

En este ejemplo, el campo "codigoprovincia" de "clientes" es una clave foránea, se emplea para
enlazar la tabla "clientes" con "provincias".

2- Intente agregar una restricción "foreign key" a la tabla "clientes" que haga referencia al campo
"codigo" de "provincias":

```
 alter table clientes
 add constraint FK_clientes_codigoprovincia
  foreign key (codigoprovincia)
  references provincias(codigo);
```

No se puede porque "provincias" no tiene restricción "primary key" ni "unique".

3- Establezca una restricción "primary key" al campo "codigo" de "provincias":

```
 alter table provincias
 add constraint PK_provincias_codigo
  primary key (codigo);
```

4- Ingrese algunos registros para ambas tablas:

```
 insert into provincias values(1,'Cordoba');
 insert into provincias values(2,'Santa Fe');
 insert into provincias values(3,'Misiones');
 insert into provincias values(4,'Rio Negro');

 insert into clientes values('Perez Juan','San Martin 123','Carlos Paz',1);
 insert into clientes values('Moreno Marcos','Colon 234','Rosario',2);
 insert into clientes values('Acosta Ana','Avellaneda 333','Posadas',3);
 insert into clientes values('Luisa Lopez','Juarez 555','La Plata',6);
```

5- Intente agregar la restricción "foreign key" del punto 2 a la tabla "clientes":

```
 alter table clientes
 add constraint FK_clientes_codigoprovincia
  foreign key (codigoprovincia)
  references provincias(codigo);
```

No se puede porque hay un registro en "clientes" cuyo valor de "codigoprovincia" no existe en "provincias".

6- Elimine el registro de "clientes" que no cumple con la restricción y establezca la restricción nuevamente:

```
 delete from clientes where codigoprovincia=6;
 alter table clientes
 add constraint FK_clientes_codigoprovincia
  foreign key (codigoprovincia)
  references provincias(codigo);
```

7- Intente agregar un cliente con un código de provincia inexistente en "provincias".
No se puede.

8- Intente eliminar el registro con código 3, de "provincias".
No se puede porque hay registros en "clientes" al cual hace referencia.

9- Elimine el registro con código "4" de "provincias".
Se permite porque en "clientes" ningún registro hace referencia a él.

10- Intente modificar el registro con código 1, de "provincias".
No se puede porque hay registros en "clientes" al cual hace referencia.

11- Vea las restricciones de "clientes".
aparece la restricción "foreign key".

12- Vea las restricciones de "provincias".
aparece la restricción "primary key" y nos informa que la tabla es rerenciada por una "foreign key"
de la tabla "clientes" llamada "FK_clientes_codigoprovincia".

77 - Restricciones foreign key en la misma tabla

Primer problema:

Una empresa registra los datos de sus clientes en una tabla llamada "clientes". Dicha tabla contiene
un campo que hace referencia al cliente que lo recomendó denominado "referenciadopor". Si un cliente
no ha sido referenciado por ningún otro cliente, tal campo almacena "null".
1- Elimine la tabla si existe y créela:

```
 if object_id('clientes') is not null
  drop table clientes;
 create table clientes(
  codigo int not null,
  nombre varchar(30),
  domicilio varchar(30),
  ciudad varchar(20),
  referenciadopor int,
  primary key(codigo)
 );
```

2- Ingresamos algunos registros:

```
 insert into clientes values (50,'Juan Perez','Sucre 123','Cordoba',null);
 insert into clientes values(90,'Marta Juarez','Colon 345','Carlos Paz',null);
 insert into clientes values(110,'Fabian Torres','San Martin 987','Cordoba',50);
 insert into clientes values(125,'Susana Garcia','Colon 122','Carlos Paz',90);
 insert into clientes values(140,'Ana Herrero','Colon 890','Carlos Paz',9);
```

3- Intente agregar una restricción "foreign key" para evitar que en el campo "referenciadopor" se
ingrese un valor de código de cliente que no exista.
No se permite porque existe un registro que no cumple con la restricción que se intenta establecer.

4- Cambie el valor inválido de "referenciadopor" del registro que viola la restricción por uno
válido.

5- Agregue la restricción "foreign key" que intentó agregar en el punto 3.

6- Vea la información referente a las restricciones de la tabla "clientes".

7- Intente agregar un registro que infrinja la restricción.
No lo permite.

8- Intente modificar el código de un cliente que está referenciado en "referenciadopor".
No se puede.

9- Intente eliminar un cliente que sea referenciado por otro en "referenciadopor".
No se puede.

10- Cambie el valor de código de un cliente que no referenció a nadie.

11- Elimine un cliente que no haya referenciado a otros.

78 - Restricciones foreign key (acciones)

Primer problema:

Una empresa tiene registrados sus clientes en una tabla llamada "clientes", también tiene una tabla
"provincias" donde registra los nombres de las provincias.
1- Elimine las tablas "clientes" y "provincias", si existen:

```
 if object_id('clientes') is not null
  drop table clientes;
 if object_id('provincias') is not null
  drop table provincias;
```

2- Créelas con las siguientes estructuras:

```
 create table clientes (
  codigo int identity,
  nombre varchar(30),
  domicilio varchar(30),
  ciudad varchar(20),
  codigoprovincia tinyint,
```

```
  primary key(codigo)
 );

 create table provincias(
  codigo tinyint,
  nombre varchar(20),
  primary key (codigo)
 );
```

3- Ingrese algunos registros para ambas tablas:

```
 insert into provincias values(1,'Cordoba');
 insert into provincias values(2,'Santa Fe');
 insert into provincias values(3,'Misiones');
 insert into provincias values(4,'Rio Negro');

 insert into clientes values('Perez Juan','San Martin 123','Carlos Paz',1);
 insert into clientes values('Moreno Marcos','Colon 234','Rosario',2);
 insert into clientes values('Acosta Ana','Avellaneda 333','Posadas',3);
```

4- Establezca una restricción "foreign key" especificando la acción "en cascade" para
actualizaciones y "no action" para eliminaciones.

5- Intente eliminar el registro con código 3, de "provincias".
No se puede porque hay registros en "clientes" al cual hace referencia y la opción para
eliminaciones se estableció como "no action".

6- Modifique el registro con código 3, de "provincias".

7- Verifique que el cambio se realizó en cascada, es decir, que se modificó en la tabla "provincias"
y en "clientes":

```
 select * from provincias;
 select * from clientes;
```

8- Intente modificar la restricción "foreign key" para que permita eliminación en cascada.
Mensaje de error, no se pueden modificar las restricciones.

9- Intente eliminar la tabla "provincias".
No se puede eliminar porque una restricción "foreign key" hace referencia a ella.

79 - Restricciones foreign key deshabilitar y eliminar (with check - nocheck)

Primer problema:

Una empresa tiene registrados sus clientes en una tabla llamada "clientes", también tiene una tabla
"provincias" donde registra los nombres de las provincias.
1- Elimine las tablas "clientes" y "provincias", si existen:

```
 if object_id('clientes') is not null
  drop table clientes;
 if object_id('provincias') is not null
  drop table provincias;
```

2- Créelas con las siguientes estructuras:

```
 create table clientes (
  codigo int identity,
  nombre varchar(30),
  domicilio varchar(30),
  ciudad varchar(20),
  codigoprovincia tinyint,
  primary key(codigo)
 );

 create table provincias(
  codigo tinyint,
  nombre varchar(20),
  primary key (codigo)
 );
```

3- Ingrese algunos registros para ambas tablas:

```
 insert into provincias values(1,'Cordoba');
 insert into provincias values(2,'Santa Fe');
 insert into provincias values(3,'Misiones');
 insert into provincias values(4,'Rio Negro');

 insert into clientes values('Perez Juan','San Martin 123','Carlos Paz',1);
 insert into clientes values('Moreno Marcos','Colon 234','Rosario',2);
 insert into clientes values('Garcia Juan','Sucre 345','Cordoba',1);
 insert into clientes values('Lopez Susana','Caseros 998','Posadas',3);
 insert into clientes values('Marcelo Moreno','Peru 876','Viedma',4);
 insert into clientes values('Lopez Sergio','Avellaneda 333','La Plata',5);
```

4- Intente agregar una restricción "foreign key" para que los códigos de provincia de "clientes"

existan en "provincias" con acción en cascada para actualizaciones y eliminaciones, sin especificar
la opción de comprobación de datos:

```
 alter table clientes
  add constraint FK_clientes_codigoprovincia
  foreign key (codigoprovincia)
  references provincias(codigo)
  on update cascade
  on delete cascade;
```

No se puede porque al no especificar opción para la comprobación de datos, por defecto es "check" y
hay un registro que no cumple con la restricción.

5- Agregue la restricción anterior pero deshabilitando la comprobación de datos existentes:

```
 alter table clientes
  with nocheck
  add constraint FK_clientes_codigoprovincia
  foreign key (codigoprovincia)
  references provincias(codigo)
  on update cascade
  on delete cascade;
```

6- Vea las restricciones de "clientes":

```
 sp_helpconstraint clientes;
```

Aparece la restricción "primary key" y "foreign key", las columnas "delete_action" y "update_action"
contienen "cascade" y la columna "status_enabled" contiene "Enabled".

7- Vea las restricciones de "provincias":

```
 sp_helpconstraint provincias;
```

Aparece la restricción "primary key" y la referencia a esta tabla de la restricción "foreign key" de
la tabla "clientes.

8- Deshabilite la restricción "foreign key" de "clientes":

```
 alter table clientes
 nocheck constraint FK_clientes_codigoprovincia;
```

9- Vea las restricciones de "clientes":

```
 exec sp_helpconstraint clientes;
```

la restricción "foreign key" aparece inhabilitada.

10- Vea las restricciones de "provincias":

exec sp_helpconstraint provincias;
informa que la restricción "foreign key" de "clientes" hace referencia a ella, aún cuando está
deshabilitada.

11- Agregue un registro que no cumpla la restricción "foreign key":
insert into clientes values('Garcia Omar','San Martin 100','La Pampa',6);
Se permite porque la restricción está deshabilitada.

12- Elimine una provincia de las cuales haya clientes:
delete from provincias where codigo=2;

13- Corrobore que el registro se eliminó de "provincias" pero no se extendió a "clientes":
select * from clientes;
select * from provincias;

14- Modifique un código de provincia de la cual haya clientes:
update provincias set codigo=9 where codigo=3;

15- Verifique que el cambio se realizó en "provincias" pero no se extendió a "clientes":
select * from clientes;
select * from provincias;

16- Intente eliminar la tabla "provincias":
drop table provincias;
No se puede porque la restricción "FK_clientes_codigoprovincia" la referencia, aunque esté deshabilitada.

17- Habilite la restricción "foreign key":
alter table clientes
check constraint FK_clientes_codigoprovincia;

18- Intente agregar un cliente con código de provincia inexistente en "provincias":
insert into clientes values('Hector Ludueña','Paso 123','La Plata',8);
No se puede.

19- Modifique un código de provincia al cual se haga referencia en "clientes":
update provincias set codigo=20 where codigo=4;
Actualización en cascada.

20- Vea que se modificaron en ambas tablas:
```
select * from clientes;
select * from provincias;
```

21- Elimine una provincia de la cual haya referencia en "clientes":
```
delete from provincias where codigo=1;
```
Acción en cascada.

22- Vea que los registros de ambas tablas se eliminaron:
```
select * from clientes;
select * from provincias;
```

23- Elimine la restriccion "foreign key":
```
alter table clientes
drop constraint FK_clientes_codigoprovincia;
```

24- Vea las restriciones de la tabla "provincias":
```
exec sp_helpconstraint provincias;
```
Solamente aparece la restricción "primary key", ya no hay una "foreign key" que la referencie.

25- Elimine la tabla "provincias":
```
drop table provincias;
```
Puede eliminarse porque no hay restricción "foreign key" que la referencie.

81 - Restricciones al crear la tabla

Primer problema:

Un club de barrio tiene en su sistema 4 tablas:
- "socios": en la cual almacena documento, número, nombre y domicilio de cada socio;
- "deportes": que guarda un código, nombre del deporte, día de la semana que se dicta y documento
del profesor instructor;
- "profesores": donde se guarda el documento, nombre y domicilio de los profesores e
- "inscriptos": que almacena el número de socio, el código de deporte y si la matricula está paga o
no.

1- Elimine las tablas si existen:
```
if object_id('inscriptos') is not null
  drop table inscriptos;
```

```
 if object_id('socios') is not null
  drop table socios;
 if object_id('profesores') is not null
  drop table profesores;
 if object_id('deportes') is not null
  drop table deportes;
```

2- Considere que:
- un socio puede inscribirse en varios deportes, pero no dos veces en el mismo.
- un socio tiene un documento único y un número de socio único.
- el documento del socio debe contener 8 dígitos.
- un deporte debe tener asignado un profesor que exista en "profesores" o "null" si aún no tiene un
instructor definido.
- el campo "dia" de "deportes" puede ser: lunes, martes, miercoles, jueves, viernes o sabado.
- el campo "dia" de "deportes" por defecto debe almacenar 'sabado'.
- un profesor puede ser instructor de varios deportes o puede no dictar ningún deporte.
- un profesor no puede estar repetido en "profesores".
- el documento del profesor debe contener 8 dígitos.
- un inscripto debe ser socio, un socio puede no estar inscripto en ningún deporte.
- una inscripción debe tener un valor en socio existente en "socios" y un deporte que exista en
"deportes".
- el campo "matricula" de "inscriptos" debe aceptar solamente los caracteres 's' o 'n'.

3- Cree las tablas con las restricciones necesarias:

```
 create table profesores(
  documento char(8) not null,
  nombre varchar(30),
  domicilio varchar(30),
  constraint CK_profesores_documento_patron check (documento like
'[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]'),
  constraint PK_profesores_documento
   primary key (documento)
 );

 create table deportes(
  codigo tinyint identity,
  nombre varchar(20) not null,
```

```
  dia varchar(30)
   constraint DF_deportes_dia default('sabado'),
  profesor char(8),--documento del profesor
  constraint CK_deportes_dia_lista check (dia in
('lunes','martes','miercoles','jueves','viernes','sabado')),
  constraint PK_deportes_codigo
   primary key (codigo)
 );

 create table socios(
  numero int identity,
  documento char(8),
  nombre varchar(30),
  domicilio varchar(30),
  constraint CK_documento_patron check (documento like '[0-9][0-9][0-
9][0-9][0-9][0-9][0-9][0-9]'),
  constraint PK_socios_numero
   primary key nonclustered(numero),
  constraint UQ_socios_documento
   unique clustered(documento)
 );

 create table inscriptos(
  numerosocio int not null,
  codigodeporte tinyint,
  matricula char(1),
  constraint PK_inscriptos_numerodeporte
   primary key clustered (numerosocio,codigodeporte),
  constraint FK_inscriptos_deporte
   foreign key (codigodeporte)
   references deportes(codigo)
   on update cascade,
  constraint FK_inscriptos_socios
   foreign key (numerosocio)
   references socios(numero)
   on update cascade
   on delete cascade,
  constraint CK_matricula_valores check (matricula in ('s','n'))
);

4- Ingrese registros en "profesores":
 insert into profesores values('21111111','Andres Acosta','Avellaneda
111');
 insert into profesores values('22222222','Betina Bustos','Bulnes 222');
```

```
 insert into profesores values('23333333','Carlos Caseros','Colon 333');
```

5- Ingrese registros en "deportes". Ingrese el mismo día para distintos deportes, un deporte sin día
confirmado, un deporte sin profesor definido:

```
 insert into deportes values('basquet','lunes',null);
 insert into deportes values('futbol','lunes','23333333');
 insert into deportes values('natacion',null,'22222222');
 insert into deportes values('padle',default,'23333333');
 insert into deportes (nombre,dia) values('tenis','jueves');
```

6- Ingrese registros en "socios":

```
 insert into socios values('30111111','Ana Acosta','America 111');
 insert into socios values('30222222','Bernardo Bueno','Bolivia 222');
 insert into socios values('30333333','Camila Conte','Caseros 333');
 insert into socios values('30444444','Daniel Duarte','Dinamarca 444');
```

7- Ingrese registros en "inscriptos". Inscriba a un socio en distintos deportes, inscriba varios
socios en el mismo deporte.

```
 insert into inscriptos values(1,3,'s');
 insert into inscriptos values(1,5,'s');
 insert into inscriptos values(2,1,'s');
 insert into inscriptos values(4,1,'n');
 insert into inscriptos values(4,4,'s');
```

8- Realice un "join" (del tipo que sea necesario) para mostrar todos los datos del socio junto con
el nombre de los deportes en los cuales está inscripto, el día que tiene que asistir y el nombre del
profesor que lo instruirá.
5 registros.

9- Realice la misma consulta anterior pero incluya los socios que no están inscriptos en ningún
deporte.
6 registros.

10- Muestre todos los datos de los profesores, incluido el deporte que dicta y el día, incluyendo
los profesores que no tienen asignado ningún deporte.
4 registros.

11- Muestre todos los deportes y la cantidad de inscriptos, incluyendo aquellos deportes para los
cuales no hay inscriptos.
5 registros.

12- Muestre las restricciones de "socios".
3 restricciones y 1 "foreign key" de "inscriptos" que la referencia.

13- Muestre las restricciones de "deportes".
3 restricciones y 1 "foreign key" de "inscriptos" que la referencia.

14- Muestre las restricciones de "profesores".
2 restricciones.

15- Muestre las restricciones de "inscriptos".
4 restricciones.

82 - Unión

Primer problema:

Un supermercado almacena en una tabla denominada "proveedores" los datos de las compañías que le
proveen de mercaderías; en una tabla llamada "clientes", los datos de los comercios que le compran y
en otra tabla "empleados" los datos de los empleados.
1- Elimine las tablas si existen:

```
 if object_id('clientes') is not null
  drop table clientes;
 if object_id('proveedores') is not null
  drop table proveedores;
 if object_id('empleados') is not null
  drop table empleados;
```

2- Cree las tablas:

```
 create table proveedores(
  codigo int identity,
  nombre varchar (30),
  domicilio varchar(30),
  primary key(codigo)
 );
 create table clientes(
  codigo int identity,
  nombre varchar (30),
  domicilio varchar(30),
```

```
  primary key(codigo)
 );
 create table empleados(
  documento char(8) not null,
  nombre varchar(20),
  apellido varchar(20),
  domicilio varchar(30),
  primary key(documento)
 );
```

3- Ingrese algunos registros:

```
 insert into proveedores values('Bebida cola','Colon 123');
 insert into proveedores values('Carnes Unica','Caseros 222');
 insert into proveedores values('Lacteos Blanca','San Martin 987');
 insert into clientes values('Supermercado Lopez','Avellaneda 34');
 insert into clientes values('Almacen Anita','Colon 987');
 insert into clientes values('Garcia Juan','Sucre 345');
 insert into empleados values('23333333','Federico','Lopez','Colon 987');
 insert into empleados values('28888888','Ana','Marquez','Sucre 333');
 insert into empleados values('30111111','Luis','Perez','Caseros 956');
```

4- El supermercado quiere enviar una tarjeta de salutación a todos los proveedores, clientes y
empleados y necesita el nombre y domicilio de todos ellos. Emplee el operador "union" para obtener
dicha información de las tres tablas.

5- Agregue una columna con un literal para indicar si es un proveedor, un cliente o un empleado y
ordene por dicha columna.

83 - Agregar y eliminar campos (alter table - add - drop)

Primer problema:

Trabaje con una tabla llamada "empleados".
1- Elimine la tabla, si existe, créela y cargue un registro:

```
 if object_id('empleados') is not null
  drop table empleados;

 create table empleados(
  apellido varchar(20),
  nombre varchar(20),
  domicilio varchar(30),
  fechaingreso datetime
```

```
 );
 insert into empleados(apellido,nombre) values ('Rodriguez','Pablo');
```

2- Agregue el campo "sueldo", de tipo decimal(5,2).

3- Verifique que la estructura de la tabla ha cambiado.

4- Agregue un campo "codigo", de tipo int con el atributo "identity".

5- Intente agregar un campo "documento" no nulo.
No es posible, porque SQL Server no permite agregar campos "not null" a menos que se especifique un
valor por defecto.

6- Agregue el campo del punto anterior especificando un valor por defecto:

```
 alter table empleados
  add documento char(8) not null default '00000000';
```

7- Verifique que la estructura de la tabla ha cambiado.

8- Elimine el campo "sueldo".

9- Verifique la eliminación:

```
 exec sp_columns empleados;
```

10- Intente eliminar el campo "documento".
no lo permite.

11- Elimine los campos "codigo" y "fechaingreso" en una sola sentencia.

12- Verifique la eliminación de los campos:

```
 exec sp_columns empleados;
```

84 - Alterar campos (alter table - alter)

Primer problema:

Trabaje con una tabla llamada "empleados".
1- Elimine la tabla, si existe y créela:

```
 if object_id('empleados') is not null
  drop table empleados;

 create table empleados(
  legajo int not null,
```

```
  documento char(7) not null,
  nombre varchar(10),
  domicilio varchar(30),
  ciudad varchar(20) default 'Buenos Aires',
  sueldo decimal(6,2),
  cantidadhijos tinyint default 0,
  primary key(legajo)
 );
```

2- Modifique el campo "nombre" extendiendo su longitud.

3- Controle la modificación:

```
 sp_columns empleados;
```

4- Modifique el campo "sueldo" para que no admita valores nulos.

4- Modifique el campo "documento" ampliando su longitud a 8 caracteres.

5- Intente modificar el tipo de datos del campo "legajo" a "tinyint":

```
 alter table empleados
  alter column legajo tinyint not null;
```

No se puede porque tiene una restricción.

6- Ingrese algunos registros, uno con "nombre" nulo:

```
 insert into empleados values(1,'22222222','Juan Perez','Colon 123','Cordoba',500,3);
 insert into empleados values(2,'30000000',null,'Sucre 456','Cordoba',600,2);
```

7- Intente modificar el campo "nombre" para que no acepte valores nulos:

```
 alter table empleados
  alter column nombre varchar(30) not null;
```

No se puede porque hay registros con ese valor.

8- Elimine el registro con "nombre" nulo y realice la modificación del punto 7:

```
 delete from empleados where nombre is null;
 alter table empleados
  alter column nombre varchar(30) not null;
```

9- Modifique el campo "ciudad" a 10 caracteres.

10- Intente agregar un registro con el valor por defecto para "ciudad":
```
 insert into empleados values(3,'33333333','Juan Perez','Sarmiento
856',default,500,4);
```
No se puede porque el campo acepta 10 caracteres y el valor por defecto tiene 12 caracteres.

11- Modifique el campo "ciudad" sin que afecte la restricción dándole una longitud de 15 caracteres.

12- Agregue el registro que no pudo ingresar en el punto 10:
```
 insert into empleados values(3,'33333333','Juan Perez','Sarmiento
856',default,500,4);
```

13- Intente agregar el atributo identity de "legajo".
No se puede agregar este atributo.

85 - Agregar campos y restricciones (alter table)

Primer problema:

Trabaje con una tabla llamada "empleados".
1- Elimine la tabla, si existe y créela:
```
 if object_id('empleados') is not null
  drop table empleados;

 create table empleados(
  documento char(8) not null,
  nombre varchar(10),
  domicilio varchar(30),
  ciudad varchar(20) default 'Buenos Aires'
 );
```

2- Agregue el campo "legajo" de tipo int identity y una restricción "primary key":
```
 alter table empleados
  add legajo int identity
  constraint PK_empleados_legajo primary key;
```

3- Vea si la estructura cambió y si se agregó la restricción:
```
 sp_columns empleados;
 exec sp_helpconstraint empleados;
```

4- Agregue el campo "hijos" de tipo tinyint y en la misma sentencia una restricción "check" que no
permita valores superiores a 30:

```
 alter table empleados
  add hijos tinyint
  constraint CK_empleados_hijos check (hijos<=30);
```

5- Ingrese algunos registros:

```
 insert into empleados values('22222222','Juan Lopez','Colon
123','Cordoba',2);
 insert into empleados values('23333333','Ana Garcia','Sucre
435','Cordoba',3);
```

6- Intente agregar el campo "sueldo" de tipo decimal(6,2) no nulo y una restricción "check" que no
permita valores negativos para dicho campo:

```
 alter table empleados
  add sueldo decimal(6,2) not null
  constraint CK_empleados_sueldo check (sueldo>=0);
```

No lo permite porque no damos un valor por defecto para dicho campo no nulo y los registros
existentes necesitan cargar un valor.

7- Agregue el campo "sueldo" de tipo decimal(6,2) no nulo, una restricción "check" que no permita
valores negativos para dicho campo y una restricción "default" que almacene el valor "0":

```
 alter table empleados
  add sueldo decimal(6,2) not null
  constraint CK_empleados_sueldo check (sueldo>=0)
  constraint DF_empleados_sueldo default 0;
```

8- Recupere los registros:

```
 select * from empleados;
```

9- Vea la nueva estructura de la tabla:

```
 exec sp_columns empleados;
```

10- Vea las restricciones:

```
 exec sp_helpconstraint empleados;
```

86 - Campos calculados

Primer problema:

Un comercio almacena los datos de los artículos para la venta en una tabla denominada "articulos".
1- Elimine la tabla, si existe y créela nuevamente:

```
 if object_id('articulos') is not null
  drop table articulos;

 create table articulos(
  codigo int identity,
  descripcion varchar(30),
  precio decimal(5,2) not null,
  cantidad smallint not null default 0,
  montototal as precio *cantidad
 );
```
El campo "montototal" es un campo calculado que multiplica el precio de cada artículo por la
cantidad disponible.

2- Intente ingresar un registro con valor para el campo calculado:
```
 insert into articulos values('birome',1.5,100,150);
```
No lo permite.

3- Ingrese algunos registros:
```
 insert into articulos values('birome',1.5,100);
 insert into articulos values('cuaderno 12 hojas',4.8,150);
 insert into articulos values('lapices x 12',5,200);
```

4- Recupere los registros:
```
 select * from articulos;
```

5- Actualice un precio y recupere los registros:
```
 update articulos set precio=2 where descripcion='birome';
 select * from articulos;
```
el campo calculado "montototal" recalcula los valores para cada registro automáticamente.

6- Actualice una cantidad y vea el resultado:
```
 update articulos set cantidad=200 where descripcion='birome';
 select * from articulos;
```
el campo calculado "montototal" recalcula sus valores.

7- Intente actualizar un campo calculado:
```
 update articulos set montototal=300 where descripcion='birome';
```
No lo permite.

87 - Tipo de dato definido por el usuario (crear - informacion)

Primer problema:

Un comercio almacena los datos de sus empleados en una tabla denominada "empleados".
1- Elimine la tabla si existe:

```
 if object_id ('empleados') is not null
  drop table empleados;
```

2- Defina un nuevo tipo de dato llamado "tipo_legajo". Primero debe eliminarlo (si existe) para
volver a crearlo. Para ello emplee esta sentencia que explicaremos en el siguiente capítulo:

```
 if exists (select name from systypes
  where name = 'tipo_legajo')
  exec sp_droptype tipo_legajo;
```

3- Cree un tipo de dato definido por el usuario llamado "tipo_legajo" basado en el tipo "char" de 4
caracteres que no permita valores nulos.

4- Ejecute el procedimiento almacenado "sp_help" junto al nombre del tipo de dato definido
anteriormente para obtener información del mismo.

5- Cree la tabla "empleados" con 3 campos: legajo (tipo_legajo), documento (char de 8) y nombre (30
caracteres):

```
 create table empleados(
  legajo tipo_legajo,
  documento char(8),
  nombre varchar(30)
 );
```

6- Intente ingresar un registro con valores por defecto:

```
 insert into empleados default values;
```

No se puede porque el campo "tipo_legajo" no admite valores nulos y no tiene definido un valor por
defecto.

7- Ingrese un registro con valores válidos.

88 - Tipo de dato definido por el usuario (asociación de reglas)

Primer problema:

Un comercio almacena los datos de sus empleados en una tabla denominada "empleados" y en otra

llamada "clientes" los datos de sus clientes".
1- Elimine ambas tablas, si existen:

```
 if object_id ('empleados') is not null
  drop table empleados;
 if object_id ('clientes') is not null
  drop table clientes;
```

2- Defina un nuevo tipo de dato llamado "tipo_año". Primero debe eliminarlo, si existe, para volver
a crearlo. Para ello emplee esta sentencia que explicaremos en el siguiente capítulo:

```
 if exists (select *from systypes
  where name = 'tipo_año')
   exec sp_droptype tipo_año;
```

3- Cree un tipo de dato definido por el usuario llamado "tipo_año" basado en el tipo "int" que
permita valores nulos:

```
 exec sp_addtype tipo_año, 'int','null';
```

4- Ejecute el procedimiento almacenado "sp_help" junto al nombre del tipo de dato definido
anteriormente para obtener información del mismo:

```
 sp_help tipo_año;
```

5- Cree la tabla "empleados" con 3 campos: documento (char de 8), nombre (30 caracteres) y
añoingreso (tipo_año):

```
 create table empleados(
  documento char(8),
  nombre varchar(30),
  añoingreso tipo_año
 );
```

6- Elimine la regla llamada "RG_año" si existe:

```
 if object_id ('RG_año') is not null
   drop rule RG_año;
```

7- Cree la regla que permita valores integer desde 1990 (año en que se inauguró el comercio) y el
año actual:

```
 create rule RG_año
  as @año between 1990 and datepart(year,getdate());
```

8- Asocie la regla al tipo de datos "tipo_año" especificando que solamente se aplique a los futuros
campos de este tipo:
```
exec sp_bindrule RG_año, 'tipo_año', 'futureonly';
```

9- Vea si se aplicó a la tabla empleados:
```
exec sp_helpconstraint empleados;
```
No se aplicó porque especificamos la opción "futureonly":

10- Cree la tabla "clientes" con 3 campos: nombre (30 caracteres), añoingreso (tipo_año) y domicilio
(30 caracteres):
```
create table clientes(
  documento char(8),
  nombre varchar(30),
  añoingreso tipo_año
);
```

11- Vea si se aplicó la regla en la nueva tabla:
```
exec sp_helpconstraint clientes;
```
Si aparece.

12- Ingrese registros con valores para el año que infrinjan la regla en la tabla "empleados":
```
insert into empleados values('11111111','Ana Acosta',2050);
select * from empleados;
```
Lo acepta porque en esta tabla no se aplica la regla.

13- Intente ingresar en la tabla "clientes" un valor de fecha que infrinja la regla:
```
insert into clientes values('22222222','Juan Perez',2050);
```
No lo permite.

14- Quite la asociación de la regla con el tipo de datos:
```
exec sp_unbindrule 'tipo_año';
```

15- Vea si se quitó la asociación:
```
exec sp_helpconstraint clientes;
```
Si se quitó.

16- Vuelva a asociar la regla, ahora sin el parámetro "futureonly":
```
exec sp_bindrule RG_año, 'tipo_año';
```
Note que hay valores que no cumplen la regla pero SQL Server NO lo verifica al momento de asociar

una regla.

17- Intente agregar una fecha de ingreso fuera del intervalo que admite la regla en cualquiera de
las tablas (ambas tienen la asociación):
```
 insert into empleados values('33333333','Romina Guzman',1900);
```
Mensaje de error.

18- Vea la información del tipo de dato:
```
 exec sp_help tipo_año;
```
En la columna que hace referencia a la regla asociada aparece
"RG_año".

19- Elimine la regla llamada "RG_añonegativo", si existe:
```
 if object_id ('RG_añonegativo') is not null
   drop rule RG_añonegativo;
```

20- Cree una regla llamada "RG_añonegativo" que admita valores entre -2000 y -1:
```
  create rule RG_añonegativo
  as @año between -2000 and -1;
```

21- Asocie la regla "RG_añonegativo" al campo "añoingreso" de la tabla "clientes":
```
 exec sp_bindrule RG_añonegativo, 'clientes.añoingreso';
```

22- Vea si se asoció:
```
 exec sp_helpconstraint clientes;
```
Se asoció.

23- Verifique que no está asociada al tipo de datos "tipo_año":
```
 exec sp_help tipo_año;
```
No, tiene asociada la regla "RG_año".

24- Intente ingresar un registro con valor '-1900' para el campo "añoingreso" de "empleados":
```
 insert into empleados values('44444444','Pedro Perez',-1900);
```
No lo permite por la regla asociada al tipo de dato.

25- Ingrese un registro con valor '-1900' para el campo "añoingreso" de "clientes" y recupere los
registros de dicha tabla:
```
 insert into clientes values('44444444','Pedro Perez',-1900);
 select * from clientes;
```

Note que se ingreso, si bien el tipo de dato de "añoingreso" tiene asociada una regla que no admite
tal valor, el campo tiene asociada una regla que si lo admite y ésta prevalece.

89 - Tipo de dato definido por el usuario (valores predeterminados)

Primer problema:

Un comercio almacena los datos de sus empleados en una tabla denominada "empleados" y en otra
llamada "clientes" los datos de sus clientes".
1- Elimine ambas tablas, si existen:

```
 if object_id ('empleados') is not null
  drop table empleados;
 if object_id ('clientes') is not null
  drop table clientes;
```

2- Defina un nuevo tipo de dato llamado "tipo_año". Primero debe eliminarlo, si existe, para volver
a crearlo. Para ello emplee esta sentencia que explicaremos en el siguiente capítulo:

```
 if exists (select *from systypes
  where name = 'tipo_año')
   exec sp_droptype tipo_año;
```

3- Cree un tipo de dato definido por el usuario llamado "tipo_año" basado en el tipo "int" que
permita valores nulos:

```
 exec sp_addtype tipo_año, 'int','null';
```

4- Ejecute el procedimiento almacenado "sp_help" junto al nombre del tipo de dato definido
anteriormente para obtener información del mismo:

```
 exec sp_help tipo_año;
```

5- Cree la tabla "empleados" con 3 campos: documento (char de 8), nombre (30 caracteres) y
añoingreso (tipo_año):

```
 create table empleados(
  documento char(8),
  nombre varchar(30),
  añoingreso tipo_año
```

```
 );
```

6- Elimine el valor predeterminado "VP_añoactual" si existe:

```
 if object_id ('VP_añoactual') is not null
   drop default VP_añoactual;
```

7- Cree el valor predeterminado "VP_añoactual" que almacene el año actual:

```
 create default VP_añoactual
  as datepart(year,getdate());
```

8- Asocie el valor predeterminado al tipo de datos "tipo_año" especificando que solamente se aplique
a los futuros campos de este tipo:

```
 exec sp_bindefault VP_añoactual, 'tipo_año', 'futureonly';
```

9- Vea si se aplicó a la tabla empleados:

```
 exec sp_helpconstraint empleados;
```

No se aplicó porque especificamos la opción "futureonly":

10- Cree la tabla "clientes" con 3 campos: nombre (30 caracteres), añoingreso (tipo_año) y domicilio
(30 caracteres):

```
 create table clientes(
  documento char(8),
  nombre varchar(30),
  añoingreso tipo_año
 );
```

11- Vea si se aplicó la regla en la nueva tabla:

```
 exec sp_helpconstraint clientes;
```

Si se aplicó.

12- Ingrese un registro con valores por defecto en la tabla "empleados" y vea qué se almacenó en
"añoingreso":

```
 insert into empleados default values;
 select * from empleados;
```

Se almacenó "null" porque en esta tabla no se aplica el valor predeterminado.

13- Ingrese en la tabla "clientes" un registro con valores por defecto y recupere los registros:

```
 insert into clientes default values;
```

```
 select * from clientes;
```
Se almacenó el valor predeterminado.

14- Elimine el valor predeterminado llamado "VP_año2000", si existe:
```
 if object_id ('VP_año2000') is not null
   drop default Vp_año2000;
```

15- Cree un valor predeterminado llamado "VP_año2000" con el valor 2000:
```
 create default VP_año2000
  as 2000;
```

16- Asócielo al tipo de dato definido sin especificar "futureonly":
```
 exec sp_bindefault VP_año2000, 'tipo_año';
```

17- Verifique que se asoció a la tabla "empleados":
```
 exec sp_helpconstraint empleados;
```

18- Verifique que reemplazó al valor predeterminado anterior en la tabla "clientes":
```
 exec sp_helpconstraint clientes;
```

18- Ingrese un registro en ambas tablas con valores por defecto y vea qué se almacenó en el año de
ingreso:
```
 insert into empleados default values;
 select * from empleados;
 insert into clientes default values;
 select * from clientes;
```

19- Vea la información del tipo de dato:
```
 exec sp_help tipo_año;
```
La columna que hace referencia al valor predeterminado asociado muestra "VP_año2000".

20- Intente agregar a la tabla "empleados" una restricción "default":
```
 alter table empleados
 add constraint DF_empleados_año
 default 1990
 for añoingreso;
```
No lo permite porque el tipo de dato del campo ya tiene un valor predeterminado asociado.

21- Quite la asociación del valor predeterminado al tipo de dato:

```
 exec sp_unbindefault 'tipo_año';

22- Agregue a la tabla "empleados" una restricción "default":
 alter table empleados
 add constraint DF_empleados_año
 default 1990
 for añoingreso;

23- Asocie el valor predeterminado "VP_añoactual" al tipo de dato
"tipo_año":
 exec sp_bindefault VP_añoactual, 'tipo_año';

24- Verifique que el tipo de dato tiene asociado el valor predeterminado:
 exec sp_help tipo_año;

25- Verifique que la tabla "clientes" tiene asociado el valor
predeterminado:
 exec sp_helpconstraint clientes;

26- Verifique que la tabla "empleados" no tiene asociado el valor
predeterminado "VP_añoactual"
asociado al tipo de dato y tiene la restricción "default":
 exec p_helpconstraint empleados;
```

90 - Tipo de dato definido por el usuario (eliminar)

Primer problema:

```
Un comercio almacena los datos de sus empleados en una tabla
denominada "empleados".
1- Elimine la tabla si existe:
 if object_id ('empleados') is not null
  drop table empleados;

2- Defina un nuevo tipo de dato llamado "tipo_año". Primero debe
eliminarlo, si existe para volver a
crearlo:
 if exists (select *from systypes
  where name = 'tipo_año')
  exec sp_droptype tipo_año;

3- Cree un tipo de dato definido por el usuario llamado "tipo_año"
basado en el tipo "int" que
permita valores nulos:
 exec sp_addtype tipo_año, 'int','null';
```

4- Elimine la regla llamada "RG_año" si existe:

```
 if object_id ('RG_año') is not null
   drop rule RG_año;
```

5- Cree la regla que permita valores integer desde 1990 (fecha en que se inauguró el comercio) y el
año actual:

```
 create rule RG_año
  as @año between 1990 and datepart(year,getdate());
```

6- Asocie la regla al tipo de datos "tipo_año":

```
 exec sp_bindrule RG_año, 'tipo_año';
```

7- Cree la tabla "empleados" con un campo del tipo creado anteriormente:

```
 create table empleados(
  documento char(8),
  nombre varchar(30),
  añoingreso tipo_año
 );
```

8- Intente ingresar un registro con un valor inválido para el campo "añoingreso":

```
 insert into empleados values('22222222','Juan Lopez',1980);
```

No lo permite.

9- Ingrese un registro con un valor válido para el campo "añoingreso":

```
 insert into empleados values('22222222','Juan Lopez',2000);
```

10- Intente eliminar la regla asociada al tipo de datos:

```
 drop rule RG_año;
```

No se puede porque está asociada a un tipo de datos.

11- Elimine la asociación:

```
 exec sp_unbindrule 'tipo_año';
```

12- Verifique que la asociación ha sido eliminada pero la regla sigue existiendo:

```
 sp_helpconstraint empleados;
 exec sp_help tipo_año;
```

13- Elimine la regla:

```
 drop rule RG_año;
```

14- Verifique que la regla ya no existe:
 exec sp_help RG_año;

15- Ingrese el registro que no pudo ingresar en el punto 8:
 insert into empleados values('22222222','Juan Lopez',1980);
Lo permite porque el tipo de dato ya no tiene asociada la regla.

16- Intente eliminar el tipo de datos "tipo_año":
 exec sp_droptype tipo_año;
No lo permite porque hay una tabla que lo utiliza.

17- Elimine la tabla "empleados":
 drop table empleados;

18- Verifique que el tipo de dato "tipo_año" aún existe:
 exec sp_help tipo_año;

19- Elimine el tipo de datos:
 exec sp_droptype tipo_año;

20- Verifique que el tipo de dato "tipo_año" ya no existe:
 exec sp_help tipo_año;

92 - Subconsultas como expresión

Primer problema:

Un profesor almacena el documento, nombre y la nota final de cada alumno de su clase en una tabla
llamada "alumnos".
1- Elimine la tabla, si existe:
 if object_id('alumnos') is not null
  drop table alumnos;

2- Créela con los campos necesarios. Agregue una restricción "primary key" para el campo "documento"
y una "check" para validar que el campo "nota" se encuentre entre los valores 0 y 10:
 create table alumnos(
  documento char(8),
  nombre varchar(30),
  nota decimal(4,2),
  primary key(documento),

```
  constraint CK_alumnos_nota_valores check (nota>=0 and nota
<=10),
 );

3-Ingrese algunos registros:
 insert into alumnos values('30111111','Ana Algarbe',5.1);
 insert into alumnos values('30222222','Bernardo Bustamante',3.2);
 insert into alumnos values('30333333','Carolina Conte',4.5);
 insert into alumnos values('30444444','Diana Dominguez',9.7);
 insert into alumnos values('30555555','Fabian Fuentes',8.5);
 insert into alumnos values('30666666','Gaston Gonzalez',9.70);

4- Obtenga todos los datos de los alumnos con la nota más alta,
empleando subconsulta.
2 registros.

5- Realice la misma consulta anterior pero intente que la consulta
interna retorne, además del
máximo valor de nota, el nombre.
Mensaje de error, porque la lista de selección de una subconsulta que va
luego de un operador de
comparación puede incluir sólo un campo o expresión (excepto si se
emplea "exists" o "in").

6- Muestre los alumnos que tienen una nota menor al promedio, su
nota, y la diferencia con el
promedio.
3 registros.

7- Cambie la nota del alumno que tiene la menor nota por 4.
1 registro modificado.

8- Elimine los alumnos cuya nota es menor al promedio.
3 registros eliminados.
```

93 - Subconsultas con in

Primer problema:

```
Una empresa tiene registrados sus clientes en una tabla llamada
"clientes", también tiene una tabla
"ciudades" donde registra los nombres de las ciudades.
1- Elimine las tablas "clientes" y "ciudades", si existen:
  if (object_id('ciudades')) is not null
   drop table ciudades;
```

```
  if (object_id('clientes')) is not null
   drop table clientes;
```

2- Cree la tabla "clientes" (codigo, nombre, domicilio, ciudad, codigociudad) y "ciudades" (codigo,
nombre). Agregue una restricción "primary key" para el campo "codigo" de ambas tablas y una "foreing
key" para validar que el campo "codigociudad" exista en "ciudades" con actualización en cascada:

```
 create table ciudades(
  codigo tinyint identity,
  nombre varchar(20),
  primary key (codigo)
 );

 create table clientes (
  codigo int identity,
  nombre varchar(30),
  domicilio varchar(30),
  codigociudad tinyint not null,
  primary key(codigo),
  constraint FK_clientes_ciudad
   foreign key (codigociudad)
   references ciudades(codigo)
   on update cascade,
 );
```

3- Ingrese algunos registros para ambas tablas:

```
 insert into ciudades (nombre) values('Cordoba');
 insert into ciudades (nombre) values('Cruz del Eje');
 insert into ciudades (nombre) values('Carlos Paz');
 insert into ciudades (nombre) values('La Falda');
 insert into ciudades (nombre) values('Villa Maria');

 insert into clientes values ('Lopez Marcos','Colon 111',1);
 insert into clientes values ('Lopez Hector','San Martin 222',1);
 insert into clientes values ('Perez Ana','San Martin 333',2);
 insert into clientes values ('Garcia Juan','Rivadavia 444',3);
 insert into clientes values ('Perez Luis','Sarmiento 555',3);
 insert into clientes values ('Gomez Ines','San Martin 666',4);
 insert into clientes values ('Torres Fabiola','Alem 777',5);
 insert into clientes values ('Garcia Luis','Sucre 888',5);
```

4- Necesitamos conocer los nombres de las ciudades de aquellos clientes cuyo domicilio es en calle "San Martin", empleando subconsulta.
3 registros.

5- Obtenga la misma salida anterior pero empleando join.

6- Obtenga los nombre de las ciudades de los clientes cuyo apellido no comienza con una letra específica, empleando subconsulta. 2 registros.

7- Pruebe la subconsulta del punto 6 separada de la consulta exterior para verificar que retorna una lista de valores de un solo campo.

94 - Subconsultas any - some - all

Primer problema:

Un club dicta clases de distintos deportes a sus socios. El club tiene una tabla llamada
"inscriptos" en la cual almacena el número de "socio", el código del deporte en el cual se inscribe
y la cantidad de cuotas pagas (desde 0 hasta 10 que es el total por todo el año), y una tabla
denominada "socios" en la que guarda los datos personales de cada socio.
1- Elimine las tablas si existen:

```
 if object_id('inscriptos') is not null
  drop table inscriptos;
 if object_id('socios') is not null
  drop table socios;
```

2- Cree las tablas:

```
 create table socios(
  numero int identity,
  documento char(8),
  nombre varchar(30),
  domicilio varchar(30),
  primary key (numero)
 );

 create table inscriptos (
  numerosocio int not null,
  deporte varchar(20) not null,
  cuotas tinyint
  constraint CK_inscriptos_cuotas
   check (cuotas>=0 and cuotas<=10)
```

```
  constraint DF_inscriptos_cuotas default 0,
  primary key(numerosocio,deporte),
  constraint FK_inscriptos_socio
   foreign key (numerosocio)
   references socios(numero)
   on update cascade
   on delete cascade,
 );
```

3- Ingrese algunos registros:

```
 insert into socios values('23333333','Alberto Paredes','Colon 111');
 insert into socios values('24444444','Carlos Conte','Sarmiento 755');
 insert into socios values('25555555','Fabian Fuentes','Caseros 987');
 insert into socios values('26666666','Hector Lopez','Sucre 344');

 insert into inscriptos values(1,'tenis',1);
 insert into inscriptos values(1,'basquet',2);
 insert into inscriptos values(1,'natacion',1);
 insert into inscriptos values(2,'tenis',9);
 insert into inscriptos values(2,'natacion',1);
 insert into inscriptos values(2,'basquet',default);
 insert into inscriptos values(2,'futbol',2);
 insert into inscriptos values(3,'tenis',8);
 insert into inscriptos values(3,'basquet',9);
 insert into inscriptos values(3,'natacion',0);
 insert into inscriptos values(4,'basquet',10);
```

4- Muestre el número de socio, el nombre del socio y el deporte en que está inscripto con un join de ambas tablas.

5- Muestre los socios que se serán compañeros en tenis y también en natación (empleando subconsulta)3 filas devueltas.

6- vea si el socio 1 se ha inscripto en algún deporte en el cual se haya inscripto el socio 2.3 filas.

7- Obtenga el mismo resultado anterior pero empleando join.

8- Muestre los deportes en los cuales el socio 2 pagó más cuotas que ALGUN deporte en los que se inscribió el socio 1.2 registros.

9- Muestre los deportes en los cuales el socio 2 pagó más cuotas que TODOS los deportes en que se inscribió el socio 1.1 registro.

10- Cuando un socio no ha pagado la matrícula de alguno de los deportes en que se ha inscripto, se lo borra de la inscripción de todos los deportes. Elimine todos los socios que no pagaron ninguna cuota en algún deporte.7 registros.

95 - Subconsultas correlacionadas

Primer problema:

Un club dicta clases de distintos deportes a sus socios. El club tiene una tabla llamada "inscriptos" en la cual almacena el número de "socio", el código del deporte en el cual se inscribe
y la cantidad de cuotas pagas (desde 0 hasta 10 que es el total por todo el año), y una tabla
denominada "socios" en la que guarda los datos personales de cada socio.

1- Elimine las tablas si existen:

```
 if object_id('inscriptos') is not null
  drop table inscriptos;
 if object_id('socios') is not null
  drop table socios;
```

2- Cree las tablas:

```
 create table socios(
  numero int identity,
  documento char(8),
  nombre varchar(30),
  domicilio varchar(30),
  primary key (numero)
 );

 create table inscriptos (
  numerosocio int not null,
  deporte varchar(20) not null,
  cuotas tinyint
  constraint CK_inscriptos_cuotas
   check (cuotas>=0 and cuotas<=10)
  constraint DF_inscriptos_cuotas default 0,
  primary key(numerosocio,deporte),
  constraint FK_inscriptos_socio
   foreign key (numerosocio)
   references socios(numero)
   on update cascade
   on delete cascade,
 );
```

3- Ingrese algunos registros:

```
 insert into socios values('23333333','Alberto Paredes','Colon 111');
 insert into socios values('24444444','Carlos Conte','Sarmiento 755');
 insert into socios values('25555555','Fabian Fuentes','Caseros 987');
 insert into socios values('26666666','Hector Lopez','Sucre 344');

 insert into inscriptos values(1,'tenis',1);
```

```
 insert into inscriptos values(1,'basquet',2);
 insert into inscriptos values(1,'natacion',1);
 insert into inscriptos values(2,'tenis',9);
 insert into inscriptos values(2,'natacion',1);
 insert into inscriptos values(2,'basquet',default);
 insert into inscriptos values(2,'futbol',2);
 insert into inscriptos values(3,'tenis',8);
 insert into inscriptos values(3,'basquet',9);
 insert into inscriptos values(3,'natacion',0);
 insert into inscriptos values(4,'basquet',10);
```

4- Se necesita un listado de todos los socios que incluya nombre y domicilio, la cantidad de
deportes a los cuales se ha inscripto, empleando subconsulta.
4 registros.

5- Se necesita el nombre de todos los socios, el total de cuotas que debe pagar (10 por cada
deporte) y el total de cuotas pagas, empleando subconsulta.
4 registros.

6- Obtenga la misma salida anterior empleando join.

96 - Subconsulta - Exists y Not Exists

Primer problema:
Un club dicta clases de distintos deportes a sus socios. El club tiene una tabla llamada "inscriptos" en la cual almacena el número de "socio", el código del deporte en el cual se inscribe
y la cantidad de cuotas pagas (desde 0 hasta 10 que es el total por todo el año), y una tabla
denominada "socios" en la que guarda los datos personales de cada socio.
1- Elimine las tablas si existen:

```
 if object_id('inscriptos') is not null
  drop table inscriptos;
 if object_id('socios') is not null
  drop table socios;
```

2- Cree las tablas:

```
 create table socios(
  numero int identity,
  documento char(8),
  nombre varchar(30),
  domicilio varchar(30),
  primary key (numero)
 );

 create table inscriptos (
  numerosocio int not null,
  deporte varchar(20) not null,
  cuotas tinyint
```

```
  constraint CK_inscriptos_cuotas
   check (cuotas>=0 and cuotas<=10)
  constraint DF_inscriptos_cuotas default 0,
  primary key(numerosocio,deporte),
  constraint FK_inscriptos_socio
   foreign key (numerosocio)
   references socios(numero)
   on update cascade
   on delete cascade,
 );

3- Ingrese algunos registros:
 insert into socios values('23333333','Alberto Paredes','Colon 111');
 insert into socios values('24444444','Carlos Conte','Sarmiento 755');
 insert into socios values('25555555','Fabian Fuentes','Caseros 987');
 insert into socios values('26666666','Hector Lopez','Sucre 344');

 insert into inscriptos values(1,'tenis',1);
 insert into inscriptos values(1,'basquet',2);
 insert into inscriptos values(1,'natacion',1);
 insert into inscriptos values(2,'tenis',9);
 insert into inscriptos values(2,'natacion',1);
 insert into inscriptos values(2,'basquet',default);
 insert into inscriptos values(2,'futbol',2);
 insert into inscriptos values(3,'tenis',8);
 insert into inscriptos values(3,'basquet',9);
 insert into inscriptos values(3,'natacion',0);
 insert into inscriptos values(4,'basquet',10);

4- Emplee una subconsulta con el operador "exists" para devolver la lista de socios que
se
inscribieron en un determinado deporte.
3 registros.

5- Busque los socios que NO se han inscripto en un deporte determinado empleando
"not exists".
1 registro.

6- Muestre todos los datos de los socios que han pagado todas las cuotas.
1 registro.
```

97 - Subconsulta simil autocombinación

Primer problema:

```
Un club dicta clases de distintos deportes a sus socios. El club tiene una tabla llamada
"deportes"
en la cual almacena el nombre del deporte, el nombre del profesor que lo dicta, el día
de la semana
que se dicta y el costo de la cuota mensual.
1- Elimine la tabla si existe:
 if object_id('deportes') is not null
```

```
 drop table deportes;
```

2- Cree la tabla:

```
create table deportes(
 nombre varchar(15),
 profesor varchar(30),
 dia varchar(10),
 cuota decimal(5,2),
);
```

3- Ingrese algunos registros. Incluya profesores que dicten más de un curso:

```
insert into deportes values('tenis','Ana Lopez','lunes',20);
insert into deportes values('natacion','Ana Lopez','martes',15);
insert into deportes values('futbol','Carlos Fuentes','miercoles',10);
insert into deportes values('basquet','Gaston Garcia','jueves',15);
insert into deportes values('padle','Juan Huerta','lunes',15);
insert into deportes values('handball','Juan Huerta','martes',10);
```

4- Muestre los nombres de los profesores que dictan más de un deporte empleando subconsulta.

5- Obtenga el mismo resultado empleando join.

6- Buscamos todos los deportes que se dictan el mismo día que un determinado deporte (natacion)
empleando subconsulta.

7- Obtenga la misma salida empleando "join".

98 - Subconsulta en lugar de una tabla

Primer problema:

Un club dicta clases de distintos deportes. En una tabla llamada "socios" guarda los datos de los
socios, en una tabla llamada "deportes" la información referente a los diferentes deportes que se
dictan y en una tabla denominada "inscriptos", las inscripciones de los socios a los distintos
deportes.
Un socio puede inscribirse en varios deportes el mismo año. Un socio no puede inscribirse en el
mismo deporte el mismo año. Distintos socios se inscriben en un mismo deporte en el mismo año.

1- Elimine las tablas si existen:

```
if object_id('inscriptos') is not null
 drop table inscriptos;
if object_id('socios') is not null
 drop table socios;
if object_id('deportes') is not null
 drop table deportes;
```

2- Cree las tablas con las siguientes estructuras:

```
create table socios(
 documento char(8) not null,
 nombre varchar(30),
 domicilio varchar(30),
 primary key(documento)
);
create table deportes(
 codigo tinyint identity,
 nombre varchar(20),
 profesor varchar(15),
 primary key(codigo)
);
create table inscriptos(
 documento char(8) not null,
 codigodeporte tinyint not null,
 año char(4),
 matricula char(1),--'s'=paga, 'n'=impaga
 primary key(documento,codigodeporte,año),
 constraint FK_inscriptos_socio
  foreign key (documento)
  references socios(documento)
  on update cascade
  on delete cascade
);
```

3- Ingrese algunos registros en las 3 tablas:

```
insert into socios values('22222222','Ana Acosta','Avellaneda 111');
insert into socios values('23333333','Betina Bustos','Bulnes 222');
insert into socios values('24444444','Carlos Castro','Caseros 333');
insert into socios values('25555555','Daniel Duarte','Dinamarca 44');

insert into deportes values('basquet','Juan Juarez');
insert into deportes values('futbol','Pedro Perez');
insert into deportes values('natacion','Marina Morales');
insert into deportes values('tenis','Marina Morales');

insert into inscriptos values ('22222222',3,'2006','s');
insert into inscriptos values ('23333333',3,'2006','s');
insert into inscriptos values ('24444444',3,'2006','n');
insert into inscriptos values ('22222222',3,'2005','s');
insert into inscriptos values ('22222222',3,'2007','n');
insert into inscriptos values ('24444444',1,'2006','s');
insert into inscriptos values ('24444444',2,'2006','s');
```

4- Realice una consulta en la cual muestre todos los datos de las inscripciones, incluyendo el
nombre del deporte y del profesor.
Esta consulta es un join.

5- Utilice el resultado de la consulta anterior como una tabla derivada para emplear en lugar de una

tabla para realizar un "join" y recuperar el nombre del socio, el deporte en el cual está inscripto,
el año, el nombre del profesor y la matrícula.

99 - Subconsulta (update - delete)

Primer problema:

Un club dicta clases de distintos deportes a sus socios. El club tiene una tabla llamada "inscriptos" en la cual almacena el número de "socio", el código del deporte en el cual se inscribe
y si la matricula está o no paga, y una tabla denominada "socios" en la que guarda los datos
personales de cada socio.

1- Elimine las tablas si existen:

```
 if object_id('inscriptos') is not null
  drop table inscriptos;
 if object_id('socios') is not null
  drop table socios;
```

2- Cree las tablas:

```
 create table socios(
  numero int identity,
  documento char(8),
  nombre varchar(30),
  domicilio varchar(30),
  primary key (numero)
 );

 create table inscriptos (
  numerosocio int not null,
  deporte varchar(20) not null,
  matricula char(1),-- 'n' o 's'
  primary key(numerosocio,deporte),
  constraint FK_inscriptos_socio
   foreign key (numerosocio)
   references socios(numero)
 );
```

3- Ingrese algunos registros:

```
 insert into socios values('23333333','Alberto Paredes','Colon 111');
 insert into socios values('24444444','Carlos Conte','Sarmiento 755');
 insert into socios values('25555555','Fabian Fuentes','Caseros 987');
 insert into socios values('26666666','Hector Lopez','Sucre 344');

 insert into inscriptos values(1,'tenis','s');
 insert into inscriptos values(1,'basquet','s');
 insert into inscriptos values(1,'natacion','s');
 insert into inscriptos values(2,'tenis','s');
 insert into inscriptos values(2,'natacion','s');
 insert into inscriptos values(2,'basquet','n');
 insert into inscriptos values(2,'futbol','n');
```

```
 insert into inscriptos values(3,'tenis','s');
 insert into inscriptos values(3,'basquet','s');
 insert into inscriptos values(3,'natacion','n');
 insert into inscriptos values(4,'basquet','n');
```

4- Actualizamos la cuota ('s') de todas las inscripciones de un socio determinado (por documento)
empleando subconsulta.

5- Elimine todas las inscripciones de los socios que deben alguna matrícula (5 registros eliminados)

100 - Subconsulta (insert)

Primer problema:

Un comercio que vende artículos de librería y papelería almacena la información de sus ventas en una
tabla llamada "facturas" y otra "clientes".
1- Elimine las tablas si existen:

```
 if object_id('facturas') is not null
  drop table facturas;
 if object_id('clientes') is not null
  drop table clientes;
```

2-Créelas:

```
 create table clientes(
  codigo int identity,
  nombre varchar(30),
  domicilio varchar(30),
  primary key(codigo)
 );

 create table facturas(
  numero int not null,
  fecha datetime,
  codigocliente int not null,
  total decimal(6,2),
  primary key(numero),
  constraint FK_facturas_cliente
   foreign key (codigocliente)
   references clientes(codigo)
   on update cascade
 );
```

3-Ingrese algunos registros:

```
 insert into clientes values('Juan Lopez','Colon 123');
 insert into clientes values('Luis Torres','Sucre 987');
 insert into clientes values('Ana Garcia','Sarmiento 576');
 insert into clientes values('Susana Molina','San Martin 555');

 insert into facturas values(1200,'2007-01-15',1,300);
```

```
 insert into facturas values(1201,'2007-01-15',2,550);
 insert into facturas values(1202,'2007-01-15',3,150);
 insert into facturas values(1300,'2007-01-20',1,350);
 insert into facturas values(1310,'2007-01-22',3,100);
```

4- El comercio necesita una tabla llamada "clientespref" en la cual quiere almacenar el nombre y
domicilio de aquellos clientes que han comprado hasta el momento más de 500 pesos en mercaderías.
Elimine la tabla si existe y créela con esos 2 campos:

```
 if object_id ('clientespref') is not null
  drop table clientespref;
 create table clientespref(
  nombre varchar(30),
  domicilio varchar(30)
 );
```

5- Ingrese los registros en la tabla "clientespref" seleccionando registros de la tabla "clientes" y
"facturas".

6- Vea los registros de "clientespref":

```
 select * from clientespref;
```