

Repaso básico.

Actividad 6.1



Actividad:

Ejercicio de Consultas básicas.

Vamos a trabajar con una Base de Datos de un Jardín Botánico, que denominaremos JardinBotanicoBasicas.

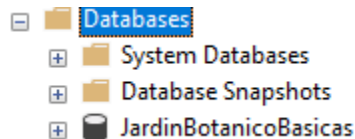
La estructura de la Base de Datos consta de 2 tablas.

- Planta: contiene las plantas del Jardín Botánico
- Familia: contiene las Familias Botánicas a las que pueden pertenecer las plantas.

Se suministra archivo sql de creación de tablas y carga de datos. Se trata de resolver los siguientes supuestos en SQL.

1.- Crear la Base de Datos JardinBotanicoBasicas.

```
--Crear base de datos
create database JardinBotanicoBasicas
--go
--http://www.jardinbotanico.org/parque-jardin-botanico-de-moraleja-de-enmedio/familias-de-plantas/
use JardinBotanicoBasicas
go
```



Creamos la base de datos y comprobamos que está creada haciendo *refresh*. Le decimos al programa que queremos usarla para seguir trabajando a continuación en esta base de datos.

```
create database JardinBotanicoBasicas
```

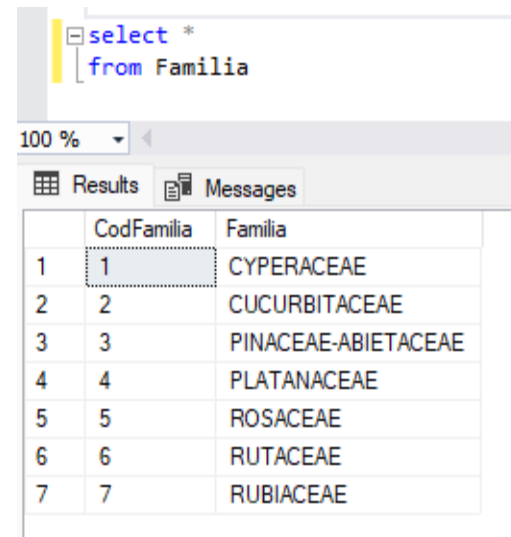
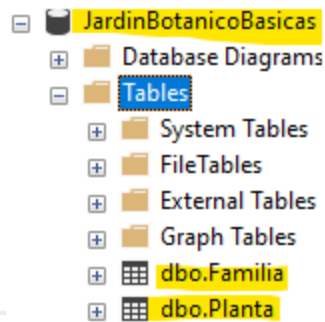
```
use JardinBotanicoBasicas
```

2.- Con el archivo suministrado crear las tablas Planta y Familia y cargar los datos.

```
if object_id('Planta') is not null
    drop table Planta;
go
if object_id('Familia') is not null
    drop table Familia;
go
```

```
--Crear tabla Familia
create table Familia
(
    CodFamilia integer,
    Familia varchar(50)
);

--Crear tabla Planta
create table Planta
(
    CodPlanta integer,
    DescripcionPlanta varchar(50),
    CodFamilia integer,
    Precio decimal(6,2)
);
```



```
select *
from Familia
```

	CodFamilia	Familia
1	1	CYPERACEAE
2	2	CUCURBITACEAE
3	3	PINACEAE-ABIETACEAE
4	4	PLATANACEAE
5	5	ROSACEAE
6	6	RUTACEAE
7	7	RUBIACEAE

Creamos las tablas suministradas por el ejercicio, luego comprobamos que están creadas haciendo *refresh* en la base de datos. Posteriormente procedemos a cargar los datos y comprobamos que están correctamente cargados.

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

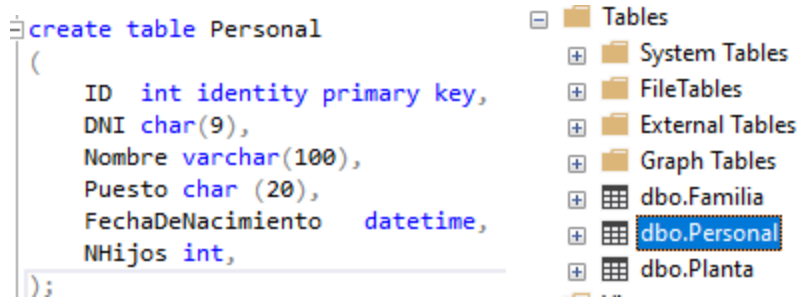
```
if object_id('Familia') is not null
drop table Familia;
go
```

```
--Crear tabla Familia
create table Familia
(
    CodFamilia integer,
    Familia varchar(50)
);
```

```
--Crear tabla Planta
create table Planta
(
    CodPlanta integer,
    DescripcionPlanta varchar(50),
    CodFamilia integer,
    Precio decimal(6,2)
);
```

3.- Crear la tabla Personal. Esta tabla contendrá los siguientes elementos:

ID	entero autoincrementable y clave primaria
DNI	cadena de caracteres de 9 caracteres
Nombre	cadena de caracteres de longitud variable de 100 caracteres
Puesto	cadena de caracteres de longitud fija de 20 caracteres
FechaDeNacimiento	Fecha/Hora NHijos entero



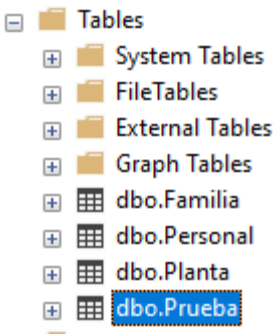
Creamos la tabla con los campos que nos pide el ejercicio y luego comprobamos con un *refresh* que está correctamente creada en la base de datos.

```
create table Personal
(
    ID int identity primary key,
    DNI char(9),
    Nombre varchar(100),
    Puesto char (20),
    FechaDeNacimiento datetime,
    NHijos int,
);
```

4.- Crear la tabla Prueba. Esta tabla contendrá los siguientes elementos:

Id	entero
Dato	cadena de caracteres de longitud fija de 20 caracteres

```
create table Prueba
(
    ID int,
    Dato char(20),
);
```



Al igual que en el ejercicio anterior, creamos la tabla con los campos correspondientes y comprobamos que está creada.

```
create table Prueba
(
    ID int,
    Dato char(20),
);
```

5.- Insertar los siguientes datos en la tabla prueba: 1,'elemento1' 2,'elemento2'.

```
insert Prueba
(ID,Dato)
values(1,'elemento1')

insert Prueba
(ID,Dato)
values (2, 'elemento2')
go
```

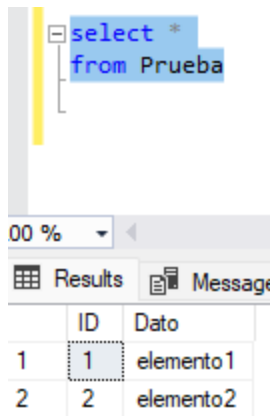
Insertamos los datos en la tabla correspondiente.

```
insert Prueba
(ID,Dato)
values(1,'elemento1')

insert Prueba
(ID,Dato)
values (2, 'elemento2')
```

Alejandro Afonso Barber

6.- Ver el contenido de la tabla prueba.



```
select *  
from Prueba
```

00 %

Results Message

	ID	Dato
1	1	elemento1
2	2	elemento2

Comprobamos que los datos creados en el ejercicio anterior están correctamente creados.

```
select *  
from Prueba
```

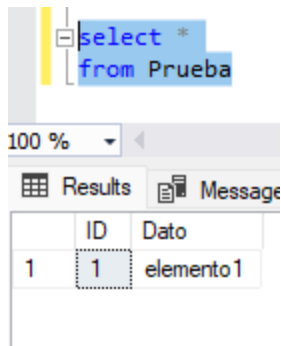
7.- Borrar el elemento de id=2 de la tabla prueba.

```
delete from Prueba  
where ID='2'
```

Borramos el elemento que nos pide el ejercicio.

```
delete from Prueba  
where ID='2'
```

8.- Ver el contenido de la tabla prueba.



100 %

ID	Dato
1	elemento1

Comprobamos que el elemento que hemos borrado efectivamente ya no está en la tabla.

```
select *  
from Prueba
```

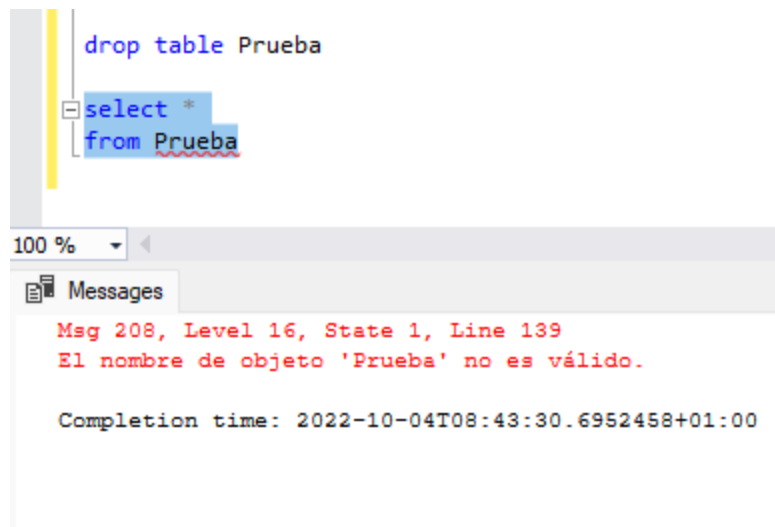
9.- Borrar la tabla prueba.



Borramos la tabla y al actualizar vemos que ya no aparece a la izquierda.

```
drop table Prueba
```

10.- Ver el contenido de la tabla prueba.



The screenshot shows a SQL query editor with the following code:

```
drop table Prueba  
  
select *  
from Prueba
```

Below the query editor, the 'Messages' pane displays an error:

```
Msg 208, Level 16, State 1, Line 139  
El nombre de objeto 'Prueba' no es válido.  
  
Completion time: 2022-10-04T08:43:30.6952458+01:00
```

Al intentar ver el contenido de la tabla nos salta un mensaje de error confirmando que la tabla “Prueba” no existe ya que la hemos borrado en el apartado anterior.

```
select *  
from Prueba
```

11.- Insertar los siguientes datos en la tabla personal:

```
set identity_insert Personal on;  
go  
  
insert into Personal  
    (ID, DNI, Nombre, Puesto, FechaDeNacimiento)  
values (33, '65656546G', 'Antonio', 'Jardinero', '23/05/1978');  
go
```

Como antes habíamos usado *identity* ahora lo hemos desactivado para poder poner el ID de forma manual.

```
set identity_insert Personal on;
```

```
insert into Personal  
    (ID, DNI, Nombre, Puesto, FechaDeNacimiento)  
values (33, '65656546G', 'Antonio', 'Jardinero', '23/05/1978');
```



```
select *
from Personal
```

100 %

Results Messages

	ID	DNI	Nombre	Puesto	FechaDeNacimiento	NHijos
1	1	32456789H	Maria	Jefa	1975-03-27 00:00:00.000	NULL
2	2	23456789W	Juan	Técnico	1968-04-23 00:00:00.000	NULL
3	3	45454545J	Ana	Jardinero	1980-01-21 00:00:00.000	NULL
4	33	65656546G	Antonio	Jardinero	1978-05-23 00:00:00.000	NULL

Comprobamos que los datos han sido creados correctamente.

```
select *
from Personal
```

12.- Borrar el registro con DNI 65656546G.

```
delete from Personal
where DNI='65656546G';
go
```

```
select *
from Personal
```

100 %

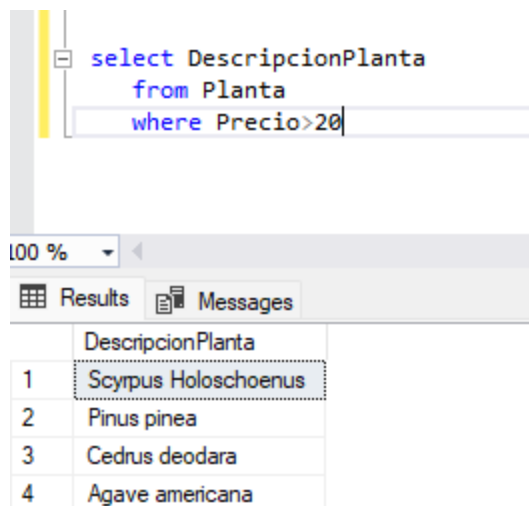
Results Messages

	ID	DNI	Nombre	Puesto	FechaDeNacimiento	NHijos
1	1	32456789H	Maria	Jefa	1975-03-27 00:00:00.000	NULL
2	2	23456789W	Juan	Técnico	1968-04-23 00:00:00.000	NULL
3	3	45454545J	Ana	Jardinero	1980-01-21 00:00:00.000	NULL

Borramos el registro que nos pide y comprobamos con un *select all* que efectivamente el registro ya no está en la tabla.

```
delete from Personal
where DNI='65656546G';
go
```

13.- Mostrar las descripciones de las plantas de precio superior a 20.



The screenshot shows a SQL query window with the following text:

```
select DescripcionPlanta
from Planta
where Precio>20
```

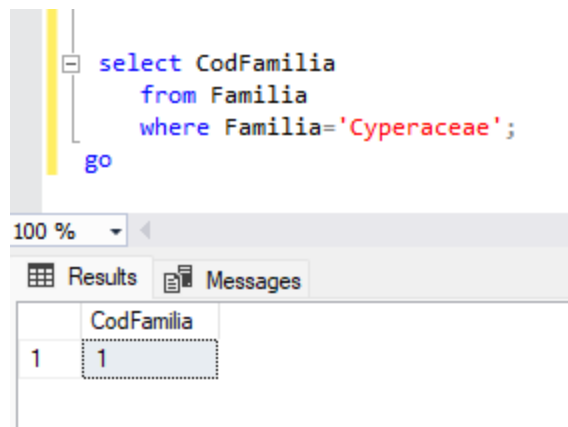
Below the query window, the 'Results' tab is active, displaying a table with the following data:

	DescripcionPlanta
1	Scyrpus Holoschoenus
2	Pinus pinea
3	Cedrus deodara
4	Agave americana

Mostramos las descripciones de las plantas que están dentro de la tabla “Planta” cuyo precio es superior a 20.

```
select DescripcionPlanta
from Planta
where Precio>20
```

14.- Mostrar el código de la familia Cyperaceae.



The screenshot shows a SQL query window with the following text:

```
select CodFamilia
from Familia
where Familia='Cyperaceae';
go
```

Below the query window, the 'Results' tab is active, displaying a table with the following data:

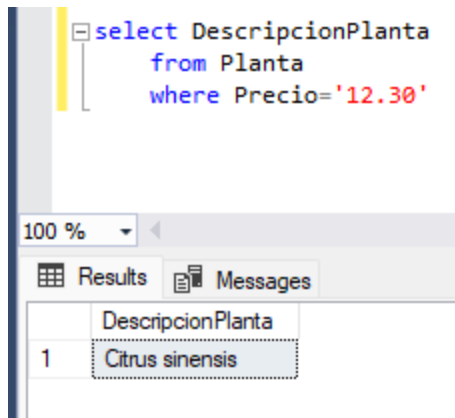
	CodFamilia
1	1

Aquí mostramos el Código de Familia de la tabla “Familia” que coincide con el nombre de Familia de “Cyperaceae”.

```
select CodFamilia
from Familia
where Familia='Cyperaceae';
```

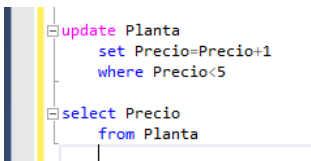
Alejandro Afonso Barber

15.- Mostrar la descripción de las plantas con precio 12 con treinta céntimos. Comprobarlo.



```
select DescripcionPlanta
      from Planta
      where Precio='12.30'
```

16.- Actualizar el precio de las plantas de precio menor que 5, incrementándolas en un euro. Comprobarlo.



Se me olvidó hacer captura del precio de las plantas antes de la está la captura del código y posterior al cambio.

The screenshot shows the 'Results' tab with a table containing 17 rows of prices. The first row is highlighted.

	Precio
1	23.00
2	14.00
3	5.00
4	12.00
5	45.00
6	23.30
7	10.40
8	12.50
9	14.30
10	2.05
11	3.30
12	6.20
13	20.30
14	4.20
15	10.20
16	12.30
17	17.35

```
update Planta
      set Precio=Precio+1
      where Precio<5
```

```
select Precio
      from Planta
```

17.- Cambiar el nombre de la familia de código 5 a Rosaceae officinalis. Comprobarlo.

```
select *  
from Familia
```

100 %

Results Messages

	CodFamilia	Familia
1	1	CYPERACEAE
2	2	CUCURBITACEAE
3	3	PINACEAE-ABIETACEAE
4	4	PLATANACEAE
5	5	ROSACEAE
6	6	RUTACEAE
7	7	RUBIACEAE

Comprobamos antes la tabla de Familia.

```
update Familia  
set Familia='Rosaceae officinalis'  
where CodFamilia=5  
go  
  
select *  
from Familia
```

100 %

Results Messages

	CodFamilia	Familia
1	1	CYPERACEAE
2	2	CUCURBITACEAE
3	3	PINACEAE-ABIETACEAE
4	4	PLATANACEAE
5	5	Rosaceae officinalis
6	6	RUTACEAE
7	7	RUBIACEAE

Volvemos a comprobar después que efectivamente se ha realizado el cambio en el Código de Familia 5.

```
update Familia  
set Familia='Rosaceae officinalis'  
where CodFamilia=5  
go  
  
select *  
from Familia
```