

SQL Introductorio, SQL SERVER 2008

Manual de estudio
MATI. Marianella Solano Orias

2011

Tabla de contenido

Introducción	5
Introducción a Bases de Datos	5
Elementos de bases de datos.....	5
Diferencia entre SQL y Microsoft SQL Server.....	6
Uso de la interfaz de Microsoft SQL Server Management Studio.....	6
Partes de Microsoft SQL Server Management Studio.....	7
Explorador de objetos	7
Nueva Consulta	15
Otros botones.....	16
Componentes del SQL.....	16
Comandos y cláusulas	17
Sentencias DDL.....	17
Sentencias DML.....	22
Práctica #1 Crear objetos	30
Operadores Lógicos.....	34
Operadores de Comparación	35
Funciones de agregado	37
Consultas de Selección	40
Consultas Básicas	40
Ordenar registros	41
Consultas con predicado	43
Alias	47
Bases de datos externas.....	48
Criterios de selección	50
Operadores lógicos	50
Intervalos de valores	54
Operador Like	55
Operador In	57
Cláusula Where	60
Práctica # 2 Creación de las tablas	61

Agrupamiento de registros y funciones agregadas.....	64
Cláusula GROUP BY	64
AVG Media Aritmética.....	66
COUNT Contar Registros	67
MAX, MIN Máximo y mínimo	69
Stdv, stdvp Desviación estándar	70
Sum Suma de valores	72
Var VarP Varianza.....	72
Funciones de Manipulación de fecha.....	74
DATEDIFF	74
DATEPART.....	75
GETDATE.....	76
Porciones de fecha y hora usadas para las funciones de fecha	79
Tipos de datos	79
Tipo de datos Caracter	79
Tipo de datos numérico	80
Tipos de dato de fecha y hora.....	80
Conversión de tipos de datos.....	81
Conversión de datos implícita o automática.....	81
Conversión de datos explícita	82
Práctica #3 Aplicación de conceptos.....	84
Creación de tablas.....	85
Ingresar datos en una tabla.....	93
Tipos de Consultas.....	94
Consultas de Actualización.....	94
Actualización simple.....	94
Actualización compuesta	97
Consultas de Eliminación	107
Consulta de datos añadidos	111
SELECT INTO	114
Práctica # 4 Consultas	117

La cláusula CONSTRAINT	119
Creación de Índices	120
Eliminar y Añadir campos e índices.....	123
Creación de una base de datos	132
Sub Consultas	136
Consultas de referencias cruzadas	139
Consultas de Unión Internas	140
Consultas de Unión Externa	147
Práctica# 5 para la casa	159
Práctica # 6 Práctica General para examen.....	162
Anexo I Solución de Práctica 1	164
Anexo II Solución de Práctica 2	165
Anexo III Solución Práctica 3	170
Anexo IV Solución Práctica 4.....	171
Anexo V Solución Práctica 5.....	173
Anexo VI Solución Práctica 6	177
Bibliografía	182

Introducción

SQL son tres siglas que significan Structured Query Language (Lenguaje Estructurado de Consultas). SQL es un lenguaje informático que se puede utilizar para interaccionar con una base de datos relacional, es una herramienta para organizar, gestionar y recuperar datos almacenados en una base de datos, mediante el uso de sentencias que deben escribirse de acuerdo con unas reglas sintácticas y semánticas de este lenguaje.

Introducción a Bases de Datos

Una base de datos o banco de datos (BD) es un conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso, la mayoría de las bases de datos están en formato digital (electrónico).

Existen programas denominados sistemas gestores de bases de datos (**SGBD**), que permiten almacenar y posteriormente acceder a los datos de forma rápida y estructurada. Las propiedades de estos SGBD, así como su utilización y administración, se estudian dentro del ámbito de la informática.

Elementos de bases de datos

Los elementos básicos de una base de datos son:

Entidades: Persona, lugar, objeto u evento para el cual se obtiene y mantienen datos. Ejemplo: Estudiante, Curso.

Campos: Atributos o características de la entidad.

Ejemplo:

Entidad: Estudiante

Campos: Nombre, Apellido, Edad.

Registros (Records o filas): Grupo de campos que describen un miembro de una entidad.

Archivos: Grupo de registros que contienen datos sobre una entidad.

Llaves (keys): Campo o combinación de campos que permite localizar, acceder o identificar un registro en específico.

Diferencia entre SQL y Microsoft SQL Server

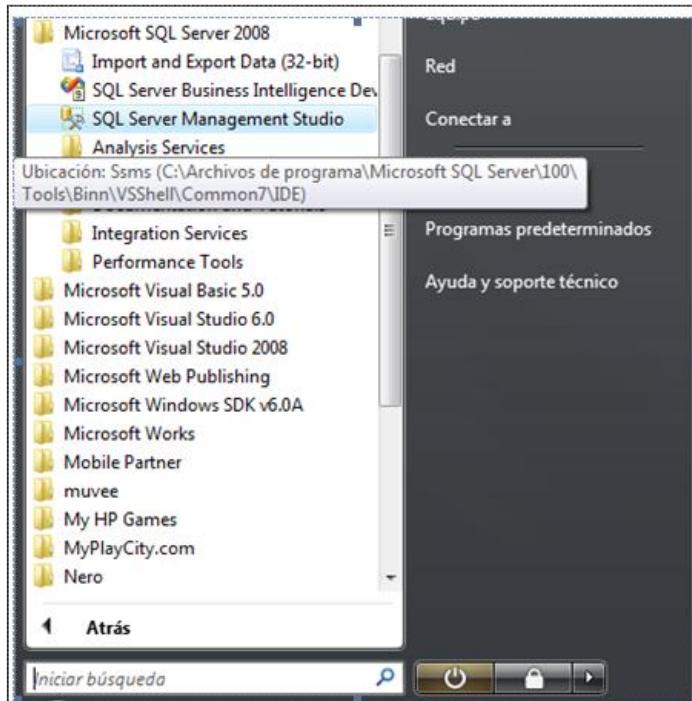
SQL, como se vio anteriormente es un lenguaje informático que se puede utilizar para interactuar con una base de datos y más concretamente en una base de datos relacional.

Microsoft SQL Server es un sistema para la gestión de bases de datos (**SGBD**) producido por Microsoft basado en el modelo relacional. Sus lenguajes para consultas son T-SQL y ANSI SQL.

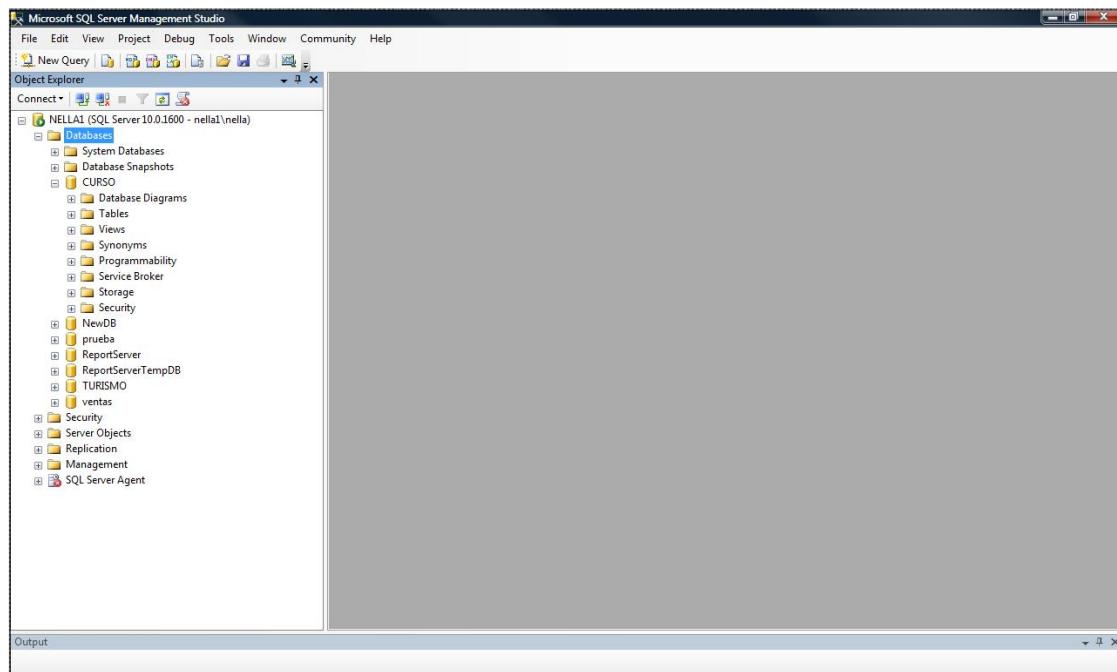
Los sistemas de gestión de bases de datos (en inglés database management system, abreviado DBMS) sirven de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan.

Uso de la interfaz de Microsoft SQL Server Management Studio.

Para ingresar a la interfaz de Microsoft Server Management Studio, se ingresa al menú Inicio, Programas o Todos los programas (dependiendo la versión de Windows que tenga instalado el computador), Microsoft SQL Server 2008, SQL Sever Management Studio; como se muestra en la siguiente imagen.

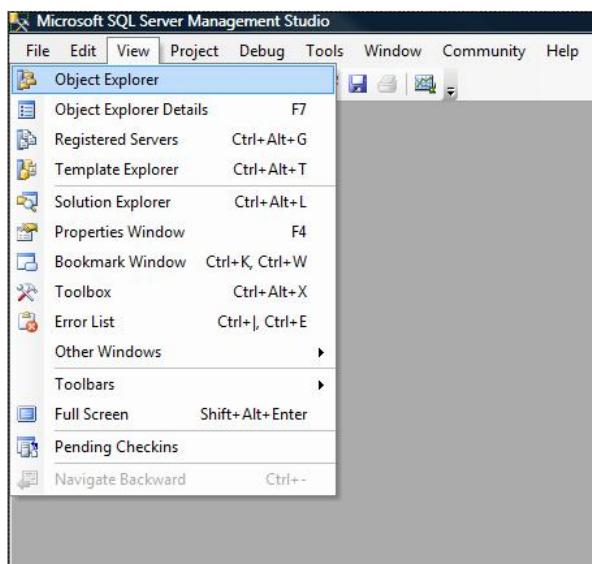


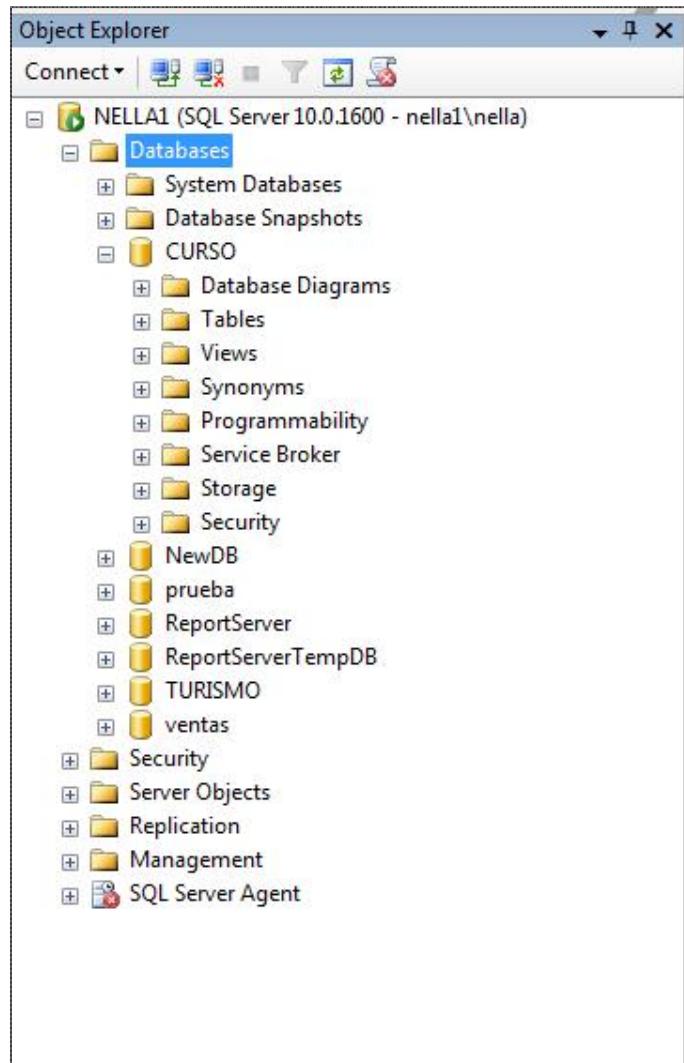
Partes de Microsoft SQL Server Management Studio



Explorador de objetos

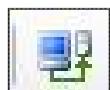
El Explorador de objetos es visible en forma predeterminada en Management Studio, si por alguna razón no se muestra se puede abrir ingresando al menú **Ver (View)**, Explorador de Objetos





Conectarse y desconectarse de un servidor

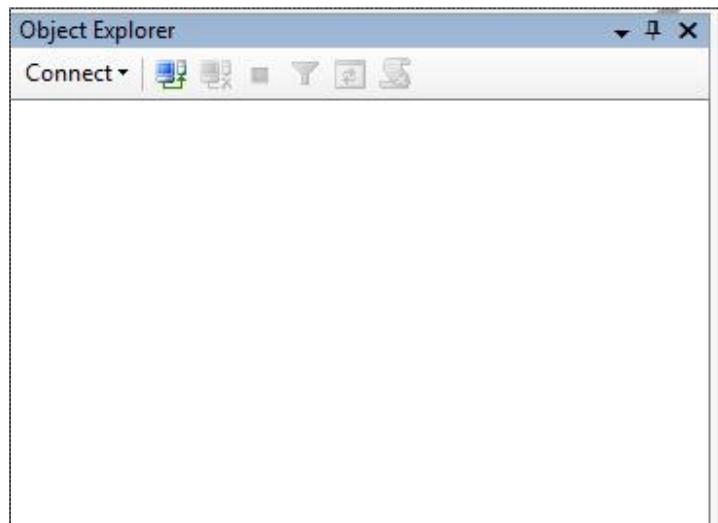
En la barra de herramientas se encuentran los siguientes botones



Permite conectarse a un servidor, al dar clic sobre él, muestra la siguiente pantalla donde debe incluirse al menos el nombre del servidor y la información de autenticación.

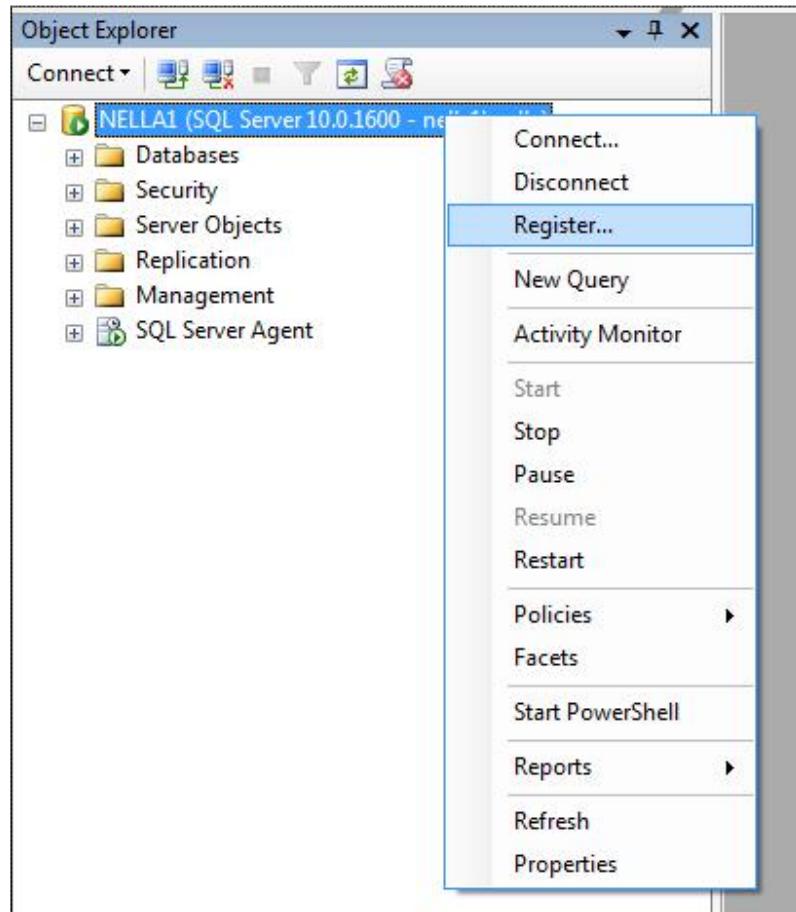


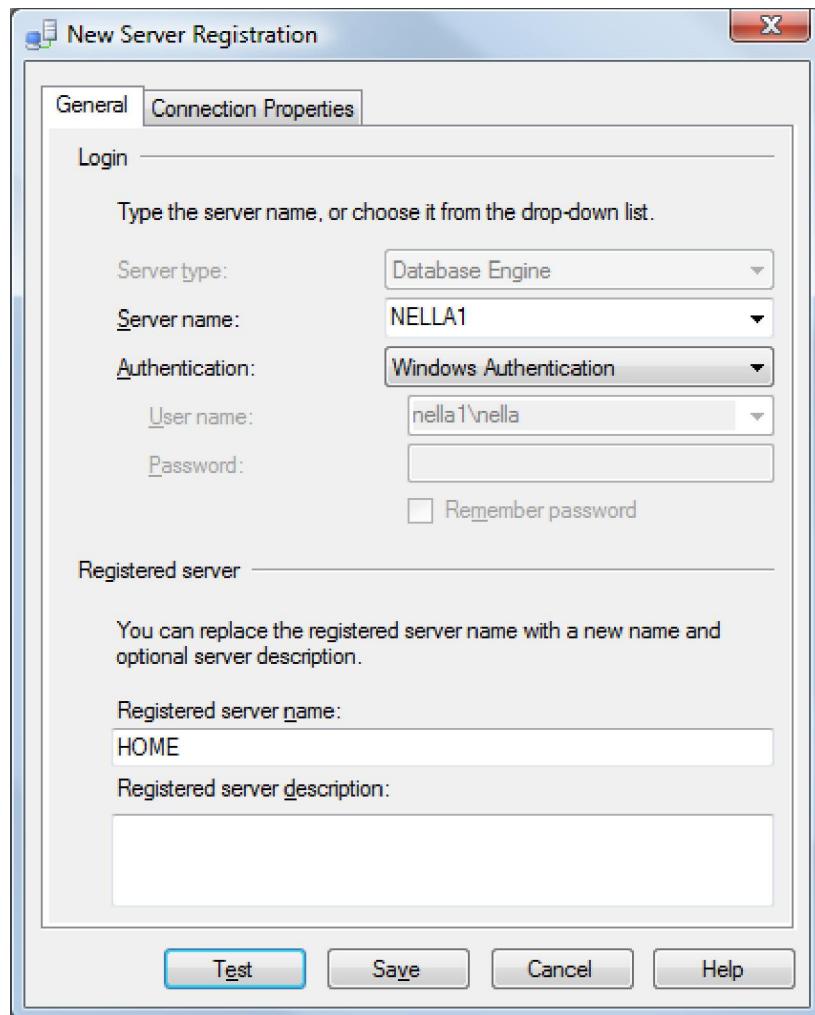
Permite desconectar un servidor, al dar clic sobre él se desconecta el servidor.



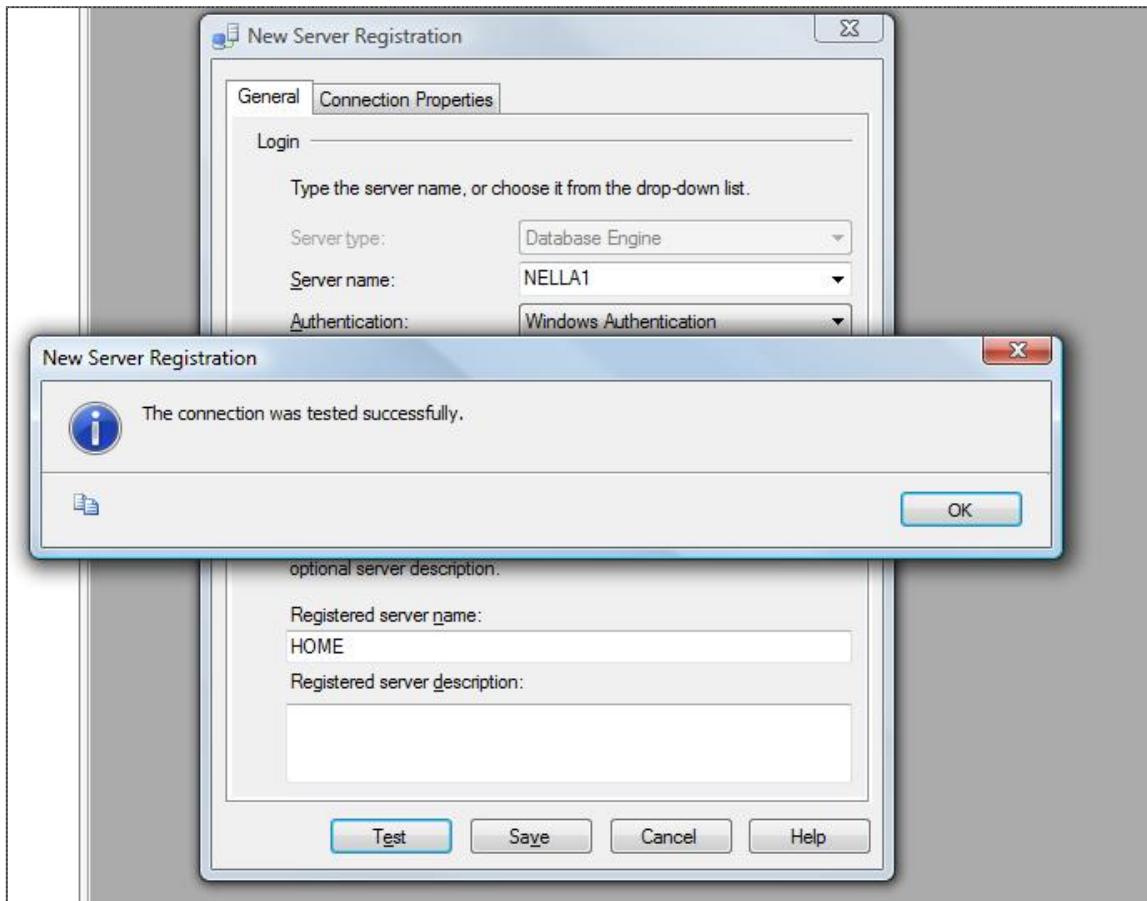
Registrar un servidor del explorador de objetos

Una vez que el servidor se encuentra conectado al dar clic derecho sobre el nombre del servidor se despliega un menú, el cual muestra la opción de **Registrar (Register)**, al seleccionar esta opción se muestra un cuadro de dialogo en donde ese permite especificar en qué lugar del árbol del grupo de servidores se quiere que se muestre el servidor y cambiar el nombre.





Por medio del botón **Test**, se puede verificar si los cambios no afectan la conexión del servidor antes de **Salvar (Save)**



Una vez verificada la conexión, se salvan los cambios.

Ver objetos en el explorador de objetos

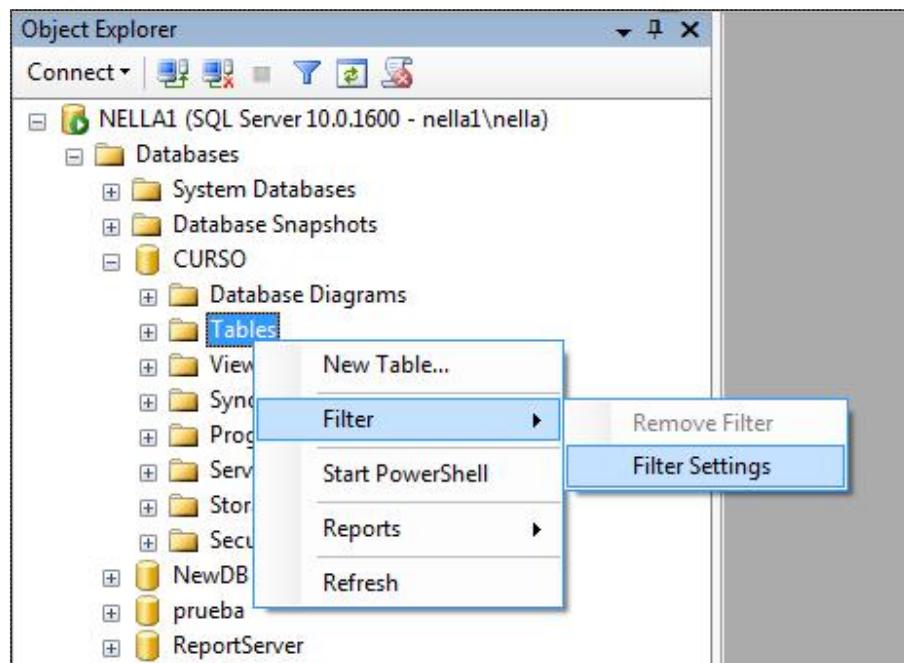
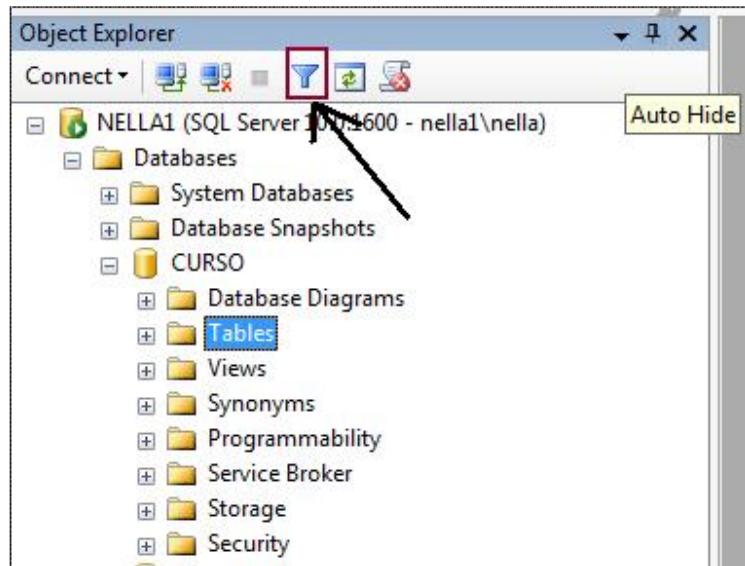
En el Explorador de objetos la información se muestra en una estructura de árbol agrupada por carpetas, para expandir cada carpeta se da clic sobre el signo de mas (+) que se encuentra al lado izquierdo de cada carpeta o doble clic sobre la carpeta.

Para actualizar el contenido de una carpeta, se da clic derecho sobre la carpeta y clic sobre la opción **Actualizar (Refresh)** o clic sobre el botón .

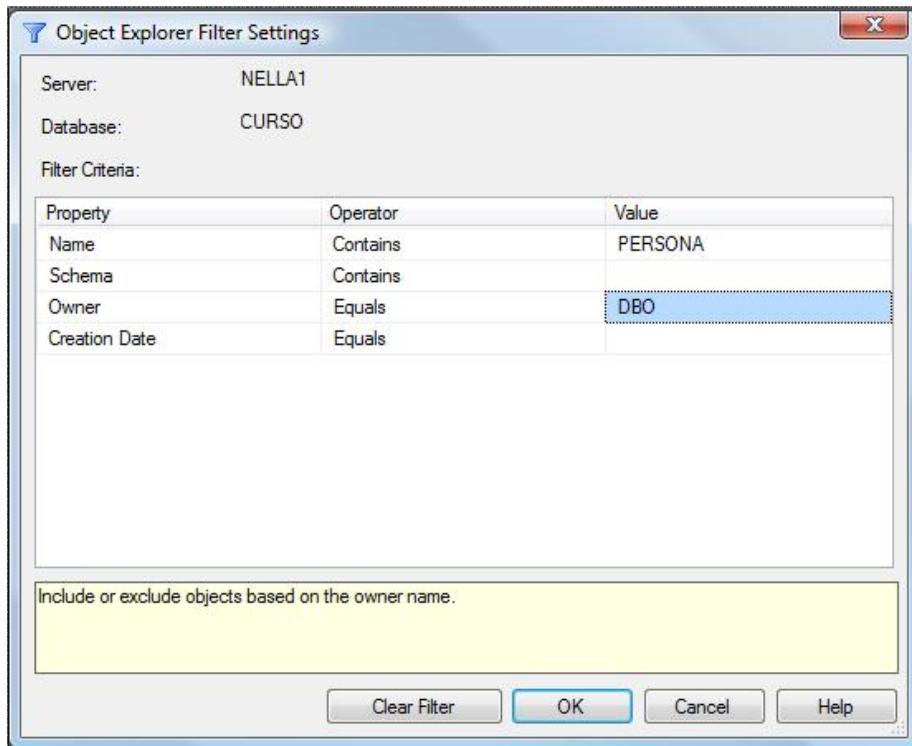
Filtrar la lista de objetos en el Explorador de objetos

En el momento que se necesite encontrar un objeto específico perteneciente a una carpeta, se

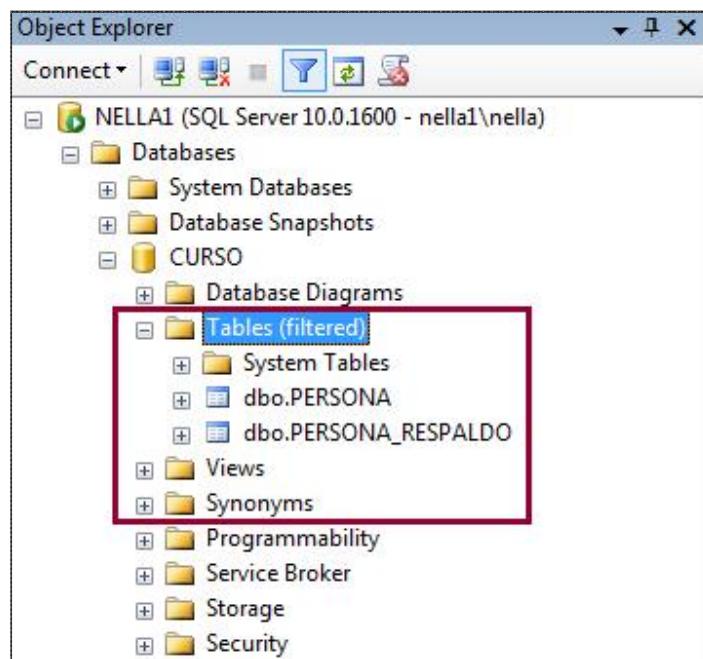
marca la carpeta y se da clic sobre el botón de filtro  o clic derecho sobre la carpeta y se da clic sobre la opción de **Filtro (Filter)**, **Configuración del Filtro (Filter Settings)**.



Se muestra la siguiente ventana de dialogo



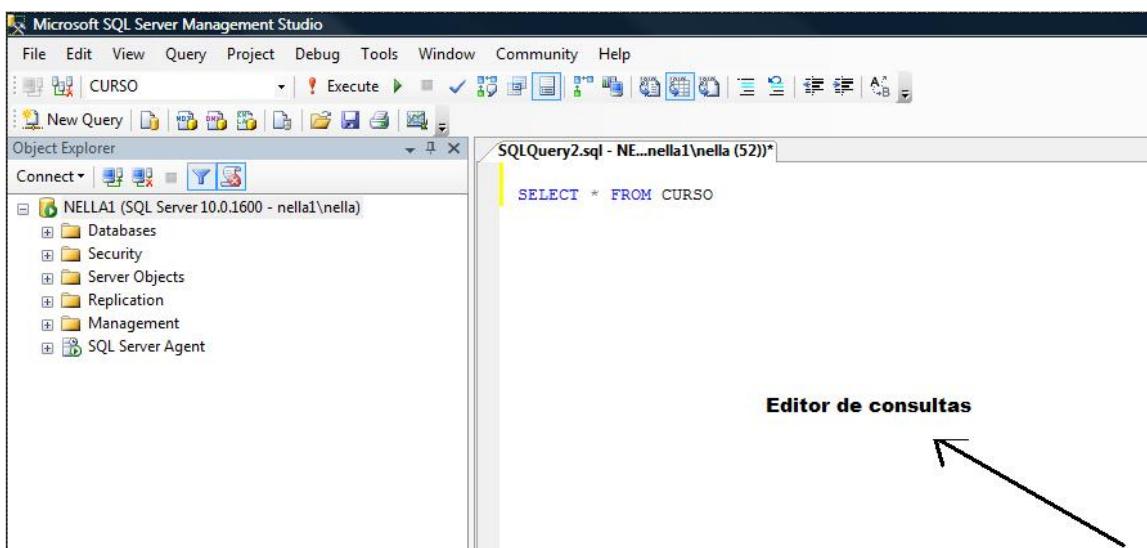
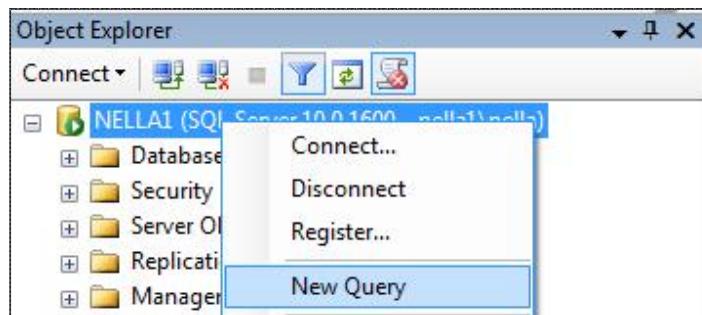
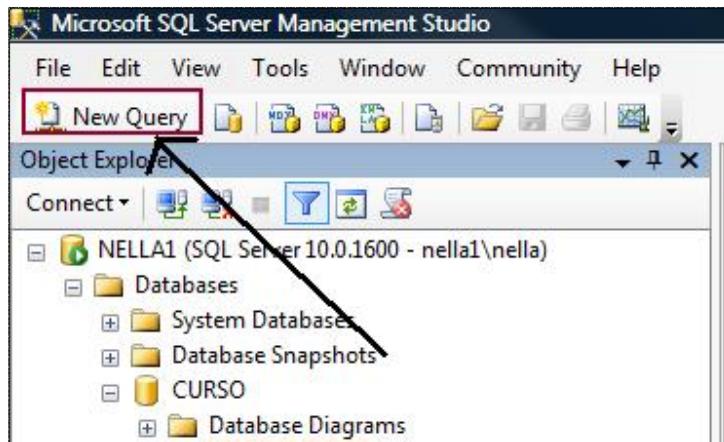
En este caso se buscara el objeto que el nombre sea persona y el dueño dbo, se da clic en **OK**.



Para quitar el filtro se da clic nuevamente en el botón de filtrar y clic sobre **Limpiar Filtro (Clear Filter)**.

Nueva Consulta

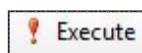
Una vez que el servidor se encuentra conectado , al dar clic sobre el botón **Nueva Consulta (New Query)**, o dando clic derecho sobre el servidor , **Nueva Consulta(New Query)**, muestra la ventana del editor de código.



Otros botones



Permite debuggar el código o query para saber si es correcto.



Permite la ejecución del query.



Permite abrir un query existente.



Guarda el query.



Permite seleccionar la base de datos sobre la cual se va a trabajar.

Componentes del SQL

Los comandos de SQL se dividen en tres categorías:

Data Definition Language (DDL): Lenguaje de Definición de datos o esquemas de estructuras, comandos de creación, modificación de tablas , bases de datos , constrains, índices y otros objetos de base de datos , los comandos son: CREATE, ALTER y DROP.

Data Manipulation Language (DML): Manejo de datos, incluye las operaciones como insertar nuevos registros, modificarlos, borrarlos y realizar consultas por cualquier criterio. Se utilizan los comandos: SELECT, INSERT, UPDATE y DELETE.

Data Control Language (DCL): Lenguaje de Control de Datos, son los comandos que se refieren a la seguridad, como por ejemplo permiso de acceso a determinadas tablas por parte de determinados usuarios; políticas de seguridad y privacidad de los datos; organización de grupos de usuario, etc. Aquí se utilizan los siguientes comandos: GRANT, REVOKE y DENY.

Comandos y cláusulas

Sentencias DDL

CREATE

Permite la creación de objetos tales como: base de datos, tablas, procedimientos almacenados , disparados entre otros.

Cláusula

```
CREATE nombreObjeto
```

Creación de una base de datos

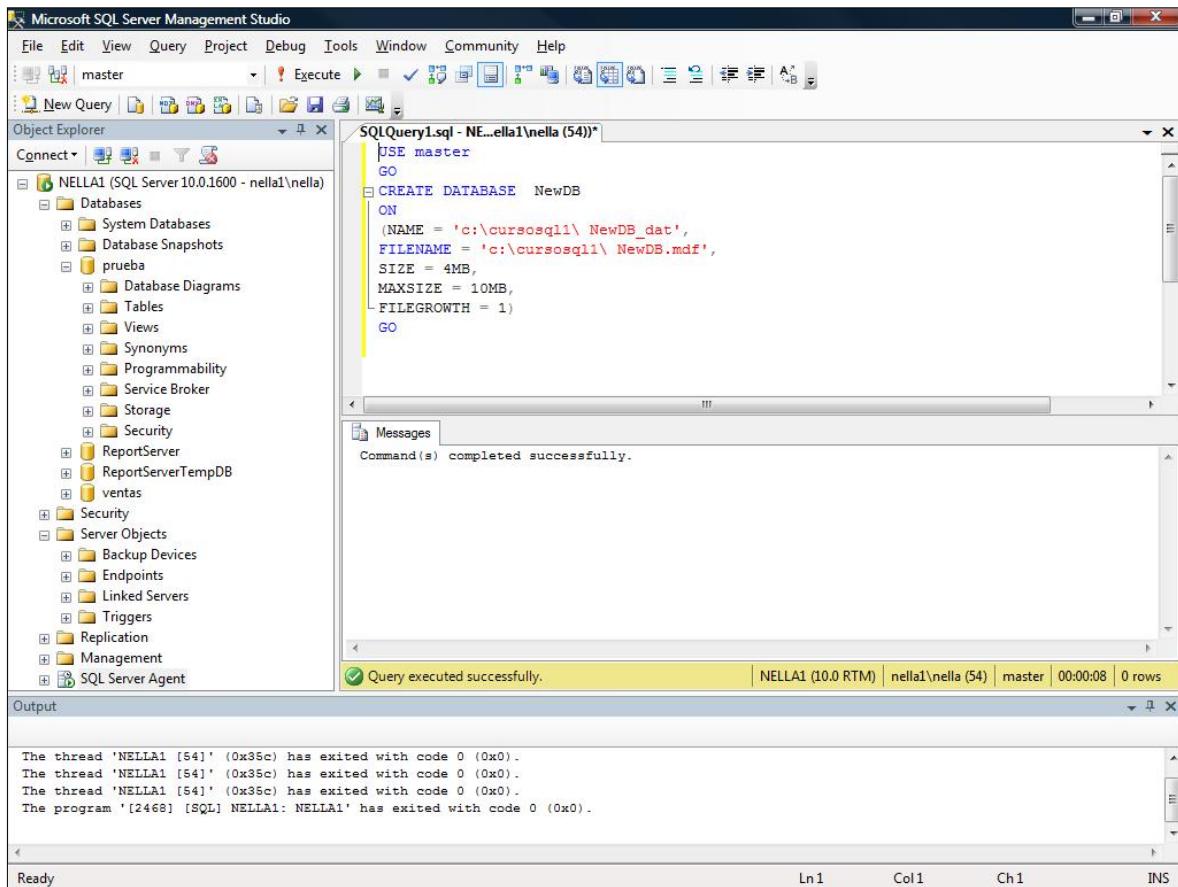
Por medio de la cláusula CREATE, se procede a crear la base de datos NewDB.

```
USE master
GO
CREATE DATABASE NewDB
ON
(NAME = 'c:\cursosql1\ NewDB_dat',
FILENAME = 'c:\cursosql1\ NewDB.mdf',
SIZE = 4MB,
MAXSIZE = 10MB,
FILEGROWTH = 1)
GO
```

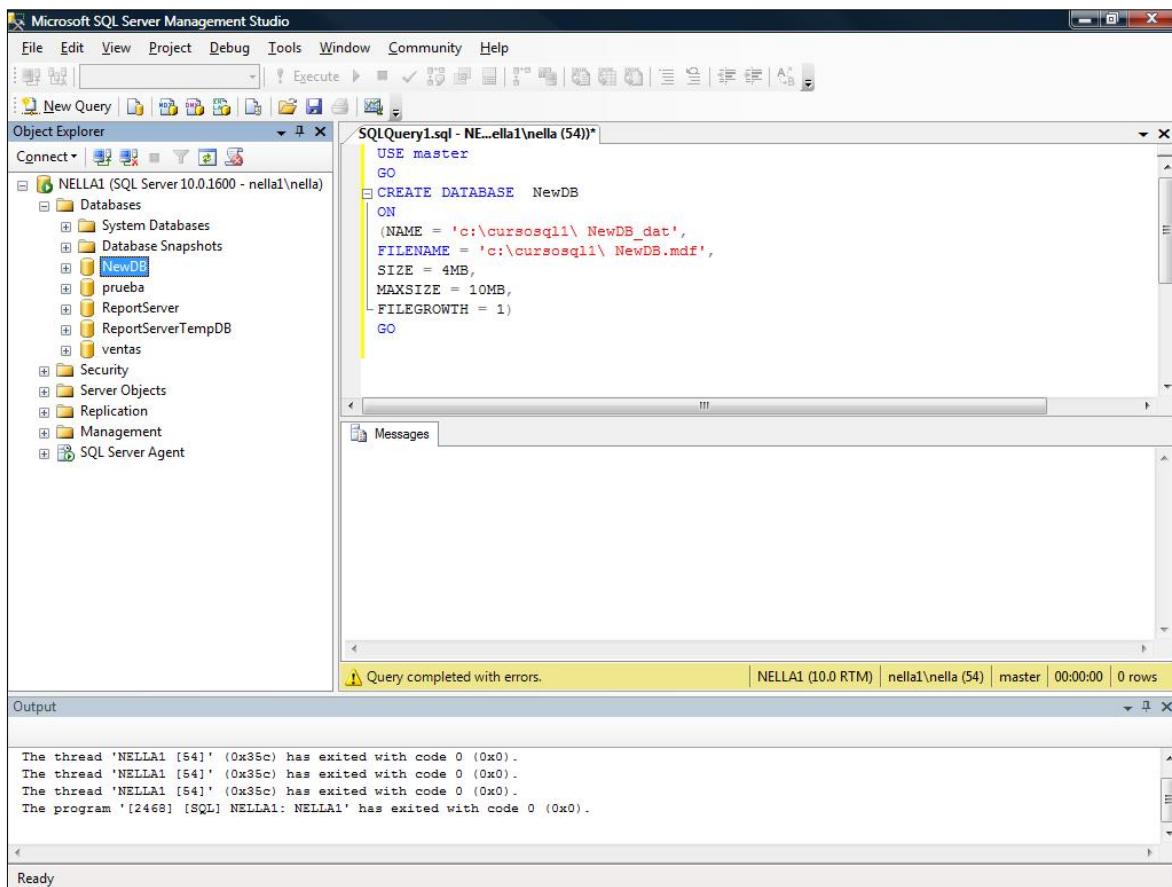
SIZE: se puede expresar en kilobytes (KB), megabytes (MB), gigabytes (GB) o terabytes (TB), si no se especifica, toma MB por defecto, es aquí donde se especifica el tamaño inicial de la base de datos.

MAXSIZE: se puede expresar en kilobytes (KB), megabytes (MB), gigabytes (GB) o terabytes (TB), si no se especifica, toma MB por defecto, es aquí donde se especifica el tamaño máximo de la base de datos.

FILEGROWTH: Crecimiento del archivo de base de datos.



Una vez ejecutado el código anterior, al dar F5 se refresca la pantalla y se muestra la base de datos NewDB en el Explorador de Objetos (Object Explorer).



Creación de una tabla

Cláusula

```

CREATE TABLE T1
(COL1 tipo (tamaño) índice 1,
COL2 tipo (tamaño) índice 2,...,)

```

T1: Nombre de la tabla o entidad a crear.

COL1: Nombre del campo o los campos que se van a crear en la nueva tabla.

COL2: La tabla creada debe tener más de un campo.

tipo: Tipo de datos del campo

tamaño: Es el tamaño del campo, aplica para campos de texto.

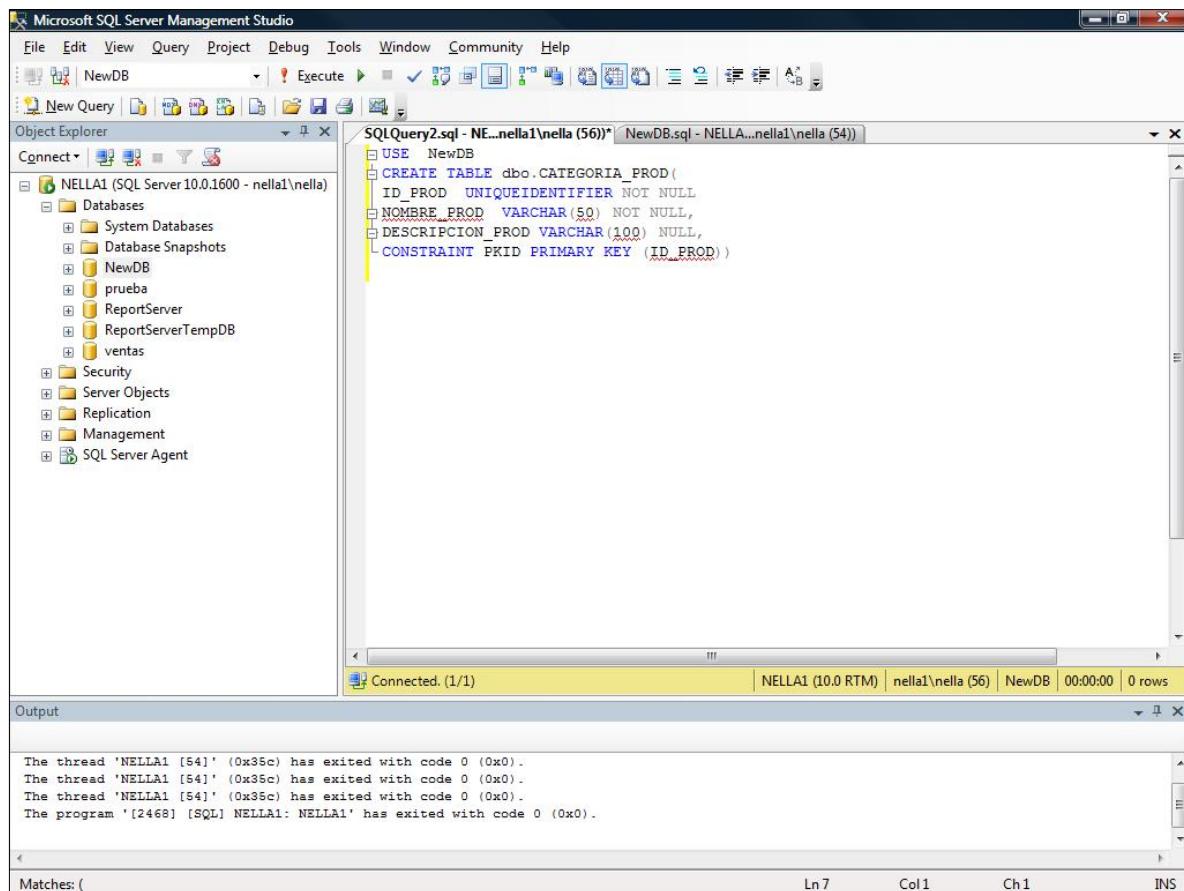
Índice: Cláusa CONSTRAINT

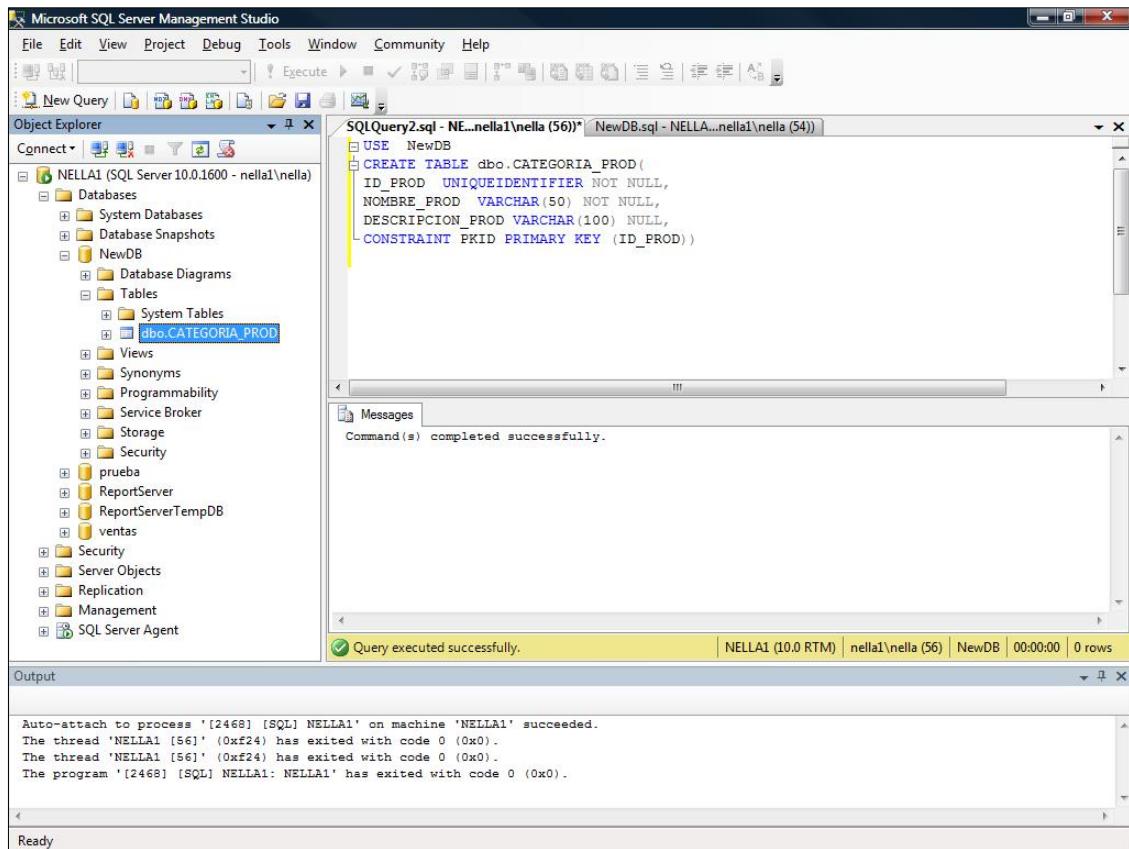
Por medio de la cláusula CREATE, se procede a crear la tabla dbo.CATEGORIA_PROD, en la base de datos NewDB, para hacer referencia a la base de datos a utilizar se utiliza el comando USE, seguido del nombre de la base de datos.

```

USE NewDB
CREATE TABLE dbo.CATEGORIA_PROD(
ID_PROD UNIQUEIDENTIFIER NOT NULL
NOMBRE_PROD VARCHAR(50) NOT NULL,
DESCRIPCION_PROD VARCHAR(100) NULL,
CONSTRAINT PKID PRIMARY KEY (ID_PROD ))

```





ALTER

Permite la modificación de tablas o base de datos.

Cláusula

ALTER TABLE Nombre de Tabla
ADD Nombre de Columna Tipo de Dato;

Agregar constraint por medio del ALTER

Cláusula

ALTER TABLE nombre de tabla
ADD CONSTRAINT
 Nombre de llave primaria **PRIMARY**
KEY(Campo de llave primaria)
ON [PRIMARY];

ALTER TABLE nombre de tabla hija
ADD CONSTRAINT nombre de llave foránea
FOREIGN KEY (columna o columnas que conforman la llave foránea)
REFERENCES nombre de tabla padre (Campo o campos de llave primaria de tabla padre);

Borrado de columna y constraint por medio de Alter

```
ALTER TABLE nombre de tabla  
DROP CONSTRAINT nombre del constraint;
```

Borrado de la columna de una tabla por medio del ALTER

Cláusula

```
ALTER TABLE nombre de tabla  
DROP COLUMN nombre de columna;
```

DROP

Borra físicamente un objeto de base de datos: tablas, vistas lógicas.

Cláusula

```
DROP TABLE nombre de objeto;
```

Sentencias DML

SELECT

Lanza consultas, determinar cuáles son los campos (columnas) que se quieren leer y su organización, su sintaxis en forma general:

Cláusula

```
SELECT columna1, columna2, columna3 FROM nombre de tabla
```

Extrae la información de las columnas de la tabla seleccionada

```
SELECT * FROM nombre de tabla
```

Extrae toda la información de una tabla

Cláusula general

```
SELECT [DISTINCT][TOP (n)] *, columnas, or expresiones  
FROM datos de origen(nombre de tabla o tablas)  
[WHERE condiciones]  
[GROUP BY columnas]  
[HAVING condiciones]  
[ORDER BY columnas];
```

[DISTINCT]: Elimina los duplicados a la hora de mostrar los resultados de la consulta.

[TOP()]: Muestra los primeros registros según la cantidad especificada.

[FROM]: Indica de que tabla se extraerán los datos a mostrar en la consulta.

[WHERE]: Funciona como filtro, mostrando solamente en la consulta los datos que cumplen con la condición aquí establecida.

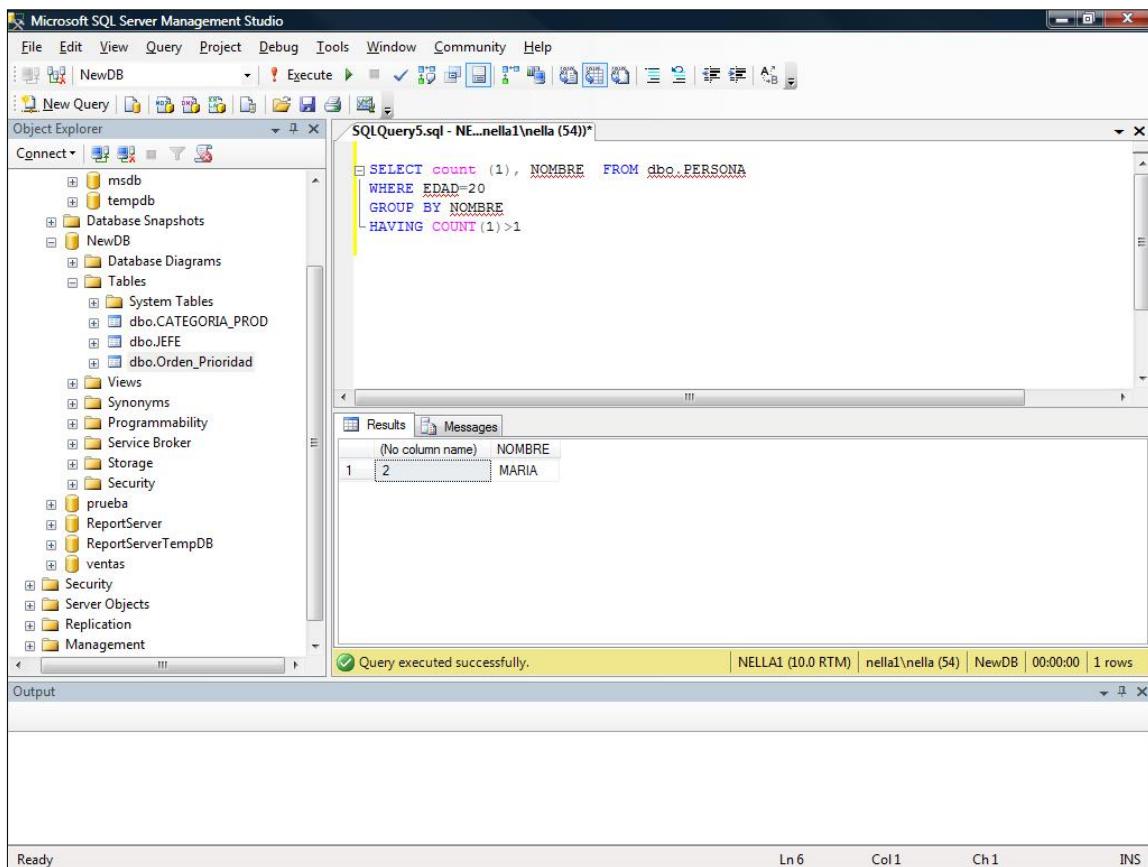
[GROUP BY]: Ordena los datos mostrados según los campos establecidos en este comando

[HAVING]: Funciona como el where .

[ORDER BY]: Ordena los datos de forma ascendente o descendente, por defecto es ascendente.

Ejemplo

```
SELECT count (1), NOMBRE FROM dbo.PERSONA  
WHERE EDAD=20  
GROUP BY NOMBRE  
HAVING COUNT(1)>1
```



INSERT

Permite el ingreso de datos a una tabla

Cláusula

```

INSERT [INTO] tabla [(columnas, ...)]
VALUES (value,...), (value,...), ... ;

```

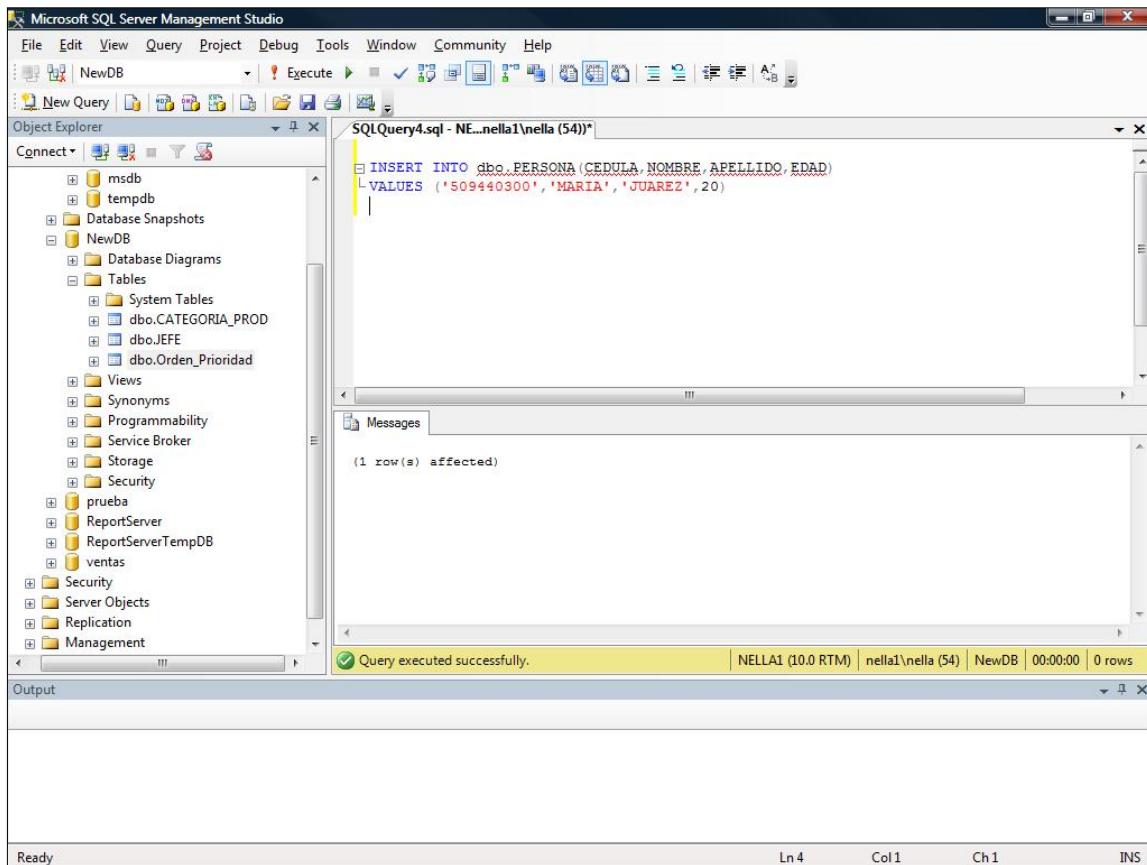
Lo que se encuentra entre corchetes es opcional.

Ejemplo:

```

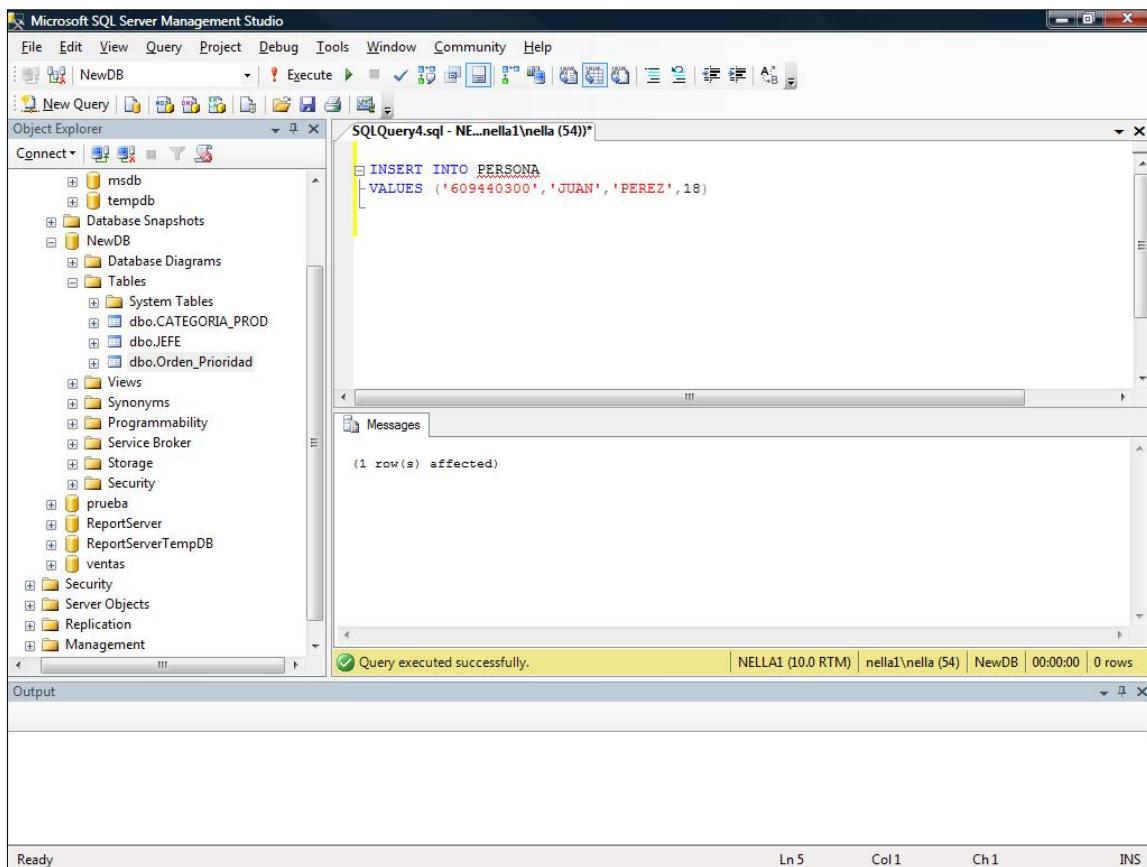
INSERT INTO PERSONA (CEDULA, NOMBRE, APELLIDO, EDAD)
VALUES (509440300,'MARIA','JUAREZ',20)

```



O se puede realizar de la siguiente forma siempre y cuando estas sean las únicas columnas de la tabla y en la sentencia values se ingresen en orden

```
INSERT INTO PERSONA
VALUES ('609440300','JUAN','PEREZ',18)
```



UPDATE

Permite la actualización de datos en una tabla

Cláusula

```
UPDATE Tabla
SET columna = expresión,
columna = valor...
[FROM datos de origen]
[WHERE condiciones];
```

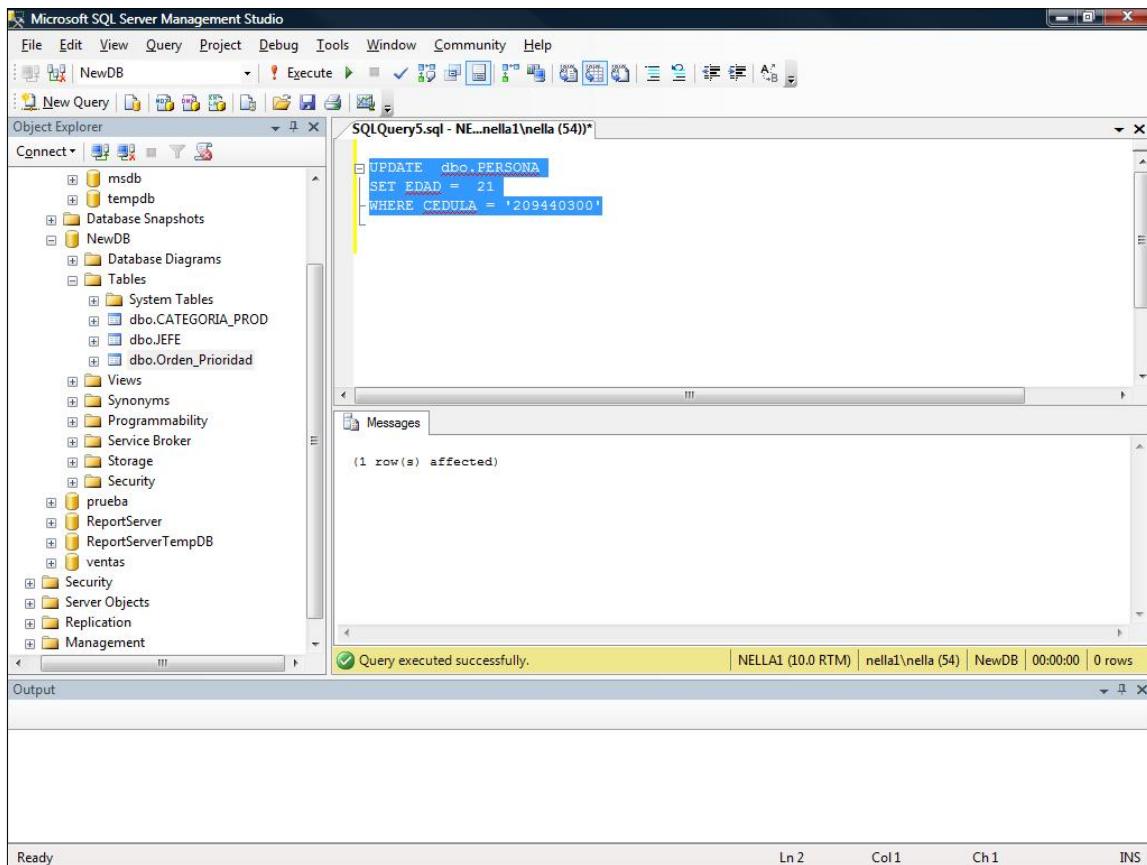
Lo que se encuentra entre corchetes es opcional

Importante

Si no se utiliza el comando WHERE, se modifica la información de la columna seleccionada para todos los registros de la tabla.

Ejemplo

```
UPDATE dbo.PERSONA
SET EDAD = 15
WHERE CEDULA = '209440300'
```



DELETE

Permite eliminar los registros de una tabla, manteniendo su estructura.

Cláusula

```
DELETE [FROM] Tabla
[FROM datos de origen]
[WHERE condición(es)];
```

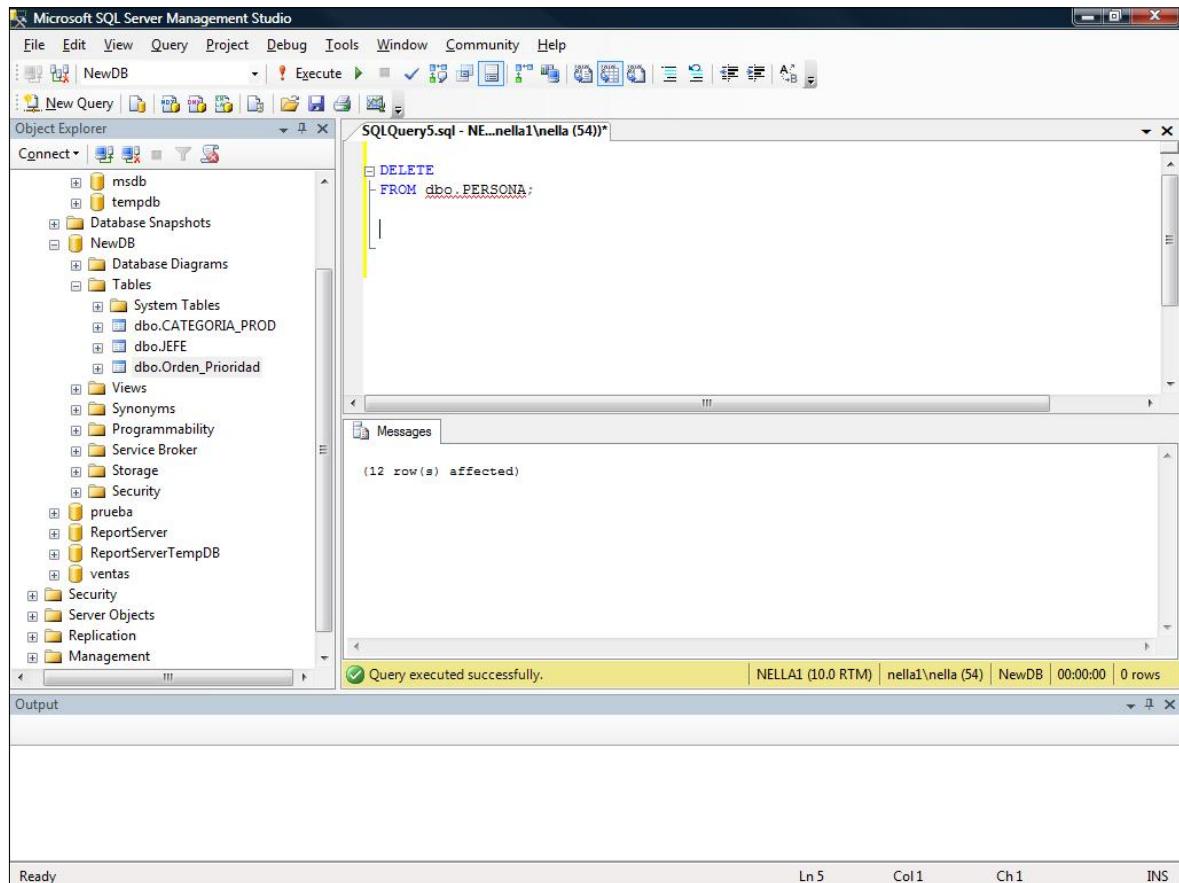
Lo que se encuentra entre corchetes es opcional.

Importante:

Si no se utiliza la cláusula el WHERE se eliminaran todos los registros de la tabla.

Ejemplo :

```
DELETE  
FROM dbo.PERSONA;
```



Elimina todos los registros de la tabla PERSONA, sin eliminar la tabla.

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, the database 'NewDB' is selected. In the center pane, a query window titled 'SQLQuery5.sql - NE...nella1\abella (54)*' contains the following SQL code:

```
SELECT * FROM dbo.PERSONA
```

The Results tab shows a table with columns CEDULA, NOMBRE, APELLIDO, and EDAD, but it is empty. Below the results, a message indicates: "Query executed successfully." At the bottom of the screen, status information includes: NELLA1 (10.0 RTM) | nella1\abella (54) | NewDB | 00:00:00 | 0 rows.

**DELETE FROM PERSONA
WHERE CEDULA = '209440300';**

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, the database 'NewDB' is selected. In the center pane, a query window titled 'SQLQuery5.sql - NE...nella1\abella (54)*' contains the following SQL code:

```
DELETE FROM PERSONA  
- WHERE CEDULA = '209440300';
```

The Messages tab shows the output: "(1 row(s) affected)". Below the messages, a message indicates: "Query executed successfully." At the bottom of the screen, status information includes: NELLA1 (10.0 RTM) | nella1\abella (54) | NewDB | 00:00:00 | 0 rows.

Elimina de la tabla dbo.PERSONA, todos los registros donde el campo Cedula sea igual a '209440300'.

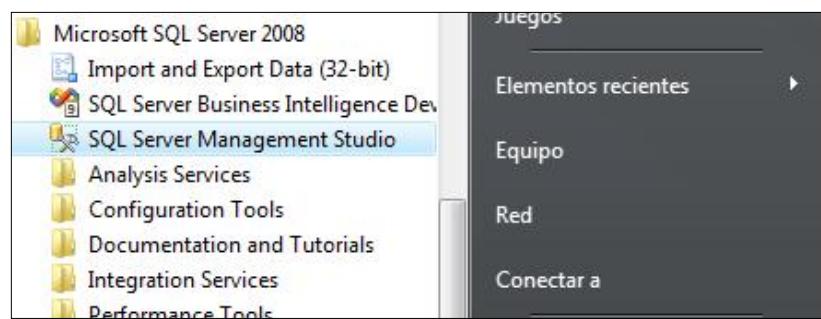
Práctica #1 Crear objetos

Objetivo :

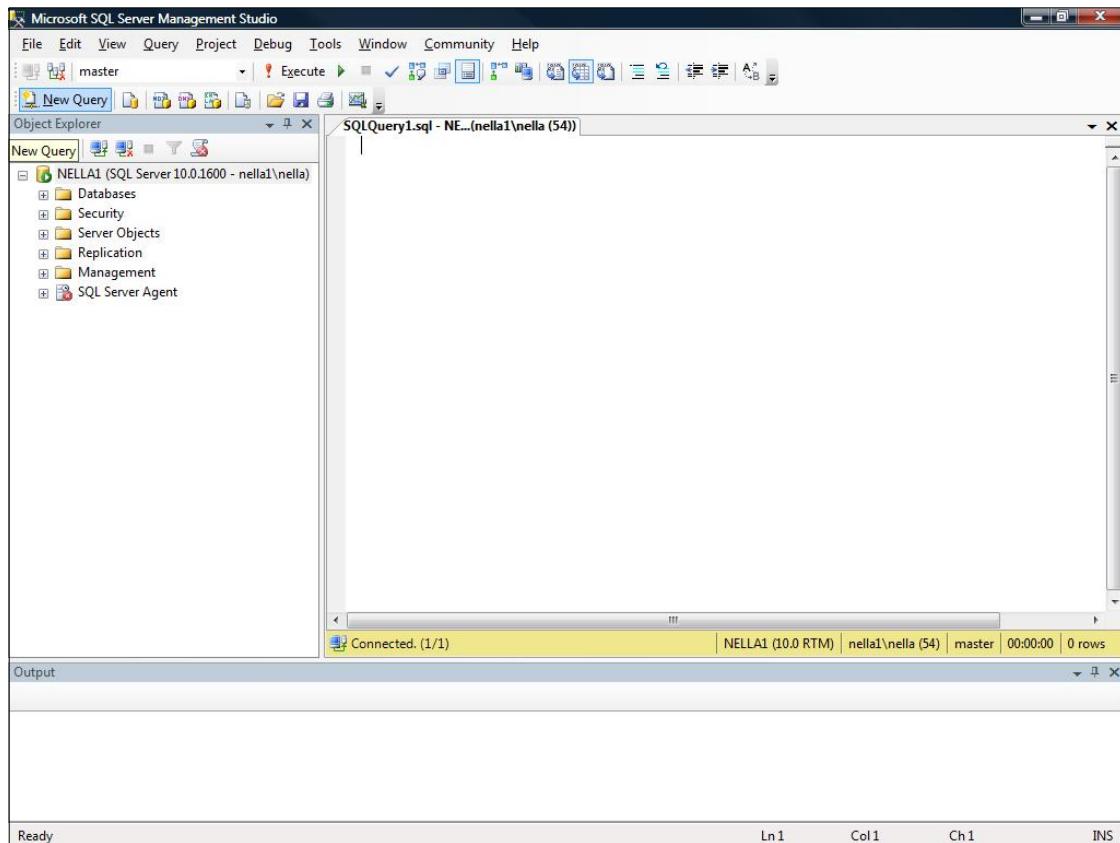
Se pretende que el estudiante con esta práctica ingrese al SQL Server 2008 e inicie su familiarización con la herramienta.

Como crear el query

En el menú de inicio, se ingresa a Microsoft SQL Server 2008 y ahí a SQL Server Management Studio

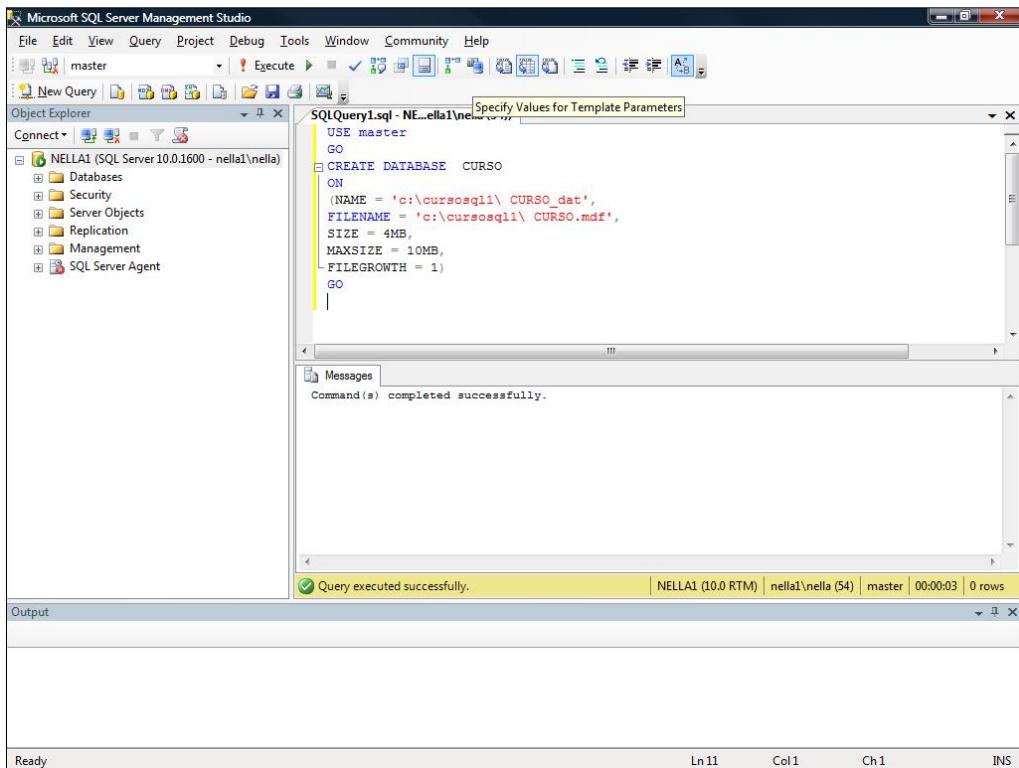


Se da click en Connect, para conectar al servidor y al lado superior izquierdo de la pantalla se da click en New Query .

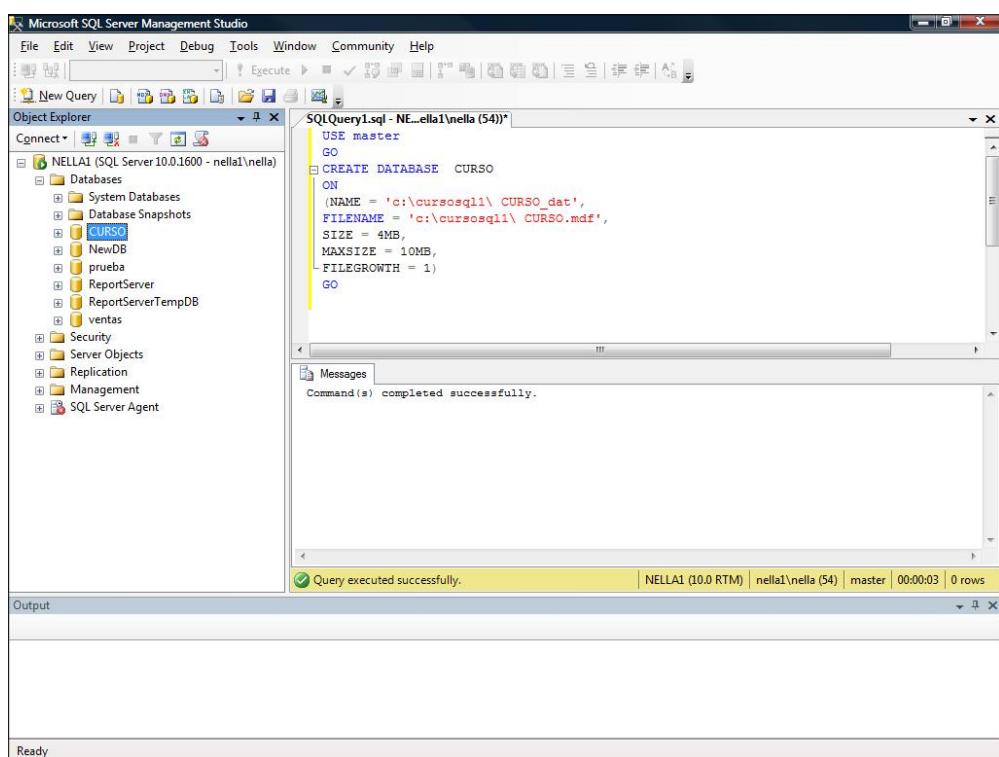


1. Con base al siguiente código crear la base de datos CURSO

```
USE master
GO
CREATE DATABASE CURSO
ON
(NAME = 'c:\cursosql1\CURSO_dat',
FILENAME = 'c:\cursosql1\CURSO.mdf',
SIZE = 4MB,
MAXSIZE = 10MB,
FILEGROWTH = 1)
GO
```



Una vez ejecutado el código anterior, al dar F5 se refresca la pantalla y se muestra la base de datos CURSO en el Explorador de Objetos.



2. Crear tanto en la base de datos curso como en la base de datos NEWDB, la siguiente tabla

Tabla PERSONA

Campos :

CEDULA INT

NOMBRE VARCHAR(50)

APELLIDO VARCHAR(50)

EDAD INT

3. Insertar los siguientes datos en la tabla Persona en ambas bases de datos

Cedula	Nombre	Apellido	Edad
509440300	MARIA	JUAREZ	20
609440300	JUAN	PEREZ	18
709440300	ANA	ARIAS	25
809440300	TERESA	PAZ	40
909440300	JOSE	ARGUEDAZ	30
109440300	JUANA	PAZ	50
209440300	JULIO	MONGE	35
309440300	KAREN	VARGAS	20
409440300	MARIO	MORA	18
519440300	JULIA	DIAZ	20

Nota: Solución [Anexo I](#)

Operadores Lógicos

Cuando se necesitan múltiples condiciones para el WHERE se utilizan los operadores lógicos:

AND = y

OR = O

NOT = NO

Funcionan de la siguiente manera

Condición1	Operador Lógico	Condición 2		Resultado
V	AND	V	=	V
V	AND	F	=	F
F	AND	F	=	F
V	OR	V	=	V
V	OR	F	=	V
F	OR	V	=	V
F	OR	F	=	F

El cuadro anterior se traduce de la siguiente manera

AND: Deben cumplirse ambas condiciones para que la consulta muestre las filas o registros que cumplen con las condiciones establecidas. Con una condición que no se cumpla, no se muestran filas o registros.

OR: Cualquiera de las condiciones que se cumplan la consulta muestra las filas o registros que cumplen con la o las condiciones establecidas. No necesariamente se tienen que cumplir todas las condiciones como en el AND.

NOT: Permite mostrar las filas o registros que no coincidan con un valor.

Ejemplo:

Mostrar la cédula y nombre de las personas donde el nombre inicie con MA o la edad este entre 20 y 35, y el nombre tenga una S.

```
SELECT CEDULA,NOMBRE
FROM dbo.PERSONA
WHERE
NOMBRE LIKE 'MA%'
OR
EDAD BETWEEN 20 AND 35
AND
NOMBRE LIKE '%S%';
```

```

SELECT CEDULA, NOMBRE
FROM dbo.PERSONA
WHERE
NOMBRE LIKE 'MA%'
OR
EDAD BETWEEN 20 AND 35
AND
NOMBRE LIKE '%S%';

```

CEDULA	NOMBRE
409440300	MARIO
509440300	MARIA
549440300	MARIA
909440300	JOSE

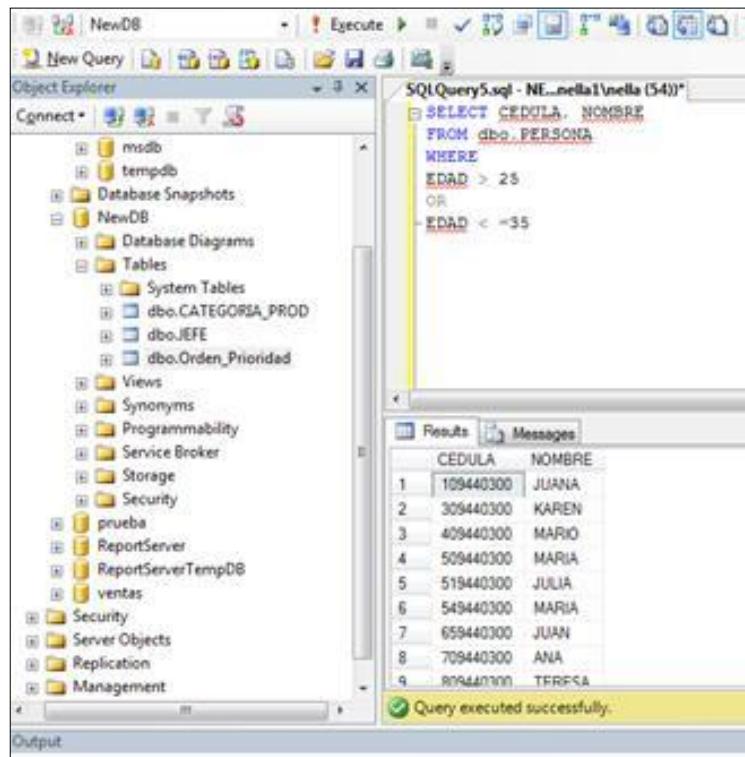
Query executed successfully. | NELLA1 (10.0 RTM) | nella1\abella (54) | NewDB | 00:00:00 | 4 rows

Operadores de Comparación

Descripción	Operador	Ejemplo
Igual	=	Cantidad = 12
Mayor que	>	Cantidad > 12
Mayor que o igual a	>=	Cantidad >= 12
Menor que	<	Cantidad < 12
Menor o igual que	<=	Cantidad <= 12
Diferente	<>, !=	Cantidad <> 12, Cantidad != 12
No menor que	!<	Cantidad !< 12
No mayor que	!>	Cantidad !> 12

Ejemplo:

```
SELECT CEDULA, NOMBRE  
FROM dbo.PERSONA  
WHERE  
EDAD > 25  
OR  
EDAD < =35
```



```
SELECT CEDULA, NOMBRE  
FROM dbo.PERSONA  
WHERE  
EDAD > 25  
OR  
EDAD < =35
```

	CEDULA	NOMBRE
1	109440300	JUANA
2	309440300	KAREN
3	409440300	MARIO
4	509440300	MARIA
5	519440300	JULIA
6	549440300	MARIA
7	659440300	JUAN
8	709440300	ANA
9	809440300	TERESA

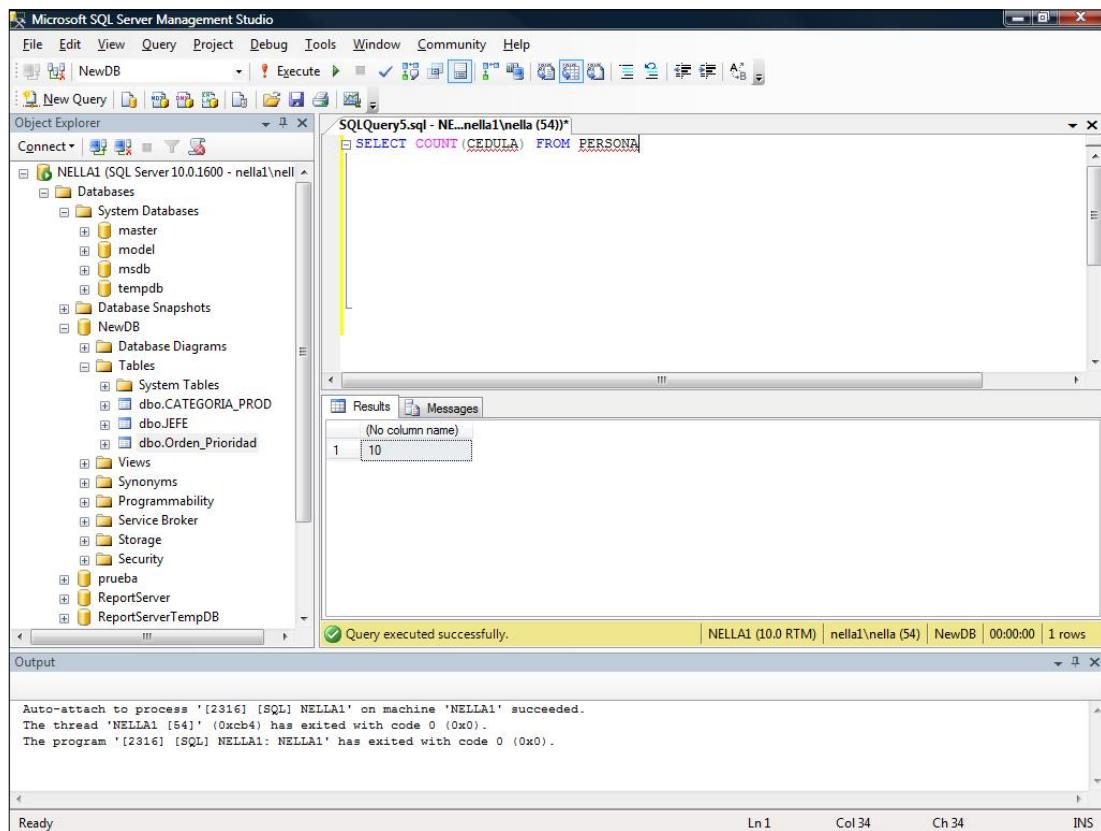
Query executed successfully.

Funciones de agregado

Función	Significado
AVG()	Calcula el promedio de un dato o de una expresión aritmética construida por las columnas proyectadas de una tabla.
Count()	Cuenta el número de valores contenidos en una columna o tabla.
MAX()	Devuelve el valor máximo de una columna numérica.
MIN()	Devuelve el valor mínimo de una columna numérica.
SUM()	Suma los valores de una columna numérica o de una expresión.
STDEV()	Desviación estándar de la población de todos los valores dados por expresión
STDEVP()	Desviación estándar de todos los valores de expresión
VARP(expresión)	Varianza estadística de la población de todos los valores dados por expresión
VAR(expresión)	Varianza estadística de todos los valores que toma la expresión

Ejemplo:

```
SELECT COUNT(CEDULA) FROM PERSONA
```



The screenshot shows the Microsoft SQL Server Management Studio interface. In the center, a query window displays the following SQL code:

```
SELECT COUNT(CEDULA) FROM PERSONA
```

The results pane below shows a single row with the value 10. At the bottom of the screen, the output pane displays the message:

```
Auto-attach to process '[2316] [SQL] NELLA1' on machine 'NELLA1' succeeded.  
The thread 'NELLA1 [54]' (0xcb4) has exited with code 0 (0x0).  
The program '[2316] [SQL] NELLA1: NELLA1' has exited with code 0 (0x0).
```

```
SELECT COUNT(CEDULA) AS Cantidad
FROM PERSONA
```

The screenshot shows the SSMS interface. In the top menu bar, the 'File' option is highlighted. The 'Object Explorer' pane on the left shows a tree structure of database objects under the 'NewDB' database, including System Databases (master, model, msdb, tempdb), Database Snapshots, and Tables (dbo.CATEGORIA_PROD, dbo.JEFE, dbo.Orden_Prioridad). The 'Results' tab in the bottom pane displays the output of the executed query:

Cantidad
10

Se agrega a la columna resultante un nombre por medio del AS, el nombre Cantidad

```
SELECT AVG(EDAD) FROM PERSONA
```

Permite mostrar la edad promedio de las personas

The screenshot shows the SSMS interface. In the top menu bar, the 'File' option is highlighted. The 'Object Explorer' pane on the left shows a tree structure of database objects under the 'NELLAI' database, including Databases, Security, Server Objects, Replication, Management, and SQL Server Agent. The 'Results' tab in the bottom pane displays the output of the executed query:

EDAD_PROMEDIO
24

```
SELECT SUM (EDAD)/COUNT (*) FROM PERSONA
```

The screenshot shows the Microsoft SQL Server Management Studio (SSMS) interface. The top menu bar includes File, Edit, View, Query, Project, Debug, Tools, Window, Community, and Help. Below the menu is a toolbar with various icons. The Object Explorer on the left shows a connection to 'NELLA1 (SQL Server 10.0.1600 - nella1\abella)'. Under the connection, there are nodes for Databases, Security, Server Objects, Replication, Management, and SQL Server Agent. The central pane displays a query window titled 'SQLQuery1.sql - NE...nella1\abella (52)*' containing the following SQL code:

```
SELECT SUM (EDAD)/COUNT (*) FROM PERSONA
```

Below the query window is a results pane. The 'Results' tab is selected, showing a single row of data:

	(No column name)
1	24

Consultas de Selección

Anteriormente se vio el comando SELECT, sin embargo se vio de una forma muy básica, en las siguientes secciones se profundizará más en la utilización de este comando.

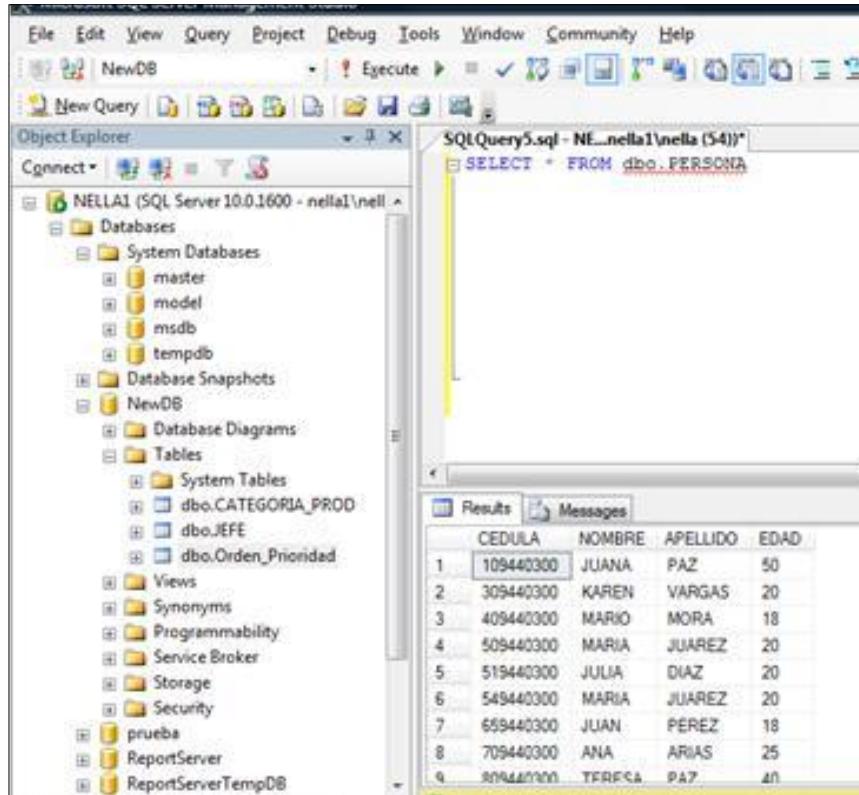
Consultas Básicas

Entre las consultas básicas se tienen, las consultas de todos los registros de una tabla o las que utilizan la cláusula WHERE y alguno de los operadores de comparación

```
SELECT * FROM TABLA
SELECT * FROM TABLA WHERE COLUM1 =D
SELECT COLUM1, COLUM2, COLUM3 FROM TABLA WHERE COLUM1 < COLUM2
SELECT COLUM1, COLUM2, COLUM3 FROM TABLA WHERE COLUM1 = COLUM2 + COLUM3
SELECT COLUM1, COLUM2, COLUM3 FROM TABLA WHERE COLUM1 >= 300
```

Ejemplo:

```
SELECT * FROM dbo.PERSONA
```



```
SELECT * FROM dbo.PERSONA
WHERE EDAD = 20
```

	CEDULA	NOMBRE	APELLIDO	EDAD
1	309440300	KAREN	VARGAS	20
2	509440300	MARIA	JUAREZ	20
3	519440300	JULIA	DIAZ	20
4	549440300	MARIA	JUAREZ	20

 A message at the bottom of the Results pane says 'Query executed successfully.'"/>

Ordenar registros

Por medio de la cláusula ORDER BY se permite ordenar un conjunto de resultados de la columna o columnas especificadas.

ASC: permite ordenar en forma ascendente, en caso de ser carácter ordena de A –Z, y en caso de números ordena de menor a mayor.

DESC: permite ordenar en forma descendente, en caso de ser carácter ordena de Z –A, y en caso de números ordena de mayor a menor.

Por defecto la cláusula ORDER BY ordena en forma ascendente

Ejemplos:

```
SELECT NOMBRE, APELLIDO FROM PERSONA
WHERE EDAD >20
ORDER BY NOMBRE ASC
```

The screenshot shows the SSMS interface. In the Object Explorer, a connection to 'NELLA1' is selected. In the center pane, a query window titled 'SQLQuery1.sql' contains the following SQL code:

```
SELECT NOMBRE, APELLIDO FROM PERSONA
WHERE EDAD >20
ORDER BY NOMBRE ASC
```

The results pane shows the output of the query:

	NOMBRE	APELLIDO
1	ANA	ARIAS
2	JUANA	PAZ
3	TERESA	PAZ

```
SELECT NOMBRE, APELLIDO FROM PERSONA
WHERE EDAD >20
ORDER BY NOMBRE
```

The screenshot shows the SSMS interface. In the Object Explorer, a connection to 'NELLA1' is selected. In the center pane, a query window titled 'SQLQuery1.sql' contains the same SQL code as the previous screenshot:

```
SELECT NOMBRE, APELLIDO FROM PERSONA
WHERE EDAD >20
ORDER BY NOMBRE ASC
```

The results pane shows the same output:

	NOMBRE	APELLIDO
1	ANA	ARIAS
2	JUANA	PAZ
3	TERESA	PAZ

Muestra el mismo resultado de la consulta anterior

```
SELECT NOMBRE, APELLIDO FROM PERSONA
WHERE EDAD >20
ORDER BY NOMBRE DESC
```

The screenshot shows the SSMS interface. In the top-left, the Object Explorer tree shows a connection to 'NELLAI' with nodes for Databases, Security, Server Objects, Replication, Management, and SQL Server Agent. The top-right shows the 'SQLQuery1.sql - NE...nella1\abella (52)*' window containing the provided SQL query. Below it, the 'Results' pane displays the query's output:

	NOMBRE	APELLIDO
1	TERESA	PAZ
2	JUANA	PAZ
3	ANA	ARIAS

Consultas con predicado

Las consultas con predicado son aquellas que llevan el predicado entre la cláusula y el nombre de la primera columna a recuperar.

Entre los predicados se tiene

ALL: Devuelve todos los datos de una columna específica de la tabla.

Ejemplo:

```
SELECT ALL
NOMBRE from dbo.PERSONA
```

```
File Edit View Query Project Debug Tools Window Community Help
NewDB New Query
Object Explorer
Connect > Connect to...
Connect Object Explorer - Microsoft SQL Server 2008 (T-SQL) - [nella]
Connect to Microsoft SQL Server 2008 (T-SQL) - [nella]
Databases Security Server Objects Replication Management SQL Server Agent
SQLQuery1.sql - NE...nella1\abella (54)*
SELECT ALL
NOMBRE
from dbo.PERSONA
Results Messages
NOMBRE
1 JUANA
2 KAREN
3 MARIO
4 MARIA
5 JULIA
6 MARIA
7 JUAN
8 ANA
9 TERESA
Query executed successfully.
```

TOP(n): Devuelve los primeros registros según la cantidad solicitada (n).

Si se utiliza la cláusula ORDER BY los datos de inicio o final según esta cláusula

Ejemplo:

```
SELECT TOP 5 NOMBRE ,APELIDO, EDAD
FROM PERSONA
```

```
File Edit View Query Project Debug Tools Window Community Help
NewDB Execute
Object Explorer Execute X
Connect New Query
NELLA1 (SQL Server 10.0.1600 - nella1\nella)
Databases Security Server Objects Replication Management SQL Server Agent
SQlQuery1.sql - NE...nella1\abella (54)*
SELECT TOP 5 NOMBRE ,APELIDO , EDAD
FROM PERSONA

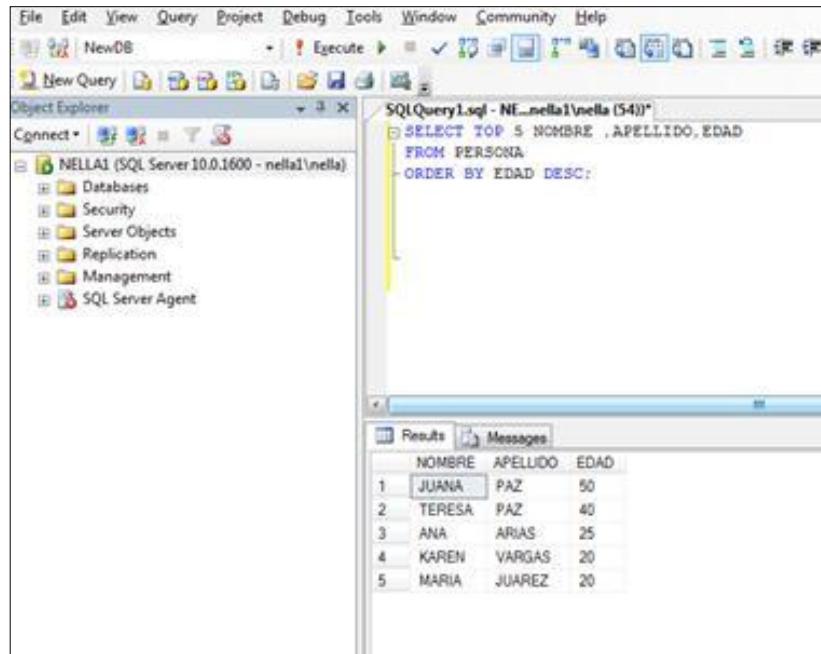
Results Messages
NOMBRE APELLIDO EDAD
1 JUANA PAZ 50
2 KAREN VARGAS 20
3 MARIO MORA 18
4 MARIA JUAREZ 20
5 JULIA DIAZ 20
```

```
SELECT TOP 5 NOMBRE ,APELIDO , EDAD
FROM PERSONA
ORDER BY EDAD ;
```

```
File Edit View Query Project Debug Tools Window Community Help
NewDB Execute
Object Explorer Execute X
Connect New Query
NELLA1 (SQL Server 10.0.1600 - nella1\nella)
Databases Security Server Objects Replication Management SQL Server Agent
SQlQuery1.sql - NE...nella1\abella (54)*
SELECT TOP 5 NOMBRE ,APELIDO , EDAD
FROM PERSONA
ORDER BY EDAD

Results Messages
NOMBRE APELLIDO EDAD
1 JOSE ARGUEDAZ 15
2 MARIO MORA 18
3 JUAN PEREZ 18
4 MARIA JUAREZ 20
5 JULIA DIAZ 20
```

```
SELECT TOP 5 NOMBRE ,APELIDO,EDAD
FROM PERSONA
ORDER BY EDAD DESC;
```



The screenshot shows the SQL Server Management Studio interface. In the Object Explorer, a connection to 'NELLAI' is selected. In the center pane, a query window titled 'SQLQuery1.sql - NE...nella1\abella (54)*' contains the following code:

```
SELECT TOP 5 NOMBRE , APELLIDO, EDAD
FROM PERSONA
ORDER BY EDAD DESC;
```

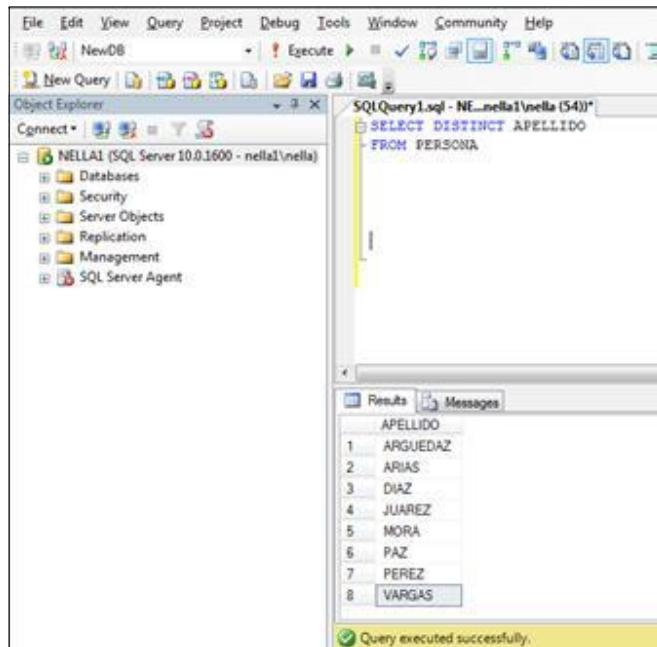
The results pane below shows the output of the query:

	NOMBRE	APELLIDO	EDAD
1	JUANA	PAZ	50
2	TERESA	PAZ	40
3	ANA	ARIAS	25
4	KAREN	VARGAS	20
5	MARIA	JUAREZ	20

DISTINCT: Elimina los duplicados a la hora de mostrar los resultados de la consulta.

Ejemplo:

```
SELECT DISTINCT APELLIDO
FROM PERSONA
```



The screenshot shows the SQL Server Management Studio interface. In the Object Explorer, a connection to 'NELLAI' is selected. In the center pane, a query window titled 'SQLQuery1.sql - NE...nella1\abella (54)*' contains the following code:

```
SELECT DISTINCT APELLIDO
FROM PERSONA
```

The results pane below shows the output of the query:

APELLIDO
ARGUEDAZ
ARIAS
DIAZ
JUAREZ
MORA
PAZ
PEREZ
VARGAS

At the bottom of the results pane, a message indicates: 'Query executed successfully.'

Alias

En algunas ocasiones es necesario renombrar las columnas que se devuelven de una consulta y para esto se utiliza la palabra reservada AS.

Ejemplo:

```
SELECT COUNT(*) AS CANTIDAD_PERSONAS
FROM PERSONA
```

The screenshot shows the SQL Server Management Studio interface. In the Object Explorer, a connection to 'NELLAL1' is selected. In the center pane, a query window titled 'SQLQuery1.sql - NE...nella1\abella (54)' contains the following code:

```
SELECT COUNT(*) AS CANTIDAD_PERSONAS
FROM PERSONA
```

In the results pane below, there is one row with the value '10' under the column labeled 'CANTIDAD_PERSONAS'. The 'Messages' tab is also visible in the results pane.

Si no se utiliza el alias, la columna resultante se muestra sin nombre

```
SELECT COUNT(*) FROM PERSONA
```

The screenshot shows the SQL Server Management Studio interface. In the Object Explorer, a connection to 'NELLAL1' is selected. In the center pane, a query window titled 'SQLQuery1.sql - NE...nella1\abella (54)' contains the following code:

```
SELECT COUNT(*) FROM PERSONA
```

In the results pane below, there is one row with the value '10' under the column labeled '(No column name)'. A message at the bottom of the results pane says 'Query executed successfully.' The 'Messages' tab is also visible in the results pane.

En algunas ocasiones se utilizan en consultas para renombrar tablas, por lo general cuando la consulta se realiza en más de una tabla.

Ejemplo:

```
SELECT A.CODIGO, A.DESCRIPCION
FROM CURSO A
CROSS JOIN HISTORIAL
```

Como se ve en el ejemplo anterior, cuando se utilizan los Alias en consultas de varias tablas no hace falta utilizar la palabra clave AS, basta con poner la letra que va a identificar la tabla, después del nombre de la tabla y esta misma letra antes del nombre de cada columna de la tabla.

Bases de datos externas

Por medio de una instrucción SQL, se puede realizar una consulta a una tabla de una base de datos externa a la cual se está trabajando.

```
SELECT FROM [SERVIDOR].BASE DE DATOS.DUEÑO(OWNER).TABLA
```

Ejemplos:

Se requiere la información de la tabla CURSO, que se encuentra en la base de datos NewDB, pero en este momento nos encontramos en la base de datos CURSO.

```
SELECT * FROM
NELLA1.NewDB.dbo.CURSO
```

The screenshot shows the SSMS interface with the following details:

- Query Window:** The title bar says "CURSO". The query text is:


```
SELECT * FROM
NELLA1.NewDB.dbo.CURSO
```
- Object Explorer:** Shows the connection to "NELLA1 (SQL Server 10.0.1600 - nella1\abella)". Under "Databases", "Security", "Server Objects", "Replication", "Management", and "SQL Server Agent" are listed.
- Results Grid:** The results of the query are displayed in a table with the following data:

	CODIGO	DESCRIPCION	FECHA_INICIO	FECHA_FIN
1	1	SQL SERVER 2008	2011-06-05 00:00:00.000	2011-08-28 00:00:00.000
2	2	INTERNET	2011-06-05 00:00:00.000	2011-08-28 00:00:00.000
3	3	ACCES	2011-06-05 00:00:00.000	2011-08-28 00:00:00.000
4	4	WORD 2007	2011-06-05 00:00:00.000	2011-08-28 00:00:00.000
5	5	EXCEL 2007	2011-06-05 00:00:00.000	2011-08-28 00:00:00.000

Como la base de datos NewDB, se encuentra en el mismo servidor que la base de datos curso, se puede omitir el nombre del servidor y el resultado sería el mismo

```
SELECT * FROM
NewDB.dbo.CURSO
```

CODIGO	DESCRIPCION	FECHA_INICIO	FECHA_FIN
1	SQL SERVER 2008	2011-06-05 00:00:00.000	2011-08-28 00:00:00.000
2	INTERNET	2011-06-05 00:00:00.000	2011-08-28 00:00:00.000
3	ACCES	2011-06-05 00:00:00.000	2011-08-28 00:00:00.000
4	WORD 2007	2011-06-05 00:00:00.000	2011-08-28 00:00:00.000
5	EXCEL 2007	2011-06-05 00:00:00.000	2011-08-28 00:00:00.000

Como se puede ver el resultado es el mismo en ambos casos.

Se necesita saber cuáles de los cursos que se encuentran en la tabla CURSO de la base de datos CURSO, se encuentran almacenados en la tabla HISTORIAL de la base de datos NewDB.

```
SELECT * FROM CURSO A
WHERE A.CODIGO IN (SELECT B.CODIGO
FROM NELLA1.NewDB.dbo.HISTORIAL B)
```

The screenshot shows the Microsoft SQL Server Management Studio (SSMS) interface. In the top navigation bar, the 'Query' tab is selected. The 'Object Explorer' pane on the left shows a connection to 'NELLA1 (SQL Server 10.0.1600 - nella1\abella)'. The 'Databases' node is expanded, showing 'Databases', 'Security', 'Server Objects', 'Replication', 'Management', and 'SQL Server Agent'. The main area contains a query window titled 'SQLQuery6.sql - NE...nella1\abella (51)*' with the following T-SQL code:

```

SELECT * FROM CURSO A
WHERE A.CODIGO IN (SELECT B.CODIGO
FROM NELLA1.NewDB.dbo.HISTORIAL B)

```

Below the query window is a results grid with the following data:

	CODIGO	DESCRIPCION	FECHA_INICIO	FECHA_FIN
1	1	SQL SERVER 2008	2011-06-05 00:00:00.000	2011-08-21 00:00:00.000
2	2	INTERNET	2011-06-06 00:00:00.000	2011-08-06 00:00:00.000
3	3	ACCES	2011-06-10 00:00:00.000	2011-08-10 00:00:00.000
4	4	WORD 2007	2011-06-15 00:00:00.000	2011-08-15 00:00:00.000
5	5	EXCEL 2007	2011-06-22 00:00:00.000	2011-08-28 00:00:00.000

Criterios de selección

Los criterios de selección son los que ayudan a recuperar todos aquellos datos que cumplan con las condiciones establecidas

Operadores lógicos

Como se vio en una sección anterior, los operadores lógicos son los siguientes

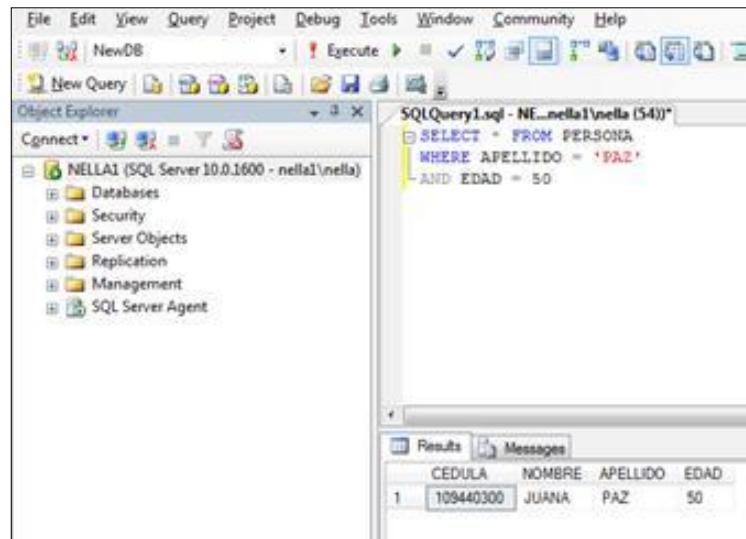
AND = y
OR = O
NOT = NO

Ejemplo AND:

```

SELECT * FROM PERSONA
WHERE APELLIDO = 'PAZ'
AND EDAD = 50

```



The screenshot shows the SQL Server Management Studio interface. In the Object Explorer, the connection to 'NELLA1' is selected. In the center pane, a query window titled 'SQLQuery1.sql' contains the following SQL code:

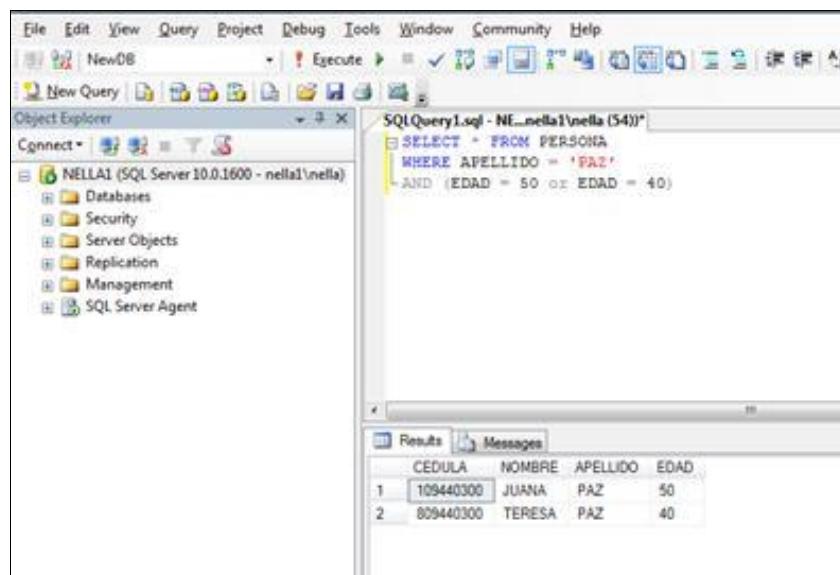
```
SELECT * FROM PERSONA
WHERE APELLIDO = 'PAZ'
AND EDAD = 50
```

In the 'Results' tab, the output is a single row:

	CEDULA	NOMBRE	APELLIDO	EDAD
1	109440300	JUANA	PAZ	50

Ejemplo de OR:

```
SELECT * FROM PERSONA
WHERE APELLIDO = 'PAZ'
AND (EDAD = 50 OR EDAD = 40)
```



The screenshot shows the SQL Server Management Studio interface. In the Object Explorer, the connection to 'NELLA1' is selected. In the center pane, a query window titled 'SQLQuery1.sql' contains the following SQL code:

```
SELECT * FROM PERSONA
WHERE APELLIDO = 'PAZ'
AND (EDAD = 50 OR EDAD = 40)
```

In the 'Results' tab, the output shows two rows:

	CEDULA	NOMBRE	APELLIDO	EDAD
1	109440300	JUANA	PAZ	50
2	809440300	TERESA	PAZ	40

Ejemplo NOT:

El NOT es un operador que a diferencia de los otros dos debe ir acompañado por IN, EXISTS o BETWEEN

Ejemplo NOT IN:

```
SELECT * FROM PERSONA  
WHERE EDAD NOT IN (50,40,25)
```

Muestra todas las personas en la que las edades sean diferentes de 50, 40 y 25.

The screenshot shows the SSMS interface. In the Object Explorer, a connection to 'NELLAI1' is selected. In the center pane, a query window titled 'SQLQuery1.sql - NE...nella1\abella (54)' contains the following SQL code:

```
SELECT * FROM PERSONA  
WHERE EDAD NOT IN (50,40,25)
```

In the bottom right pane, the 'Results' tab is active, displaying the following data:

	CEDULA	NOMBRE	APELLIDO	EDAD
1	309440300	KAREN	VARGAS	20
2	409440300	MARIO	MORA	18
3	509440300	MARIA	JUAREZ	20
4	519440300	JULIA	DIAZ	20
5	549440300	MARIA	JUAREZ	20
6	659440300	JUAN	PEREZ	18
7	909440300	JOSE	ARGUEDAZ	15

Ejemplo NOT BETWEEN:

```
SELECT * FROM PERSONA  
WHERE EDAD NOT BETWEEN 25 AND 50
```

The screenshot shows the SSMS interface. In the Object Explorer, a connection to 'NELLAI' is selected. In the center pane, a query window titled 'SQLQuery1.sql - NE...nella1\abella (54)0*' contains the following SQL code:

```
SELECT * FROM PERSONA  
WHERE EDAD NOT BETWEEN 25 AND 50
```

The results grid below shows the output of the query:

	CEDULA	NOMBRE	APELLIDO	EDAD
1	309440300	KAREN	VARGAS	20
2	409440300	MARIO	MORA	18
3	509440300	MARIA	JUAREZ	20
4	519440300	JULIA	DIAZ	20
5	549440300	MARIA	JUAREZ	20
6	659440300	JUAN	PEREZ	18
7	909440300	JOSE	ARGUEDAZ	15

El caso de NOT EXISTS, se verá más adelante en el capítulo de sub consultas

Intervalos de valores

La cláusula BETWEEN permite la recuperación de registros mediante un intervalo de valores dado.

Ejemplo:

```
SELECT * FROM PERSONA
WHERE EDAD BETWEEN 18 AND 30
```

Muestra todas las personas que tienen edad entre los 18 y 30 incluyendo los extremos.

CEDULA	NOMBRE	APELLIDO	EDAD
1. 3094403000	KAREN	VARGAS	20
2. 4094403000	MARIO	MORA	18
3. 5094403000	MARIA	JUAREZ	20
4. 5194403000	JULIA	DIAZ	20
5. 5494403000	MARIA	JUAREZ	20
6. 6594403000	JUAN	PEREZ	18
7. 7094403000	ANA	ARIAS	25

Como se vio en la sección anterior la cláusula BETWEEN puede ir acompañada del operador NOT

Ejemplo:

```
SELECT * FROM PERSONA
WHERE EDAD NOT BETWEEN 18 AND 30
```

The screenshot shows the SSMS interface. In the Object Explorer, a connection to 'NELLA1' is selected. In the center pane, a query window titled 'SQLQuery1.sql - NE...nella1\nelly (54)*' contains the following SQL code:

```
SELECT * FROM PERSONA
WHERE EDAD NOT BETWEEN 18 AND 30;
```

The results grid below shows three rows of data:

	CEDULA	NOMBRE	APELLIDO	EDAD
1	109440300	JUANA	PAZ	50
2	809440300	TERESA	PAZ	40
3	509440300	JOSE	ARGUEDAZ	15

Operador Like

Permite encontrar coincidencias parciales entre cadenas de texto.

Utiliza meta símbolos:

% Equivale a una cadena de caracteres de longitud comprendida entre 0 y n.

'AB%' AB,ABCDE,AB497

_Equivale a un único carácter

'A_B'A B,A4B, AJB

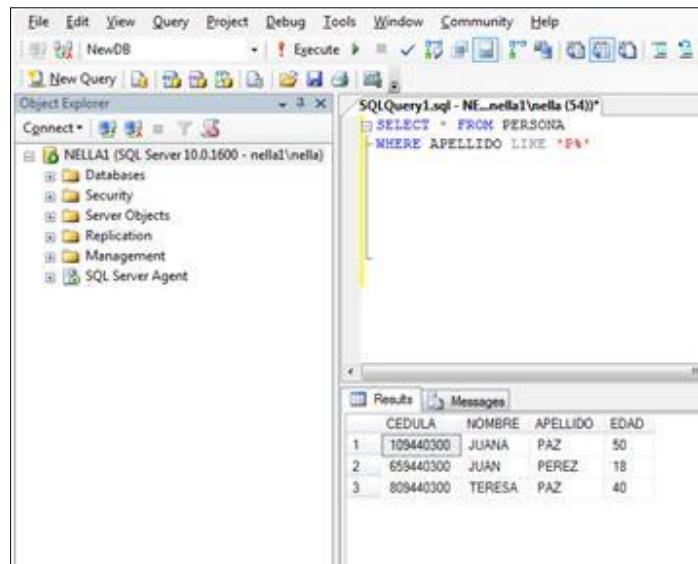
'%M[N-Q]%' Compara contra un modelo

Ejemplos:

1. Muestra las personas donde la primera letra del apellido es P

```
SELECT * FROM PERSONA
```

```
WHERE APELLIDO LIKE 'P%'
```



The screenshot shows the SQL Server Management Studio interface. In the Object Explorer, a database named 'NELLAI' is selected. In the center pane, a query window titled 'SQLQuery1.sql' contains the following SQL code:

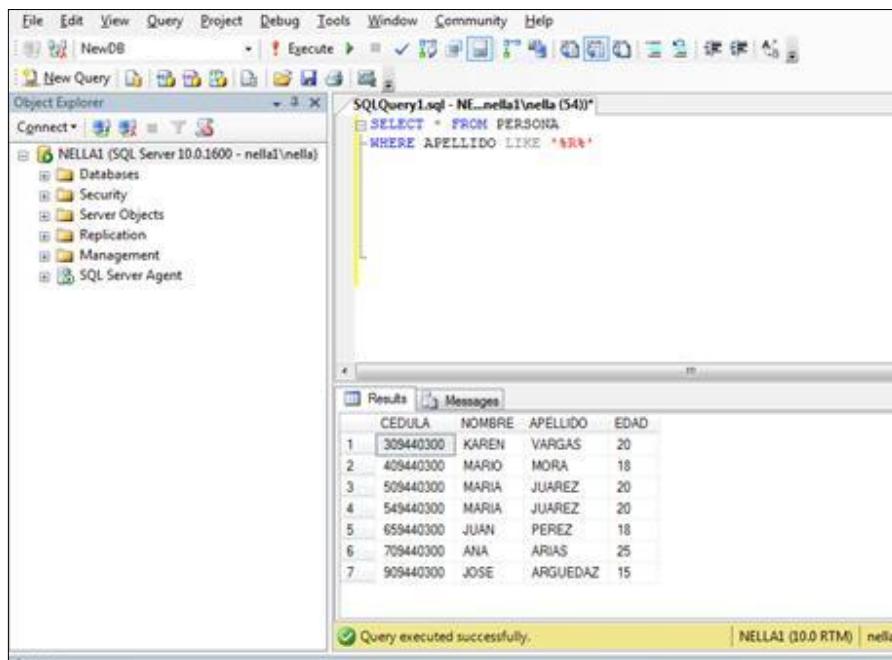
```
SELECT * FROM PERSONA
WHERE APELLIDO LIKE '%PA%'
```

The results pane below shows the output of the query:

	CEDULA	NOMBRE	APELLIDO	EDAD
1	109440300	JUANA	PAZ	50
2	659440300	JUAN	PEREZ	18
3	809440300	TERESA	PAZ	40

2. Muestra las personas que en su apellido tengan una R

```
SELECT * FROM PERSONA
WHERE APELLIDO LIKE '%R%'
```



The screenshot shows the SQL Server Management Studio interface. In the Object Explorer, a database named 'NELLAI' is selected. In the center pane, a query window titled 'SQLQuery1.sql' contains the following SQL code:

```
SELECT * FROM PERSONA
WHERE APELLIDO LIKE '%RA%'
```

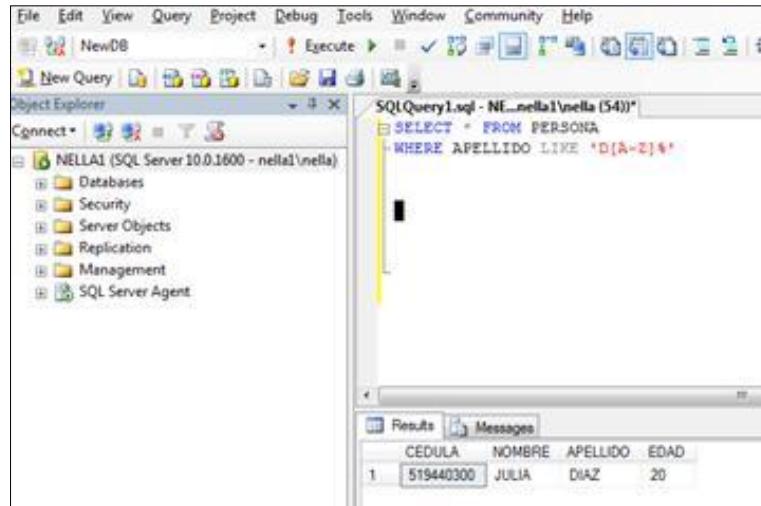
The results pane below shows the output of the query:

	CEDULA	NOMBRE	APELLIDO	EDAD
1	309440300	KAREN	VARGAS	20
2	409440300	MARIO	MORA	18
3	509440300	MARIA	JUAREZ	20
4	549440300	MARIA	JUAREZ	20
5	659440300	JUAN	PEREZ	18
6	709440300	ANA	ARIAS	25
7	909440300	JOSE	ARGUEDAZ	15

At the bottom of the results pane, a message indicates: 'Query executed successfully.'

3. Muestra las personas donde el apellido inicie con D y el resto este compuesto por letras de A o Z.

```
SELECT * FROM PERSONA  
WHERE APELLIDO LIKE 'D[A-Z]%'
```



The screenshot shows the SSMS interface with the following details:

- File Bar:** File, Edit, View, Query, Project, Debug, Tools, Window, Community, Help.
- Toolbar:** Standard toolbar with icons for NewDB, NewQuery, Save, Print, etc.
- Object Explorer:** Shows the connection to NELLA1 (SQL Server 10.0.1600 - nella1\abella) with nodes: Databases, Security, Server Objects, Replication, Management, and SQL Server Agent.
- Query Editor:** Title bar says "SQLQuery1.sql - NELLA1\abella (54)". The query is:

```
SELECT * FROM PERSONA  
WHERE APELLIDO LIKE 'D[A-Z]%'
```
- Results Grid:** Shows the results of the query with columns CEDULA, NOMBRE, APELLIDO, and EDAD. One row is displayed: 519440300, JULIA, DIAZ, 20.

Operador In

Este operador devuelve los registros que se encuentran dentro de una lista de valores dada.

Ejemplos:

1. Muestra todas las personas que tienen 25 o 45 o 50 años.

```
SELECT * FROM PERSONA  
WHERE EDAD IN (25,45,50)
```

```
File Edit View Query Project Debug Tools Window Community Help
NewDB Execute
Object Explorer
Connect ▾ NELLA1 (SQL Server 10.0.1600 - nella1\abella)
Databases Security Server Objects Replication Management SQL Server Agent
SQLQuery1.sql - NE...nella1\abella (54)*
SELECT * FROM PERSONA
WHERE EDAD IN (25,45,50)

Results Messages
CEDULA NOMBRE APELLIDO EDAD
1 109440300 JUANA PAZ 50
2 709440300 ANA ARIAS 25
```

2. Muestra todas las personas donde el apellido sea PAZ o VARGAS.

```
SELECT * FROM PERSONA
WHERE APELLIDO IN ('PAZ','VARGAS')
```

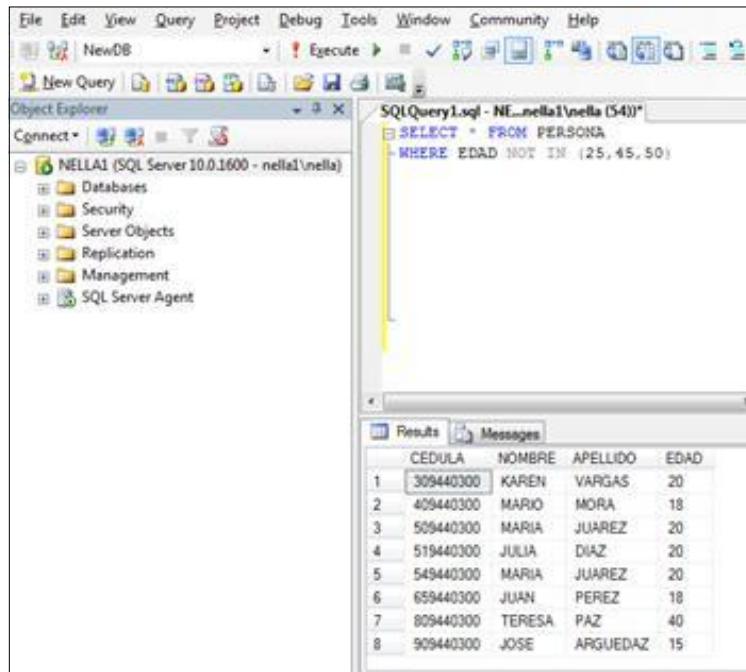
```
File Edit View Query Project Debug Tools Window Community Help
NewDB Execute
Object Explorer
Connect ▾ NELLA1 (SQL Server 10.0.1600 - nella1\abella)
Databases Security Server Objects Replication Management SQL Server Agent
SQLQuery1.sql - NE...nella1\abella (54)*
SELECT * FROM PERSONA
WHERE APELLIDO IN ('PAZ','VARGAS')

Results Messages
CEDULA NOMBRE APELLIDO EDAD
1 109440300 JUANA PAZ 50
2 309440300 KAREN VARGAS 20
3 809440300 TERESA PAZ 40
```

El operador IN se puede utilizar con el operador NOT, en este caso el resultado sería todos los registros que no se encuentren en la lista dada.

3. Muestra las personas en donde sus edades sean diferentes a 25, 45 y 50.

```
SELECT * FROM PERSONA  
WHERE EDAD NOT IN (25,45,50)
```



The screenshot shows the SSMS interface. The Object Explorer on the left shows a connection to 'NELLA1'. The 'Results' tab in the bottom pane displays the output of the query:

	CEDULA	NOMBRE	APELLIDO	EDAD
1	309440300	KAREN	VARGAS	20
2	409440300	MARIO	MORA	18
3	509440300	MARIA	JUAREZ	20
4	519440300	JULIA	DIAZ	20
5	549440300	MARIA	JUAREZ	20
6	659440300	JUAN	PEREZ	18
7	809440300	TERESA	PAZ	40
8	909440300	JOSE	ARGUEDAZ	15

4. Muestra las personas en las que el apellido sea diferente a PAZ y VARGAS

```
SELECT * FROM PERSONA  
WHERE APELLIDO NOT IN ('PAZ','VARGAS')
```

```
File Edit View Query Project Debug Tools Window Community Help
NewDB New Query Object Explorer Connect SQL Server 10.0.1600 - nella1\abella (54)*
SELECT * FROM PERSONA
WHERE APELLIDO NOT IN ('PAZ', 'VARGAS')

Results Messages
CEDULA NOMBRE APELLIDO EDAD
1 409440300 MARIO MORA 18
2 509440300 MARIA JUAREZ 20
3 519440300 JULIA DIAZ 20
4 549440300 MARIA JUAREZ 20
5 659440300 JUAN PEREZ 18
6 709440300 ANA ARIAS 25
7 909440300 JOSE ARGUEDAZ 15

Query executed successfully.
```

Cláusula Where

La cláusula WHERE permite determinar cuáles son los registros de la tabla que se mostraran en la consulta.

Ejemplos:

1. Mostrar las personas que tienen más de 40 años de edad.

```
SELECT * FROM PERSONA
WHERE EDAD > 40
```

```
File Edit View Query Project Debug Tools Window Community Help
NewDB New Query Object Explorer Connect SQL Server 10.0.1600 - nella1\abella (54)*
SELECT * FROM PERSONA
WHERE EDAD > 40

Results Messages
CEDULA NOMBRE APELLIDO EDAD
1 109440300 JUANA PAZ 50

Query executed successfully.
```

Práctica # 2 Creación de las tablas

Objetivo: Familiarizar al estudiante con la herramienta y a la vez realizar un repaso general de los puntos vistos en clase.

1. Crear las siguientes tablas en la base de datos CURSO, antes eliminar la tabla Persona de la base de datos curso por medio de un drop.

Tabla PERSONA

Campos :

CEDULA INT
NOMBRE VARCHAR(50)
APELLIDO VARCHAR(50)
EDAD INT

Tabla CURSO

Campos
CODIGO INT
DESCRIPCION VARCHAR(50)
FECHA_INICIO DATETIME
FECHA_FIN DATETIME

Tabla HISTORIAL

Campos
CEDULA INT
CODIGO INT
NOTA INT
SEDE VARCHAR(25)

2. Agregue por medio de la cláusula ALTER a la tabla PERSONA, la columna DIRECCION tipo varchar de 50.
3. Agregue por medio de la cláusula ALTER a la tabla PERSONA, el campo CEDULA como llave primaria.
4. Agregue por medio de la cláusula ALTER a la tabla CURSO, el campo CODIGO como llave primaria.
5. Agregue por medio de la cláusula ALTER al tabla HISTORIAL, los campos CEDULA y CODIGO como llave primaria

6. Agregue por medio de la cláusula ALTER a la tabla HISTORIAL, los campos CEDULA y CODIGO como llaves foráneas.
7. Inserte los siguientes valores a las tablas.

Tabla Persona

Cedula	Nombre	Apellido1	Edad	Dirección
509440300	MARIA	JUAREZ	20	HEREDIA
609440300	JUAN	PEREZ	18	ALAJUELA
709440300	ANA	ARIAS	25	SAN JOSE
809440300	TERESA	PAZ	40	HEREDIA
909440300	JOSE	ARGUEDAZ	30	PAVAS
109440300	JUANA	PAZ	50	BARVA
209440300	JULIO,	MONGE	35	ALAJUELA
309440300	KAREN	VARGAS	20	HEREDIA
409440300	MARIO	MORA	18	HEREDIA
519440300	JULIA	DIAZ	20	ALAJUELA

Tabla Curso

Código	Descripción	Fecha de inicio	Fecha Final
001	SQL SERVER 2008	20110605	20110821
002	INTERNET	20110606	20110806
003	ACCES	20110610	20110810
004	WORD 2007	20110615	20110815
005	EXCEL 2007	20110622	20110828

Tabla HISTORIAL

Cédula	Código	Nota	Sede
509440300	001	80	HEREDIA
509440300	002	85	HEREDIA
609440300	001	75	SAN JOSE
609440300	005	75	HEREDIA
609440300	003	90	SAN JOSE
709440300	003	90	SAN JOSE

809440300	002	95	SAN JOSE
809440300	001	80	SAN JOSE
809440300	003	95	SAN JOSE
909440300	002	95	SAN JOSE
909440300	001	90	HEREDIA
909440300	005	90	ALAJUELA
109440300	005	90	ALAJUELA
109440300	003	90	HEREDIA
109440300	004	80	SAN JOSE
309440300	004	75	SAN JOSE

Nota : Ver solución en [Anexo II](#)

Agrupamiento de registros y funciones agregadas

Cláusula GROUP BY

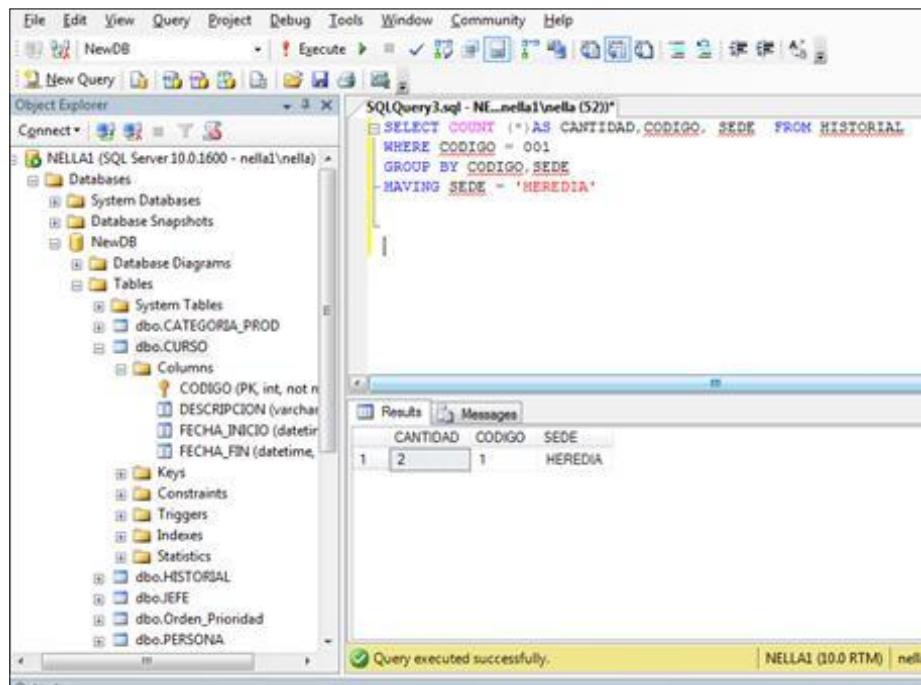
Permite agrupar filas cuando se utilizan funciones como SUM o COUNT, las cuales no toman campos línea por línea, sino resultados globales de suma de todas las filas.

En este punto se cabe mencionar la cláusula HAVING, la cual trabaja como la cláusula WHERE solo que su lógica está relacionada también con las funciones que retornan valores de grupos de registros.

Ejemplos:

1. Muestra la cantidad de personas que están estudiando SQL Server en la sede de Heredia

```
SELECT COUNT (*)AS CANTIDAD,CODIGO, SEDE FROM HISTORIAL
WHERE CODIGO = 001
GROUP BY CODIGO,SEDE
HAVING SEDE = 'HEREDIA'
```



2. Muestra la suma de las notas de las personas que están en el curso 5 en la sede de Alajuela.

```
SELECT SUM(NOTA)AS NOTA_TOTAL,CODIGO, SEDE FROM HISTORIAL  
WHERE CODIGO = 005  
GROUP BY CODIGO,SEDE  
HAVING SEDE = 'ALAJUELA'
```

```
File Edit View Query Project Debug Tools Window Community Help  
NewDB Execute Results Messages  
Object Explorer  
Connect SQL Server 10.0.1600 - nella1\abella (52)  
Databases  
System Databases  
Database Snapshots  
NewDB  
Database Diagrams  
Tables  
System Tables  
dbo.CATEGORIA_PROD  
dbo.CURSO  
Columns  
  CODIGO (PK, int, not null)  
  DESCRIPCION (varchar)  
  FECHA_INICIO (datetime)  
  FECHA_FIN (datetime)  
Keys  
Constraints  
Triggers  
Indexes  
Statistics  
dbo.HISTORIAL  
dbo.IFFE  
dbo.Orden_Prioridad  
dbo.PERSONA  
Results Messages  
NOTA_TOTAL CODIGO SEDE  
1 180 5 ALAJUELA  
Query executed successfully. NELLA1 (10.0 RTM) nella1\abella (52)
```

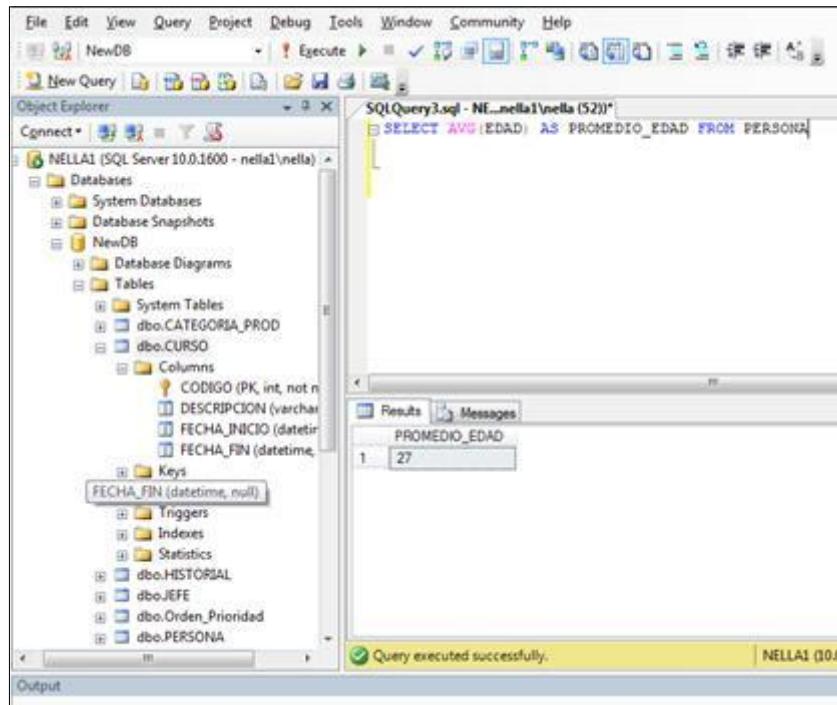
AVG Media Aritmética

Calcula la media aritmética de un conjunto de valores contenidos en un campo especificado de una consulta.

Ejemplos:

- Muestra la edad promedio de las personas que se encuentran en la tabla PERSONA

```
SELECT AVG(EDAD) AS PROMEDIO_EDAD FROM PERSONA
```



- Muestra la media de las notas del curso 5 de la sede de Alajuela.

```
SELECT AVG(NOTA)AS NOTA_TOTAL,CODIGO, SEDE  FROM HISTORIAL
WHERE CODIGO = 005
GROUP BY CODIGO,SEDE
HAVING SEDE = 'ALAJUELA'
```

```

File Edit View Query Project Debug Tools Window Community Help
New Query NewDB Execute
Object Explorer
Connect NELLAI (SQL Server 10.0.1600 - nellai\nellai)
Databases System Databases Database Snapshots NewDB Database Diagrams Tables System Tables dbo.CATEGORIA_PROD dbo.CURSO Columns CODIGO (PK, int, not null, identity(1,1)) DESCRIPCION (varchar(50)) FECHA_INICIO (datetime) FECHA_FIN (datetime) Keys Constraints Triggers Indexes Statistics dbo.HISTORIAL dbo.JEFE dbo.Orden_Prioridad dbo.PERSONA
Results Messages
NOTA_TOTAL CODIGO SEDE
1 90 5 ALAJUELA
Query executed successfully.

```

COUNT Contar Registros

La función COUNT cuenta todos aquellos registros que cumplen con una condición

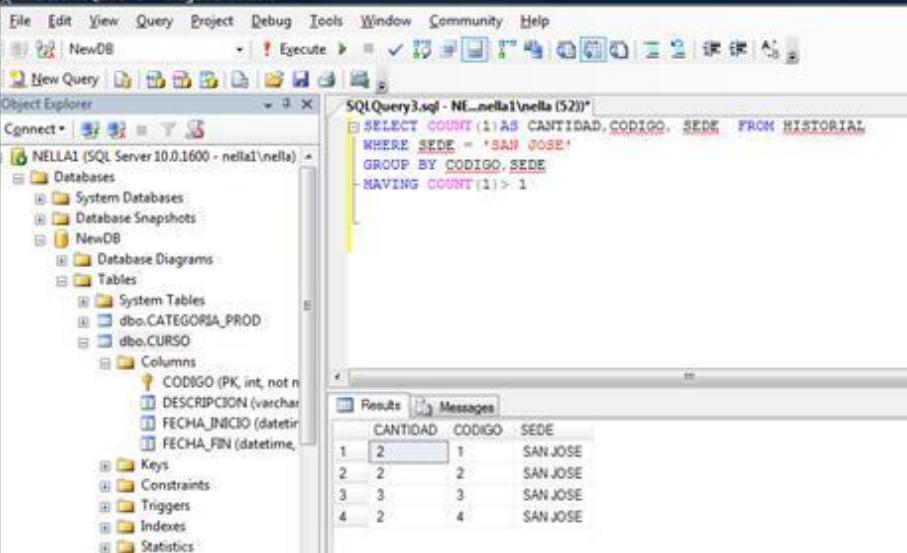
Ejemplos:

- Muestra la cantidad de personas matriculadas en cada curso en la sede de San José, siempre y cuando la cantidad sea mayor a 1.

```

SELECT COUNT(1)AS CANTIDAD,CODIGO, SEDE FROM HISTORIAL
WHERE SEDE = 'SAN JOSE'
GROUP BY CODIGO,SEDE
HAVING COUNT(1)> 1

```



The screenshot shows the SQL Server Management Studio interface. In the Object Explorer, the database 'NewDB' is selected, and under its 'Tables' node, the 'dbo.CURSO' table is expanded to show its columns: CODIGO, DESCRIPCION, FECHA_INICIO, and FECHA_FIN. In the center pane, a query window displays the following T-SQL code:

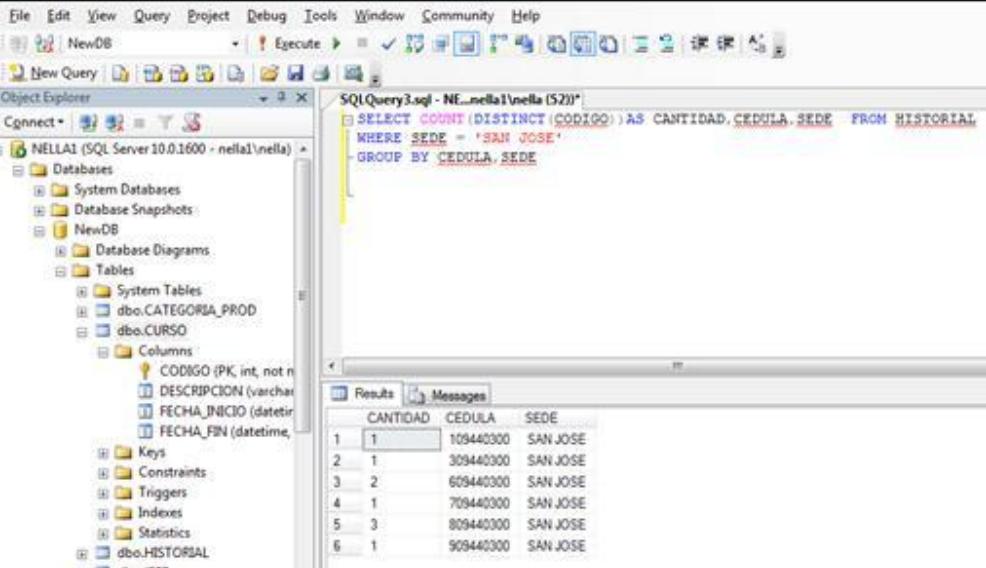
```
SELECT COUNT(1) AS CANTIDAD, CODIGO, SEDE FROM HISTORIAL
WHERE SEDE = 'SAN JOSE'
GROUP BY CODIGO, SEDE
HAVING COUNT(1) > 1
```

The results grid shows the following data:

	CANTIDAD	CODIGO	SEDE
1	2	1	SAN JOSE
2	2	2	SAN JOSE
3	3	3	SAN JOSE
4	2	4	SAN JOSE

2. Muestra la cantidad de cursos que lleva cada persona en la sede de San José

```
SELECT COUNT(DISTINCT(CODIGO))AS CANTIDAD,CEDULA,SEDE FROM HISTORIAL
WHERE SEDE = 'SAN JOSE'
GROUP BY CEDULA,SEDE
```



The screenshot shows the SQL Server Management Studio interface. In the Object Explorer, the database 'NewDB' is selected, and under its 'Tables' node, the 'dbo.HISTORIAL' table is expanded to show its columns: CANTIDAD, CEDULA, and SEDE. In the center pane, a query window displays the following T-SQL code:

```
SELECT COUNT(DISTINCT(CODIGO))AS CANTIDAD,CEDULA,SEDE FROM HISTORIAL
WHERE SEDE = 'SAN JOSE'
GROUP BY CEDULA,SEDE
```

The results grid shows the following data:

	CANTIDAD	CEDULA	SEDE
1	1	109440300	SAN JOSE
2	1	309440300	SAN JOSE
3	2	609440300	SAN JOSE
4	1	709440300	SAN JOSE
5	3	809440300	SAN JOSE
6	1	909440300	SAN JOSE

DISTINCT: Elimina los duplicados en el resultado de una consulta.

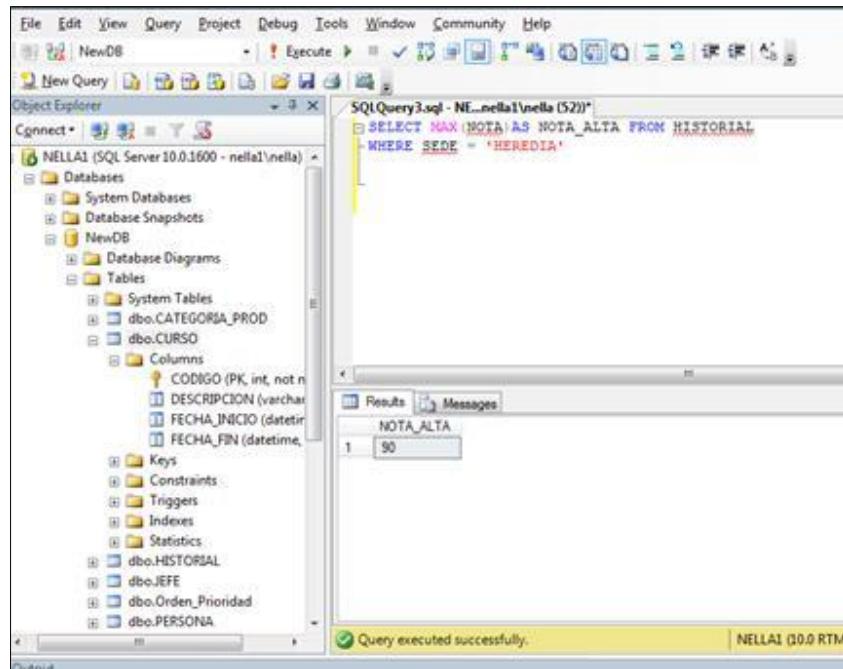
MAX, MIN Máximo y mínimo

MAX: Proporciona el valor máximo de un campo o columna dada.

Ejemplo:

1. Muestra la nota más alta en la sede de Heredia.

```
SELECT MAX(NOTA)AS NOTA_ALTA FROM HISTORIAL
WHERE SEDE = 'HEREDIA'
```

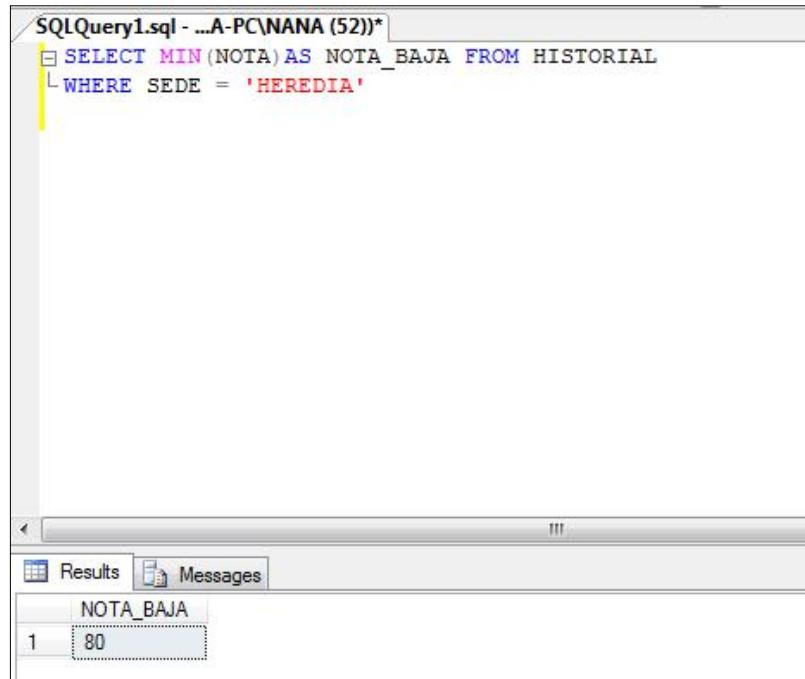


MIN: Proporciona el valor mínimo de un campo o columna.

Ejemplo:

2. Muestra la nota más baja en la sede de Heredia

```
SELECT MIN(NOTA)AS NOTA_BAJA FROM HISTORIAL
WHERE SEDE = 'HEREDIA'
```



The screenshot shows a SQL Server Management Studio (SSMS) interface. In the top pane, titled "SQLQuery1.sql - ...A-PC\NANA (52)*", there is a single line of SQL code:

```
SELECT MIN(NOTA) AS NOTA_BAJA FROM HISTORIAL  
WHERE SEDE = 'HEREDIA'
```

In the bottom pane, there are two tabs: "Results" and "Messages". The "Results" tab is selected and displays a table with one row:

	NOTA_BAJA
1	80

Stdv, stdvp Desviación estándar

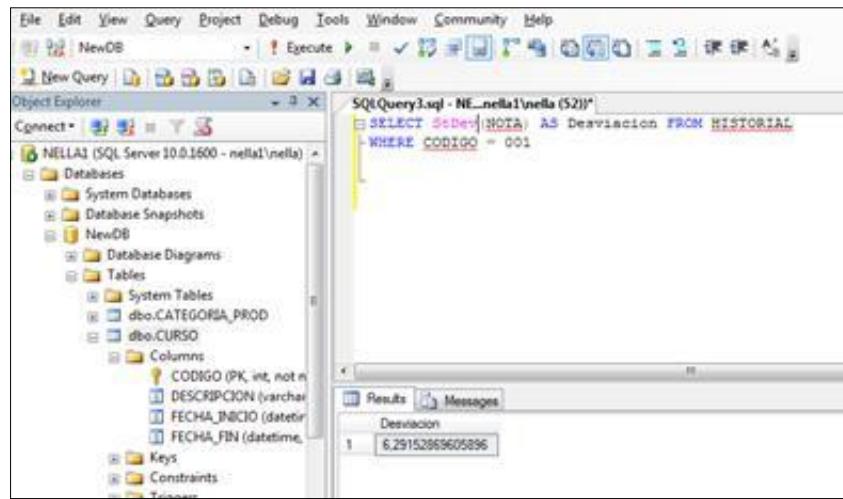
Devuelve estimaciones de la desviación estándar para la población (el total de los registros de la tabla) o una muestra de la población representada (muestra aleatoria).

StDevP evalúa una población

StDev evalúa una muestra de la población.

Ejemplo:

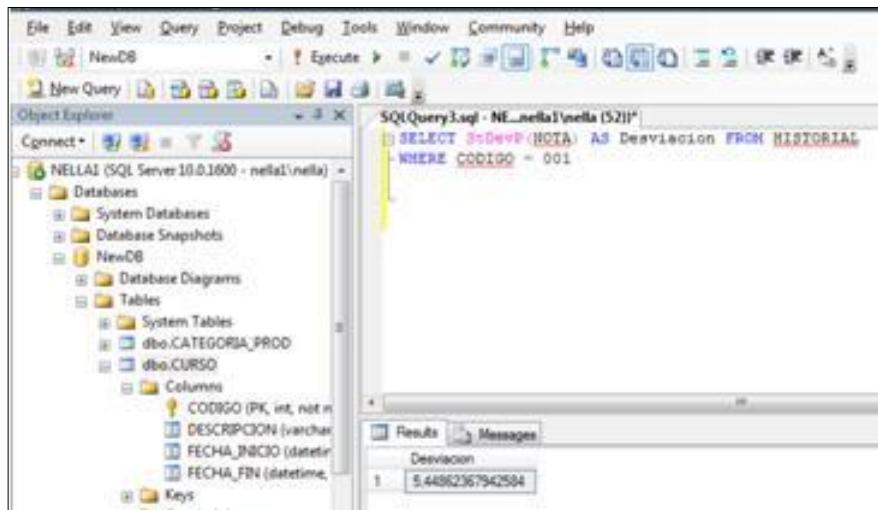
```
SELECT StDev(NOTA) AS Desviacion FROM HISTORIAL  
WHERE CODIGO = 001
```



The screenshot shows the SQL Server Management Studio interface. The Object Explorer on the left shows a database named 'NewDB' with tables 'CATEGORIA_PROD' and 'CURSO'. The 'CURSO' table has columns: CODIGO (PK, int, not null), DESCRIPCION (varchar), FECHA_INICIO (datetime), and FECHA_FIN (datetime). The Results pane on the right displays a single row with 'Desviacion' having a value of 6.29152865605896.

```
SELECT StDevP(NOTA) AS Desviacion FROM HISTORIAL
WHERE CODIGO = 001
```

**SELECT StDevP(NOTA) AS Desviacion FROM HISTORIAL
WHERE CODIGO = 001**



The screenshot shows the SQL Server Management Studio interface. The Object Explorer on the left shows a database named 'NewDB' with tables 'CATEGORIA_PROD' and 'CURSO'. The 'CURSO' table has columns: CODIGO (PK, int, not null), DESCRIPCION (varchar), FECHA_INICIO (datetime), and FECHA_FIN (datetime). The Results pane on the right displays a single row with 'Desviacion' having a value of 5.44862367942584.

```
SELECT StDevP(NOTA) AS Desviacion FROM HISTORIAL
WHERE CODIGO = 001
```

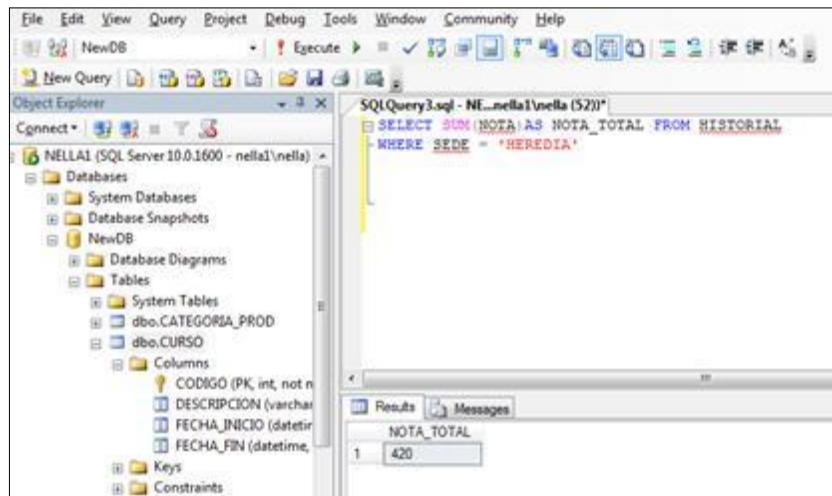
Sum Suma de valores

SUM: Proporciona la suma de un campo o columna dada.

Ejemplos:

1. Muestra la suma de las nota de todos los estudiantes de la sede de Heredia.

```
SELECT SUM(NOTA)AS NOTA_TOTAL FROM HISTORIAL
WHERE SEDE = 'HEREDIA'
```



Var VarP Varianza

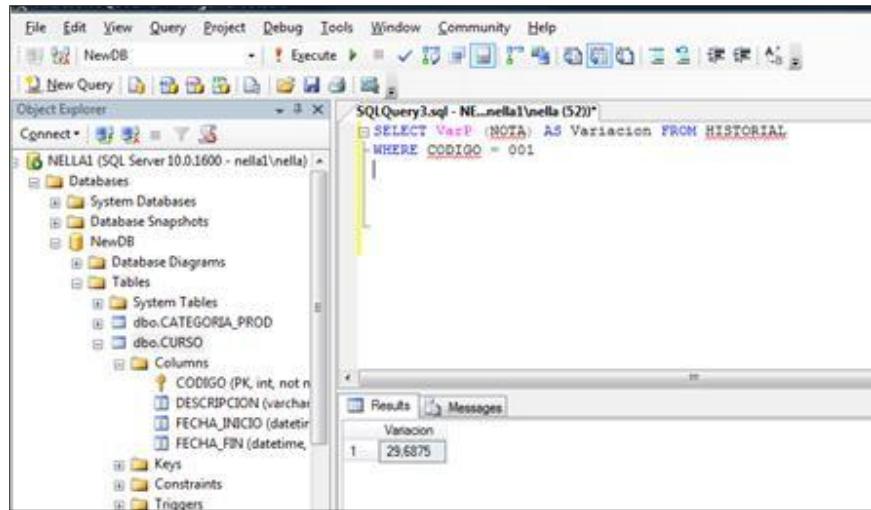
Devuelve una estimación de la varianza de una población (sobre el total de los registros) o una muestra de la población (muestra aleatoria de registros) sobre los valores de un campo.

VarP: evalúa una población

Var: evalúa una muestra de la población

Ejemplos:

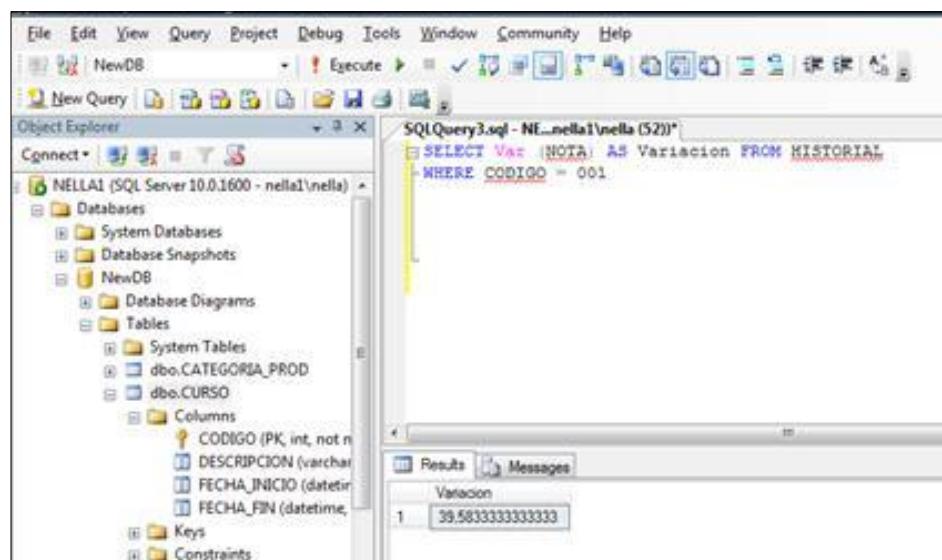
```
SELECT VarP (NOTA) AS Variacion FROM HISTORIAL
WHERE CODIGO = 001
```



The screenshot shows the SQL Server Management Studio interface. In the Object Explorer, a database named 'NewDB' is selected, which contains tables like 'CATEGORIA_PROD' and 'CURSO'. A query window titled 'SQLQuery3.sql' displays the following code and results:

```
File Edit View Query Project Debug Tools Window Community Help
New Query Connect SQL Server 2008 - NewDB Execute
Object Explorer
NELLAI (SQL Server 10.0.1600 - nella1\abella)
Databases System Databases Database Snapshots NewDB Database Diagrams Tables System Tables dbo.CATEGORIA_PROD dbo.CURSO Columns CODIGO (PK, int, not null) DESCRIPCION (varchar) FECHA_INICIO (datetime) FECHA_FIN (datetime) Keys Constraints Triggers
SQLQuery3.sql - NE...nella1\abella (52)*
SELECT VarP (NOTA) AS Variacion FROM HISTORIAL
WHERE CODIGO = 001
Results Messages
Variacion
1 29.6875
```

```
SELECT VarP (NOTA) AS Variacion FROM HISTORIAL
WHERE CODIGO = 001
```



The screenshot shows the same SQL Server Management Studio interface. The database 'NewDB' is selected in the Object Explorer. The query window 'SQLQuery3.sql' shows the same code as the previous screenshot, but the result set is much larger, indicating a full scan of the 'HISTORIAL' table.

```
File Edit View Query Project Debug Tools Window Community Help
New Query Connect SQL Server 2008 - NewDB Execute
Object Explorer
NELLAI (SQL Server 10.0.1600 - nella1\abella)
Databases System Databases Database Snapshots NewDB Database Diagrams Tables System Tables dbo.CATEGORIA_PROD dbo.CURSO Columns CODIGO (PK, int, not null) DESCRIPCION (varchar) FECHA_INICIO (datetime) FECHA_FIN (datetime) Keys Constraints Triggers
SQLQuery3.sql - NE...nella1\abella (52)*
SELECT VarP (NOTA) AS Variacion FROM HISTORIAL
WHERE CODIGO = 001
Results Messages
Variacion
1 39.5833333333333
```

Funciones de Manipulación de fecha

DATEDIFF

Devuelve el número de días, semanas, meses, años, etc.; según la parte de fecha que se desea saber, que hay entre dos fechas especificadas.

DATEDIFF(parte de fecha, fecha inicio, fecha fin)

Ejemplo:

1. Mostrar la diferencia en semanas que e encuentran entre las siguientes fechas 20101120 y 20110620.

```
SELECT DATEDIFF (WK,'20101120','20110620')
```

The screenshot shows a SQL Server Management Studio (SSMS) interface. In the top pane, there is a code editor window titled "SQLQuery1.sql - NE...nella1\nella (53)*". It contains the following SQL code:

```
select DATEDIFF (WK, '20101120', '20110620')
```

In the bottom pane, there are two tabs: "Results" and "Messages". The "Results" tab is selected and displays the following output:

	(No column name)
1	31

DATEPART

Retorna un entero que representa la parte de fecha especificada de la fecha dada.

DATEPART (parte de fecha , fecha)

Ejemplo:

1. Retorna los minutos de la fecha y hora dadas

```
SELECT DATEPART (MI,'2007-10-30 12:15:32.1234567+05:10')
```

The screenshot shows a SQL query window in SSMS. The query is:

```
SELECT DATEPART (MI, '2007-10-30 12:15:32.1234567+05:10')
```

The results tab is selected, showing the output:

(No column name)
15

GETDATE

Retorna la fecha actual del sistema.

```
GETDATE()
```

Ejemplos:

1. Retorne la fecha y hora actual del sistema

```
SELECT GETDATE()
```

The screenshot shows the SSMS interface with a query window containing the SQL command `SELECT GETDATE()`. Below the query window is a results pane. The results pane has tabs for 'Results' and 'Messages'. The 'Results' tab is selected and displays a single row with one column. The column header is '(No column name)' and the data value is '2011-06-20 23:59:58.990'.

(No column name)
1 2011-06-20 23:59:58.990

2. Retorne la siguiente fecha a la fecha actual del sistema.

```
SELECT GETDATE() AS FECHA_ACTUAL, GETDATE() + 1 AS 'FECHA SIGUIENTE DIA'
```

```
SELECT GETDATE() AS FECHA_ACTUAL, GETDATE() + 1 AS 'FECHA SIGUIENTE DIA'
```

Results Messages

	FECHA_ACTUAL	FECHA SIGUIENTE DIA
1	2011-06-21 00:02:38.827	2011-06-22 00:02:38.827

3. Actualizar la fecha de inicio del curso 5, con la fecha actual del sistema
UPDATE CURSO

```
SET FECHA_INICIO = GETDATE()  
WHERE CODIGO = 5
```

```
UPDATE CURSO  
SET FECHA_INICIO = GETDATE()  
WHERE CODIGO = 5
```

Messages

(1 row(s) affected)

4. Con base a la función GETDATE (), retornar solamente la fecha del sistema sin la hora, en el siguiente formato dd/mm/aaaa.

```
Select Convert(char(10), GETDATE(), 103) as FechaUnicamente
```

The screenshot shows a SQL Server Management Studio window. The query pane contains the following T-SQL code:

```
Select Convert(char(10), GETDATE(), 103) as FechaUnicamente
```

The results pane shows the output of the query:

	FechaUnicamente
1	02/07/2011

5. Con base a la función GETDATE (), retornar solamente la hora del sistema, en el siguiente formato hh:mm:ss

```
Select Convert(char(8), GETDATE(), 108) as HoraUnicamente
```

The screenshot shows a SQL Server Management Studio window. The query pane contains the following T-SQL code:

```
Select Convert(char(8), GETDATE(), 108) as HoraUnicamente
```

The results pane shows the output of the query:

	HoraUnicamente
1	19:15:20

Porciones de fecha y hora usadas para las funciones de fecha

Porciones de fecha y hora usados para las Funciones de fecha	
Porción	Abreviatura
año	yy, yyyy
trimestre	qq, q
mes	mm,mm
día del año	dy,d
día	dd, d
semana	wk.ww
día de la semana	dw
hora	hh
minuto	mi, n
segundos	ss,s
milisegundo	ms
microsegundo	mcs
nanosegundo	ns

Tipos de datos

Tipo de datos Caracter

Tipo de datos de carácter		
Tipo de Dato	Descripción	Tamaño en bytes
Char(n)	Datos de caracteres de longitud fija hasta un máximo de 8,000 caracteres.	Longitud definida de 1 byte
Nchar(n)	Datos de caracteres Unicode de longitud fija.	Longitud definida 2 bytes
VarChar(n)	Dato de caracteres de longitud variable hasta un máximo de 8,000 caracteres.	1 byte por carácter
VarChar(max)	Datos de caracteres de longitud variable hasta un máximo de 2GB.	1 byte por carácter
nVarChar(n)	Datos de caracteres Unicode de longitud variable hasta un máximo de 8,000 caracteres	2 bytes por carácter
nVarChar(max)	Datos de caracteres Unicode de longitud variable hasta un máximo de 2GB.	2 bytes por carácter
Text	Datos de caracteres de longitud variable	1 byte por carácter
nText	Datos de caracteres Unicode de longitud variable	2 bytes por carácter
Sysname	Tipo de dato usado para nombre de tablas o columnas equivale a nvarchar(128)	2 bytes por carácter

Tipo de datos numérico

Tipo de datos numéricos		
Tipo de dato	Descripción	Tamaño en bytes
Bit	Datos enteros con valor 1 ó 0	1 bit
Tinyint	Enteros desde 0 a 255	1 byte
Smallint	Enteros desde -32,768 a 32,767	2 bytes
Int	Enteros desde -2,147,483,648 a 2,147,483,647	4 bytes
Bigint	Enteros desde -2 ^ 63 a 2 ^ 63-1	8 bytes
Decimal or Numeric	Números de precisión exacta hasta -10 ^ 38 + 1	Varia acorde con el tamaño
Money	Números desde -2 ^ 63 a 2 ^ 63,	8 bytes
SmallMoney	Números desde -214,748.3648 a +214,748.3647	4 bytes
Float	Números con precisión de coma flotante desde -1.79E + 308 a 1.79E + 308, dependiendo del bit de precisión	4 o 8 bytes
Real	Flotante con 24-bit de precisión	4 bytes

Tipos de dato de fecha y hora

Tipo de datos Fecha/Hora		
Tipo de Dato	Descripción	Tamaño en bytes
Datetime	Valores de fecha y hora desde Enero 1, 1553 (Calendario Juliano), terminando en Diciembre 31, 9999	8 bytes
Smalldatetime	Valores de fecha y hora desde Enero 1, 1900, terminando en Junio 6, 2079.	4 bytes
DateTime2()	Valores de fecha y hora Enero 1, 0001 terminando en Diciembre 31, 9999 (Calendario Gregoriano), Precisión variable desde 01 segundos a 100 nanosegundos	6–8 bytes dependiendo de la precisión
Date	Valores de fecha y hora Enero 1, 0001 terminando Diciembre 31, 9999 (Calendario Gregoriano)	3 bytes
Time(2)	Valores de hora, precisión variable desde .01 segundos a 100 nanosegundos.	3–5 bytes dependiendo de la precisión
Datetimeoffset	Valores de fecha y hora Enero 1, 0001 terminando en Diciembre 31, 9999 (Calendario Gregoriano), precisión variable desde .01 segundos a 100 nanosegundos, incluyendo la zona horaria.	8–10 bytes dependiendo de la precisión.

Conversión de tipos de datos

Las conversiones de tipos de datos se realizan cuando los datos de un objeto se mueven, se comparan o se combinan con los datos de otro objeto, en cuyo caso puede que sea necesario convertir los datos desde el tipo de datos de un objeto al tipo de datos de otro.

Conversión de datos implícita o automática

Las conversiones de datos implícitas son invisibles para el usuario, SQL Server convierte automáticamente desde un tipo de datos al otro sin que el usuario lo perciba.

Ejemplo:

Se declaran dos variables una con el nombre @NUM1 tipo smallint y otra @NUM tipo int, con las cuales se realiza una operación aritmética.

1. Se multiplica una variable de tipo smallint por una variable int.

```
DECLARE @NUM1 SMALLINT;
DECLARE @NUM2 INT;
SET @NUM1 = 2
SET @NUM2 = 45
```

```
SELECT @NUM1*@NUM2 AS PRODUCTO
```

PRODUCTO
1 90

En este caso el dato smallint, pasa a ser implícitamente int.

2. Se una variable tipo int a una variable tipo smallint.

```
DECLARE @NUM1 SMALLINT;
DECLARE @NUM2 INT;
SET @NUM1 = 32767
SET @NUM2 = 2147443600

SELECT @NUM1+@NUM2 AS RESULTADO
```

	RESULTADO
1	2147476367

Conversión de datos explícita

La conversión de tipo de datos explícita, requiere de funciones de conversión de datos para poder modificar el tipo de dato de una expresión dada; para estos efectos se utilizan las funciones CONVERT y CAST.

Función **CONVERT**

Cláusula

```
CONVERT (tipo_de_dato [(longitud)],expresión[,estilo])
```

Argumentos

Tipo_de_dato: Tipo de datos destino.

[(longitud)]: Parámetro opcional de los tipos de datos nchar, nvarchar, char, varchar, binary o varbinary.

Expresión: Es cualquier expresión válida de Microsoft SQL Server a la cual se le necesita cambiar el tipo de dato.

Estilo: es el estilo del formato de fecha que se utiliza para convertir datos datetime o smalldatetime en datos de cadenas de caracteres (nchar, nvarchar, char, varchar).

Ejemplo:

Mostrar la fecha actual del sistema en el siguiente formato dd/mm/aaaa.
Se ejecuta la siguiente consulta

```
SELECT CONVERT (CHAR(10), GETDATE(), 103) AS FECHA_UNICAMENTE
```

FECHA_UNICAMENTE
12/10/2011

Muestra la fecha en el formato solicitado.

Función CAST**Cláusula:**

`CAST(expresión AS tipo_de_dato)`

Argumentos

Tipo_de_dato: Tipo de datos destino.

Expresión: Es cualquier expresión válida de Microsoft SQL Server a la cual se le necesita cambiar el tipo de dato.

Ejemplo:

Mostrar la fecha del sistema si la hora.

```
SELECT CAST(GETDATE() AS CHAR(13)) AS FECHA_UNICAMENTE
```

FECHA_UNICAMENTE
Oct 12 2011

Práctica #3 Aplicación de conceptos

Objetivo: que el estudiante aplique los conocimientos aprendidos en clase

Con base a las tablas creadas en la práctica 2 aplicar las siguientes consultas

1. Muestre todas las personas que tengan una M en el nombre y vivan en HEREDIA.
2. Muestre la cantidad de cursos con los que cuenta el instituto actualmente.
3. Muestre la media de las edades de todas las personas y nombre a la columna del resultado MEDIA_EDAD.
4. Muestre las personas que tienen entre 18 y 30 años de edad
5. Muestre los cursos 001,002 y 003.
6. Muestre el promedio de las notas de la sede de Alajuela, nombre a la columna resultante NOTA_PROMEDIO.
7. Actualice la fecha final de los cursos WORD 2007 y EXCEL 2007, con la fecha del sistema.

Nota: Ver solución en [Anexo III](#)

Creación de tablas

Anteriormente se crearon tablas por medio del comando CREATE

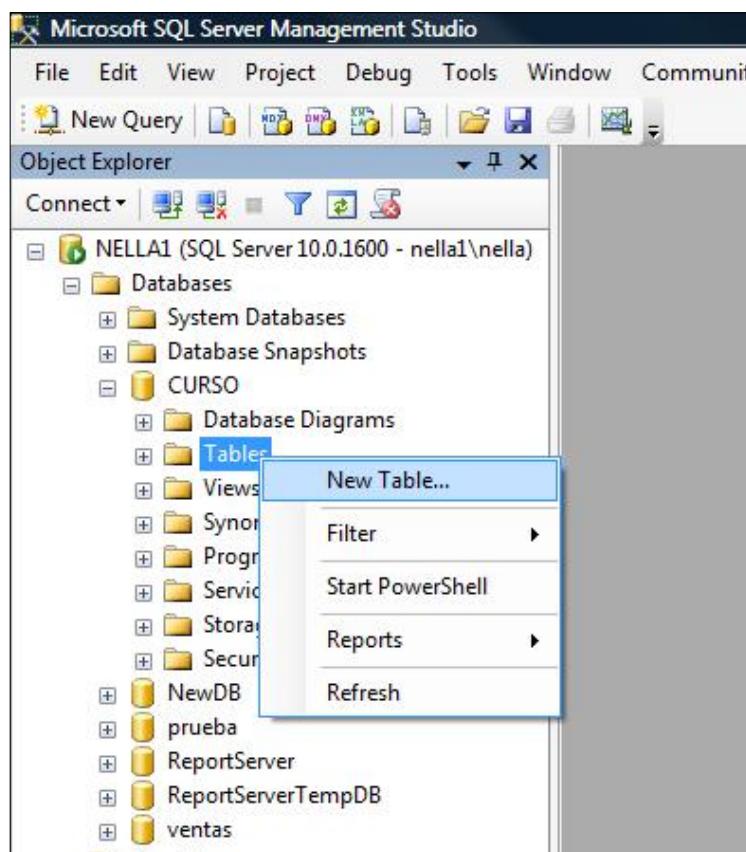
Cláusula

```
CREATE nombreObjeto
```

Ejemplo:

```
CREATE TABLE PERSONA_RESPALDO (
    CEDULA INT NOT NULL,
    NOMBRE VARCHAR(50) NOT NULL,
    APELLIDO VARCHAR(50) NOT NULL,
    EDAD INT,
    CONSTRAINT PKCURSO PRIMARY KEY(CODIGO)
)
```

Por medio del administrador o explorador de objetos también es posible crear una tabla.

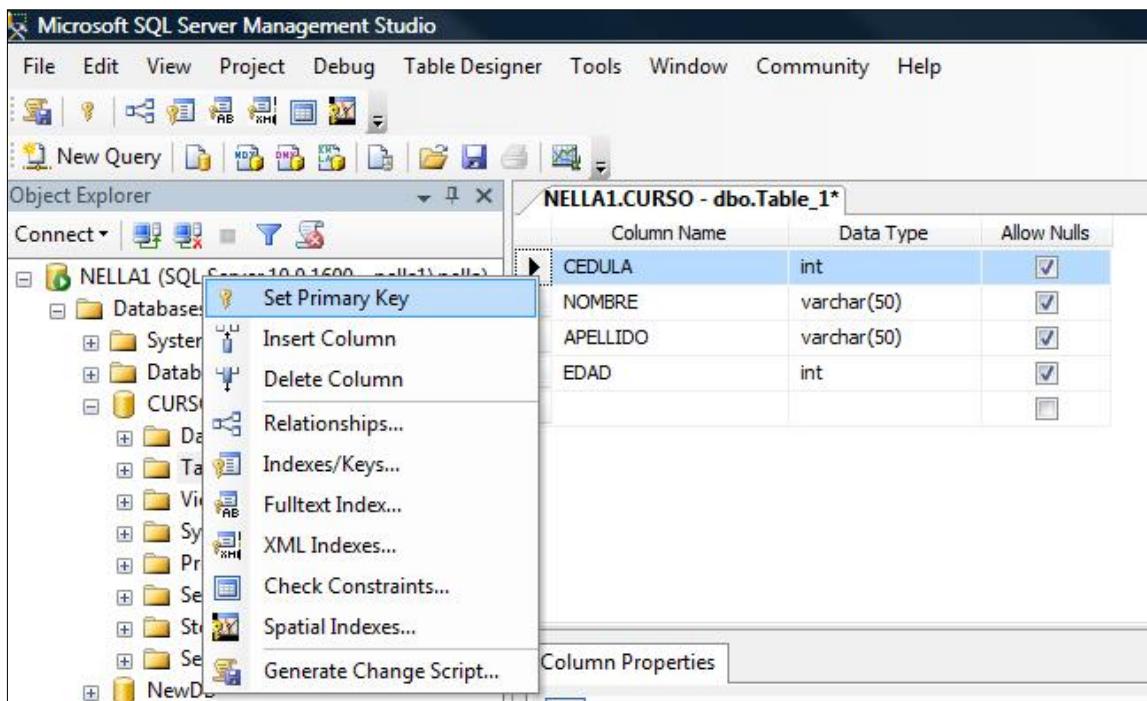


Se escoge la base de datos en la que se va a trabajar, en la carpeta TABLES se da clic derecho y se elige la opción NEW Table para ingresar los campos

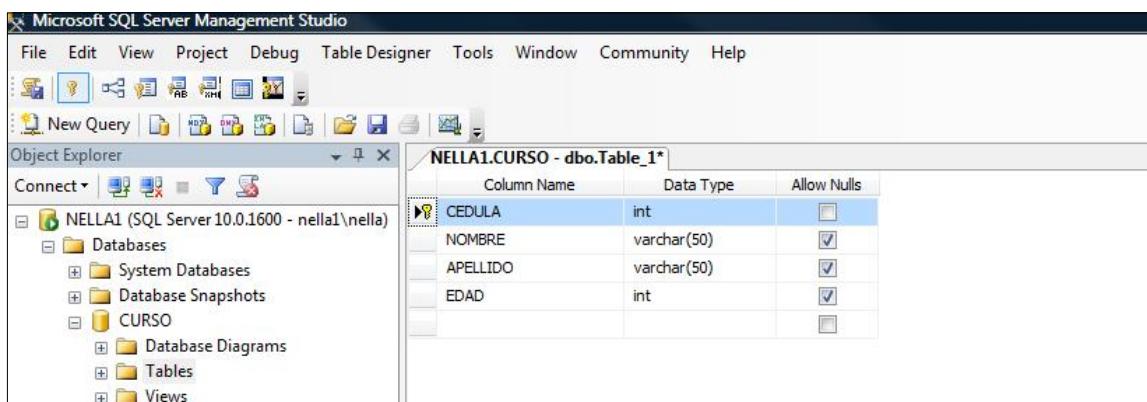
Column Name	Data Type	Allow Nulls
CEDULA	int	<input checked="" type="checkbox"/>
NOMBRE	varchar(50)	<input checked="" type="checkbox"/>
APELLIDO	varchar(50)	<input checked="" type="checkbox"/>
EDAD	int	<input checked="" type="checkbox"/>

Property	Value
DTS-published	No
Full-text Specification	No
Has Non-SQL Server Subscriber	No
Identity Specification	No
Indexable	Yes
Is Columnset	No
Is Sparse	No
Merge-published	No
Not For Replication	No
Replicated	No
RowGuid	No
Size	4

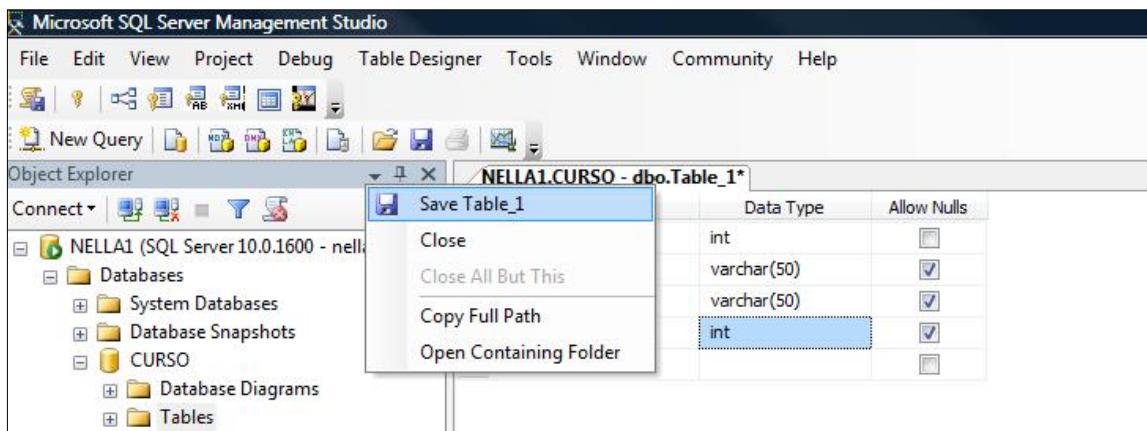
Para agregar la llave primaria se marca el campo y se da clic derecho



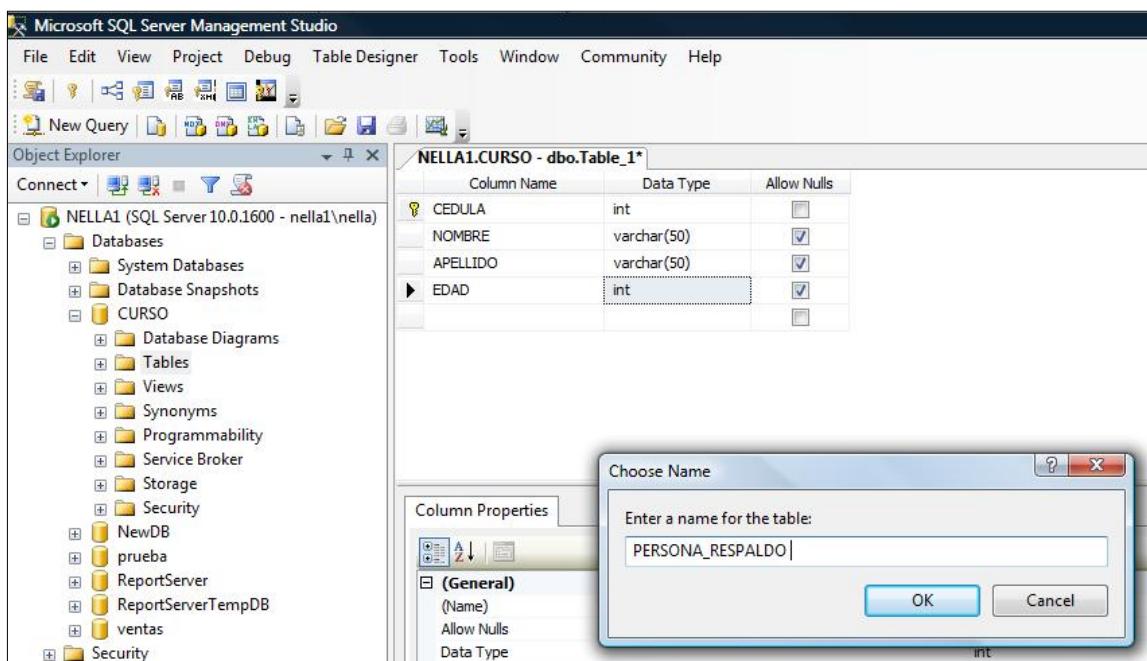
Se da clic en la opción **Set Primary Key**



Se da clic derecho en donde viene el nombre de la tabla en este caso **dbo.Table1**



Se elige la opción **Save Table_1**



Se coloca el nombre de la tabla y se da clic en **OK**

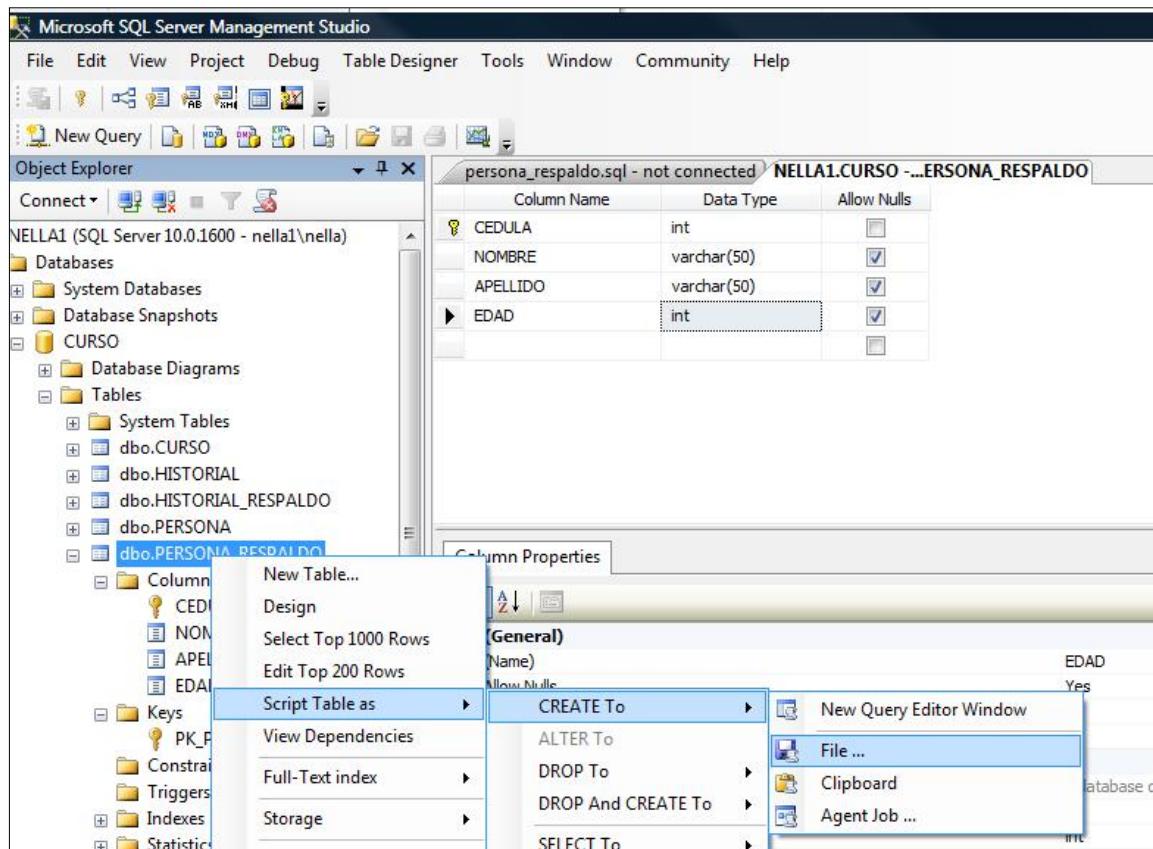
The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer on the left, under the database 'NELLA1', the table 'dbo.PERSONA_RESPALDO' is selected. The Table Designer window on the right displays the structure of this table. The table has four columns:

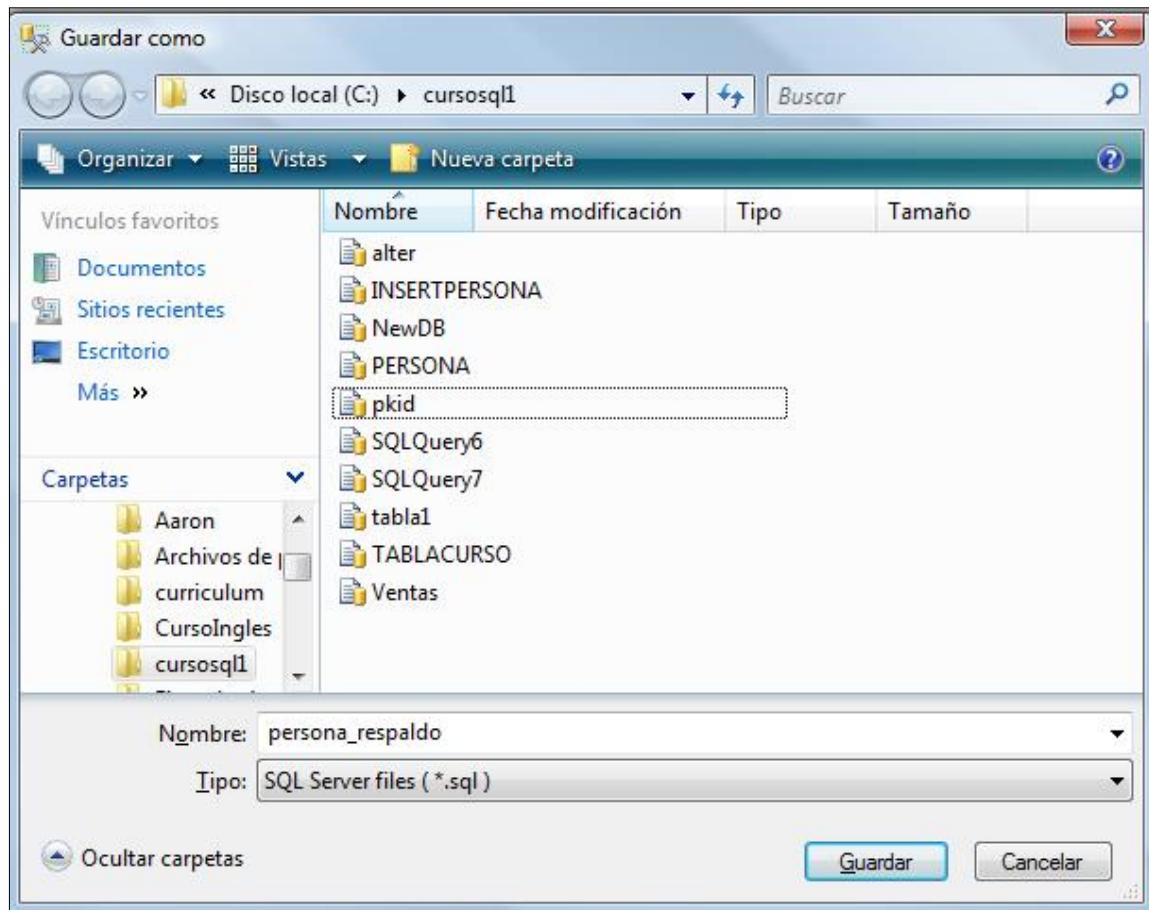
Column Name	Data Type	Allow Nulls
CEDULA	int	<input type="checkbox"/>
NOMBRE	varchar(50)	<input checked="" type="checkbox"/>
APELLIDO	varchar(50)	<input checked="" type="checkbox"/>
EDAD	int	<input checked="" type="checkbox"/>

Below the table structure, the Column Properties pane is open, showing details for the 'CEDULA' column:

- (General) (Name): CEDULA (PK, int, not null)
- Allow Nulls: Yes
- Data Type: int
- Default Value or Binding: (None)
- Table Designer: (None)

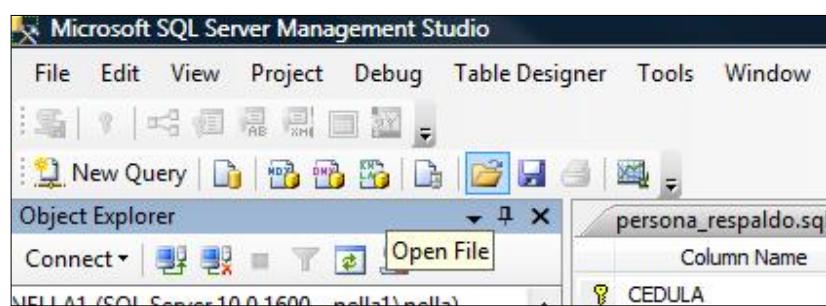
Para poder ver el código por medio del cual se creó la tabla se realiza lo siguiente



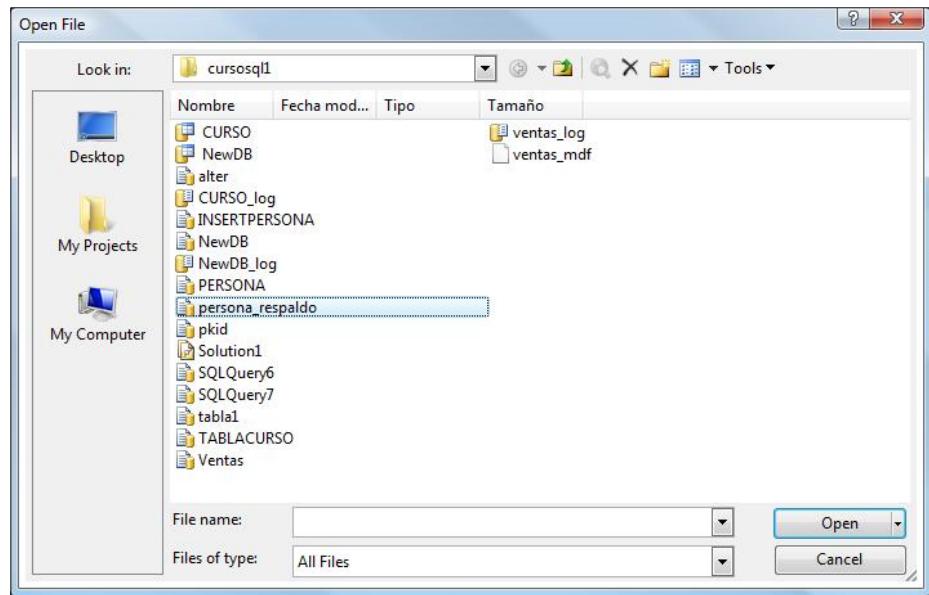


Se guarda el archivo

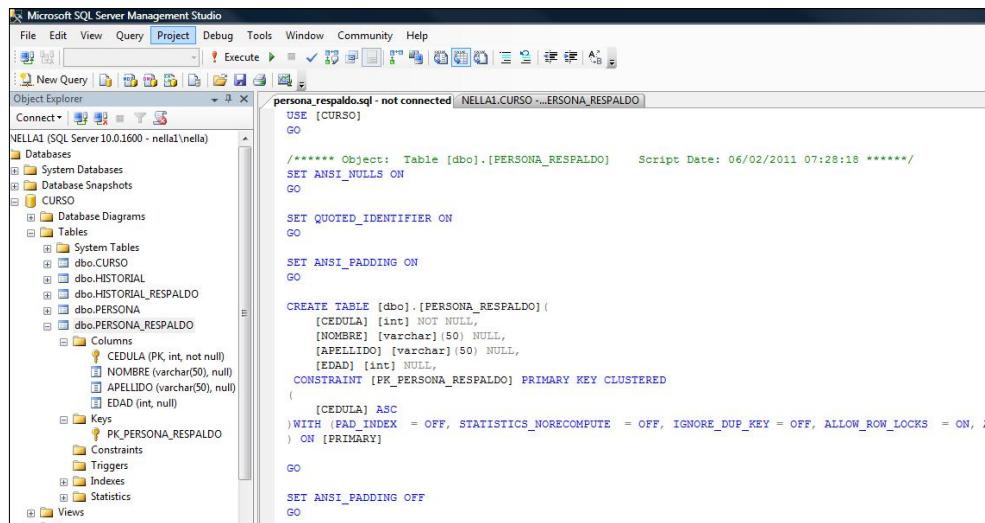
Para verlo se da click en **Open File**



Se muestra la siguiente pantalla



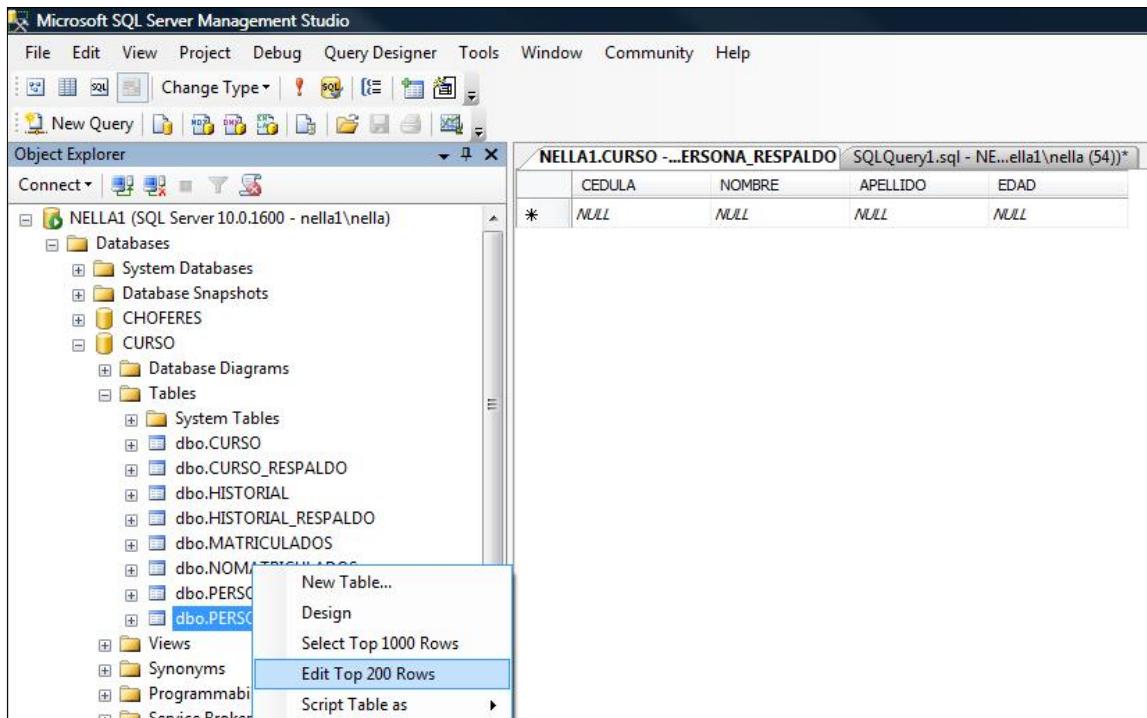
Se elige el archivo y se da clic en Open



Se puede ver el código formado al crear la tabla

Ingresar datos en una tabla

Adicional a la cláusula INSERT los datos a una tablas se pueden ingresar por medio de la opción Edit Top 200 Rows , al dar clic derecho sobre el nombre de la tabla en el explorador de objetos.



Se muestra al lado derecho de la pantalla una hoja similar a la de Excel, la cual permite el desplazamiento entre columnas o entre celda y celda por medio de la tecla TAB, el clic del mouse o las direccionales.

This screenshot shows the Microsoft SQL Server Management Studio table editor in 'Edit Top 200 Rows' mode. A new row is being added to the 'CURSO' table. The first column is marked with an asterisk (*). The data entered so far is: CEDULA (59999999), NOMBRE (marcos), APELLIDO (salazar), and EDAD (15). The cell for the fifth column is currently selected and highlighted in blue.

Al ingresar los datos se procede a cerrar la ventana y estos quedan guardados inmediatamente en la tabla.

The screenshot shows a SQL Server Management Studio window. At the top, there is a query editor pane containing the following SQL code:

```
SELECT * FROM PERSONA_RESPALDO
```

Below the query editor is a results pane. The tabs at the top of the results pane are "Results" and "Messages". The "Results" tab is selected, showing the following data in a grid:

	CEDULA	NOMBRE	APELLIDO	EDAD
1	59999999	marcos	salazar	15

Tipos de Consultas

Consultas de Actualización

Las consultas de actualización no devuelven ningún valor si no que se encargan de actualizar una tabla, en este tipo de consultas de utiliza la cláusula UPDATE.

Actualización simple

Se utiliza cuando se necesita realizar algún tipo de actualización a la tabla ya sea a todos los registros de la misma o por medio de una condición establecida con la cláusula WHERE.

Ejemplos:

Actualizar la dirección a Heredia para todas las personas

```
UPDATE PERSONA  
SET DIRECCION ='HEREDIA'
```

Es recomendable que antes de realizar una actualización se realice un SELECT de los datos que se quieren modificar.

```
SELECT * FROM PERSONA
```

```
File Edit View Query Project Debug Tools Window Community Help
CURSO Execute New Query Object Explorer Connect
NELLAI (SQL Server 10.0.1600 - nellai\nellai)
Databases System Databases CURSO NewDB prueba ReportServer ReportServerTempDB ventas Security Server Objects Replication Management SQL Server Agent
SQLQuery2.sql - NE_nella1\nellai (53)*
SELECT * FROM PERSONA
Results Messages
CEDULA NOMBRE APELLIDO EDAD DIRECCION
1 109440300 JUANA PAZ 50 BARVA
2 209440300 JULIO MONGE 35 ALAJUELA
3 309440300 KAREN VARGAS 20 HEREDIA
4 409440300 MARIO MORA 18 HEREDIA
5 509440300 MARIA JUAREZ 20 HEREDIA
6 519440300 JULIA DIAZ 20 ALAJUELA
7 609440300 JUAN PEREZ 18 ALAJUELA
8 709440300 ANA ARIAS 25 SAN JOSE
9 809440300 TERRICA PAZ 40 HEREDIA
Query executed successfully.
```

Modificar la dirección de todas aquellas personas que viven en BARVA y actualizarlas con HEREDIA.

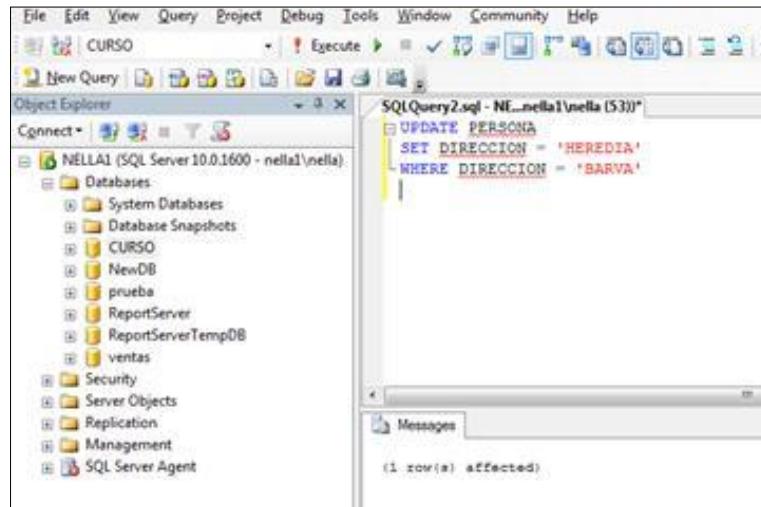
```
SELECT * FROM PERSONA
WHERE DIRECCION = 'BARVA'
```

```
File Edit View Query Project Debug Tools Window Community Help
CURSO Execute New Query Object Explorer Connect
NELLAI (SQL Server 10.0.1600 - nellai\nellai)
Databases System Databases CURSO NewDB prueba ReportServer ReportServerTempDB ventas Security Server Objects Replication Management SQL Server Agent
SQLQuery2.sql - NE_nella1\nellai (53)*
SELECT * FROM PERSONA
WHERE DIRECCION = 'BARVA'
Results Messages
CEDULA NOMBRE APELLIDO EDAD DIRECCION
1 109440300 JUANA PAZ 50 BARVA
```

A la hora de ejecutar el UPDATE, solamente se debe modificar 1 fila.

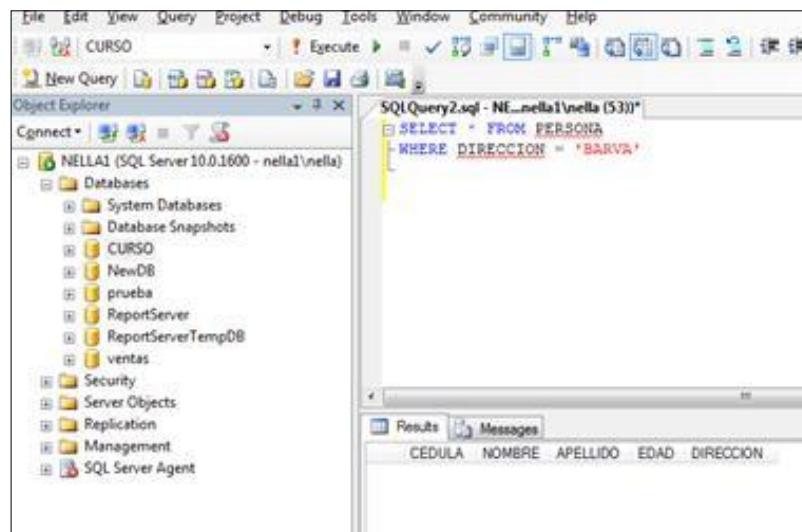
UPDATE PERSONA

```
SET DIRECCION = 'HEREDIA'
WHERE DIRECCION = 'BARVA'
```



Como se puede ver en el ejemplo solamente se actualizo un registro

```
SELECT * FROM PERSONA
WHERE DIRECCION = 'BARVA'
```



Como se puede ver ya no existen personas que tengan en la dirección BARVA.

Actualización compuesta

Las actualizaciones compuestas son: aquellas que permiten actualizar más de un campo a la vez, poseen expresiones aritméticas, utilizan funciones incorporadas o pueden ser simples como solo tener la cláusula WHERE o compuestas por operadores lógicos o de comparación.

Ejemplo:

Por medio de un SELECT, se puede visualizar como se encuentra la tabla antes de la actualización
`SELECT * FROM CURSO`

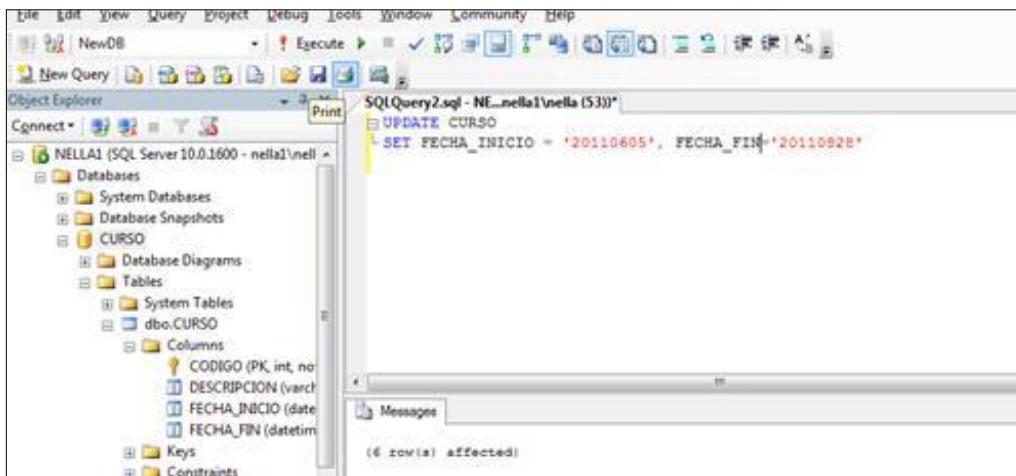
The screenshot shows the SSMS interface. On the left is the Object Explorer pane, which lists the database 'NELLA1' with its objects: Databases (System Databases, Database Snapshots, CURSO, NewDB, prueba, ReportServer, ReportServerTempDB, ventas), Security, Server Objects, Replication, Management, and SQL Server Agent. On the right is the Results pane, displaying the output of the query `SELECT * FROM CURSO`. The results show six rows of data:

CODIGO	DESCRIPCION	FECHA_INICIO	FECHA_FIN
1	SQL SERVER 2008	2011-06-05 00:00:00.000	2011-08-21 00:00:00.000
2	INTERNET	2011-06-05 00:00:00.000	2011-08-06 00:00:00.000
3	ACCES	2011-06-10 00:00:00.000	2011-08-10 00:00:00.000
4	WORD 2007	2011-06-15 00:00:00.000	2011-08-15 00:00:00.000
5	EXCEL 2007	2011-06-22 00:00:00.000	2011-08-28 00:00:00.000
6	EXCEL 2007	2011-06-21 00:00:00.000	2011-08-28 00:00:00.000

Actualizar la fecha inicio para todos los cursos en 05/06/2011 y la fecha fin para todos los cursos con 28/08/2011.

```
UPDATE CURSO
SET FECHA_INICIO = '20110605', FECHA_FIN='20110828'
```

Los campos a actualizar se separan por medio de una coma (,)



The screenshot shows the SQL Server Management Studio interface. In the Object Explorer, a database named 'NELLAI' is selected, and under its 'Tables' node, the 'CURSO' table is expanded. The 'Columns' node for the CURSO table is selected, showing columns: CODIGO (PK, int, no), DESCRIPCION (varchar), FECHA_INICIO (date), and FECHA_FIN (datetime). In the main query window, an UPDATE statement is being run:

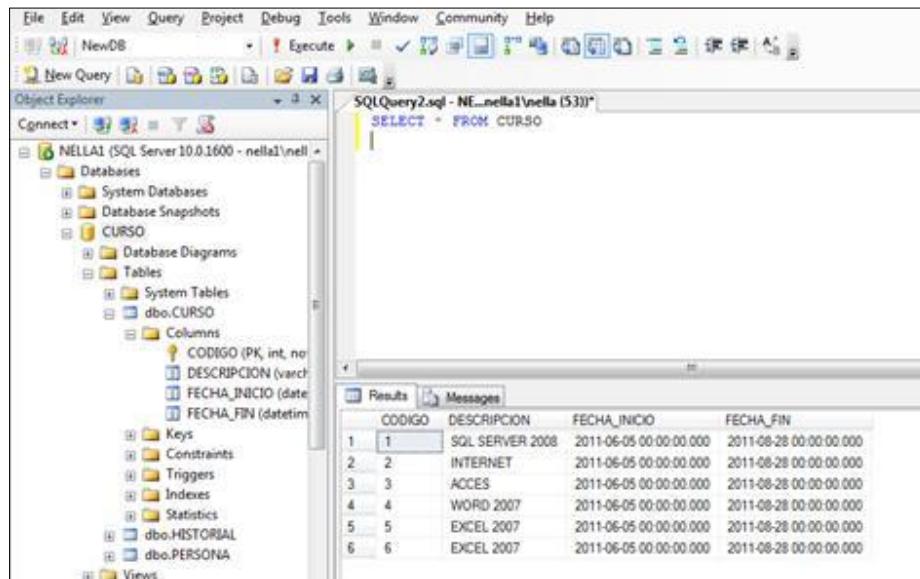
```
UPDATE CURSO
SET FECHA_INICIO = '20110605', FECHA_FIN='20110928'
```

The status bar at the bottom indicates '(6 rows affected)'.

Se modificaron 6 registros

Por medio del SELECT, se puede verificar la actualización.

SELECT * FROM CURSO



The screenshot shows the SQL Server Management Studio interface. In the Object Explorer, the 'CURSO' table is selected. In the main query window, a SELECT statement is being run:

```
SELECT * FROM CURSO
```

The results pane displays the following data:

	CODIGO	DESCRIPCION	FECHA_INICIO	FECHA_FIN
1	1	SQL SERVER 2008	2011-06-05 00:00:00.000	2011-08-28 00:00:00.000
2	2	INTERNET	2011-06-05 00:00:00.000	2011-08-28 00:00:00.000
3	3	ACCES	2011-06-05 00:00:00.000	2011-08-28 00:00:00.000
4	4	WORD 2007	2011-06-05 00:00:00.000	2011-08-28 00:00:00.000
5	5	EXCEL 2007	2011-06-05 00:00:00.000	2011-08-28 00:00:00.000
6	6	EXCEL 2007	2011-06-05 00:00:00.000	2011-08-28 00:00:00.000

Actualización con expresiones aritméticas

Ejemplos:

Actualizar la nota de todos los estudiantes para todos los cursos con 5 puntos más a la actual.

```
SELECT * FROM HISTORIAL
```

The screenshot shows the SSMS interface with the following details:

- Object Explorer:** Shows the database structure for 'NELLAI' including 'CURSO' and its tables.
- Query Editor:** Contains the query: `SELECT * FROM HISTORIAL`.
- Results Grid:** Displays the data from the HISTORIAL table.

	CEDULA	CODIGO	NOTA	SEDE
1	109440300	3	90	HEREDIA
2	109440300	4	80	SAN JOSE
3	109440300	5	90	ALAJUELA
4	309440300	4	75	SAN JOSE
5	509440300	1	80	HEREDIA
6	509440300	2	85	HEREDIA
7	609440300	1	75	SAN JOSE
8	609440300	3	90	SAN JOSE
9	609440300	5	75	HEREDIA
10	709440300	3	90	SAN JOSE
11	809440300	1	80	SAN JOSE
12	809440300	2	95	SAN JOSE
13	809440300	3	95	SAN JOSE
14	909440300	1	90	HEREDIA
15	909440300	2	95	SAN JOSE
16	909440300	5	90	ALAJUELA

```
UPDATE HISTORIAL
```

```
SET NOTA=NOTA + 5
```

The screenshot shows the SSMS interface with the following details:

- Object Explorer:** Shows the database structure for 'NELLAI' including 'CURSO' and its tables.
- Query Editor:** Contains the update query: `UPDATE HISTORIAL SET NOTA=NOTA + 5`.
- Messages Grid:** Displays the message: `(16 row(s) affected)`.

```
SELECT * FROM HISTORIAL
```

The screenshot shows the Microsoft SQL Server Management Studio (SSMS) interface. On the left, the Object Explorer pane displays the database structure of 'NELLAI' (SQL Server 10.0.1600 - nellai\nell). It shows the following hierarchy:

- Databases
 - System Databases
 - Database Snapshots
 - CURSO
 - Database Diagrams
 - Tables
 - System Tables
 - dbo.CURSO
 - Columns
 - CODIGO (PK, int, no)
 - DESCRIPCION (varchar)
 - FECHA_INICIO (date)
 - FECHA_FIN (datetime)
 - Keys
 - Constraints
 - Triggers
 - Indexes
 - Statistics
 - dbo.HISTORIAL
 - dbo.PERSONA
 - Views
 - Synonyms
 - Programmability

In the center, a query window titled 'SQLQuery2.sql - NE...nella1\nellai (53)' contains the following SQL code:

```
SELECT * FROM HISTORIAL
```

The results pane shows a table with four columns: CEDULA, CODIGO, NOTA, and SEDE. The data is as follows:

	CEDULA	CODIGO	NOTA	SEDE
1	109440300	3	95	HEREDIA
2	109440300	4	85	SAN JOSE
3	109440300	5	95	ALAJUELA
4	309440300	4	80	SAN JOSE
5	509440300	1	85	HEREDIA
6	509440300	2	90	HEREDIA
7	609440300	1	80	SAN JOSE
8	609440300	3	95	SAN JOSE
9	609440300	5	80	HEREDIA
10	709440300	3	95	SAN JOSE
11	809440300	1	85	SAN JOSE
12	809440300	2	100	SAN JOSE
13	809440300	3	100	SAN JOSE
14	909440300	1	95	HEREDIA
15	909440300	2	100	SAN JOSE
16	909440300	5	95	ALAJUELA

At the bottom of the results pane, a message states: 'Query executed successfully.'

*Actualización con expresiones aritméticas utilizando funciones incorporadas***Ejemplo:**

Suma la raíz cuadrada de 25 a la nota de todos los estudiantes.

```
SELECT * FROM HISTORIAL
```

The screenshot shows the SSMS interface with the Object Explorer on the left and the Results pane on the right. The Object Explorer shows a connection to the 'NELLAI' database, which contains the 'CURSO' schema with a 'HISTORIAL' table. The Results pane displays the following data:

	CEDULA	CODIGO	NOTA	SEDE
1	109440300	3	95	HEREDIA
2	109440300	4	85	SAN JOSE
3	109440300	5	95	ALAJUELA
4	309440300	4	80	SAN JOSE
5	509440300	1	85	HEREDIA
6	509440300	2	90	HEREDIA
7	609440300	1	80	SAN JOSE
8	609440300	3	95	SAN JOSE
9	609440300	5	80	HEREDIA
10	709440300	3	95	SAN JOSE
11	809440300	1	85	SAN JOSE
12	809440300	2	100	SAN JOSE
13	809440300	3	100	SAN JOSE
14	909440300	1	95	HEREDIA
15	909440300	2	100	SAN JOSE
16	909440300	5	95	ALAJUELA

```
UPDATE HISTORIAL
SET NOTA = NOTA + SQRT(25)
```

The screenshot shows the SSMS interface with the Object Explorer on the left and the Results pane on the right. The Object Explorer shows a connection to the 'NELLAI' database, which contains the 'CURSO' schema with a 'HISTORIAL' table. The Results pane displays the following message:

(16 row(s) affected)

SELECT * FROM HISTORIAL

The screenshot shows the SSMS interface with the following details:

- Object Explorer:** Shows the database structure for 'NELLAI'. Under the 'CURSO' node, there is a 'Tables' folder which contains 'dbo.CURSO' and 'dbo.HISTORIAL'.
- SQL Query Editor:** Contains the query: `SELECT * FROM HISTORIAL`.
- Results Grid:** Displays the data from the HISTORIAL table with columns: CEDULA, CODIGO, NOTA, and SEDE. The data is as follows:

	CEDULA	CODIGO	NOTA	SEDE
1	109440300	3	100	HEREDIA
2	109440300	4	90	SAN JOSE
3	109440300	5	100	ALAJUELA
4	309440300	4	85	SAN JOSE
5	509440300	1	90	HEREDIA
6	509440300	2	95	HEREDIA
7	609440300	1	85	SAN JOSE
8	609440300	3	100	SAN JOSE
9	609440300	5	85	HEREDIA
10	709440300	3	100	SAN JOSE
11	809440300	1	90	SAN JOSE
12	809440300	2	105	SAN JOSE
13	809440300	3	105	SAN JOSE
14	909440300	1	100	HEREDIA
15	909440300	2	105	SAN JOSE
16	909440300	5	100	ALAJUELA

- Status Bar:** Shows the message "Query executed successfully."

Actualización condicional simple

Por medio de la cláusula WHERE se permite establecer condiciones a la hora de actualizar registros en una entidad.

Para este tipo de actualización se utilizan los operadores de comparación

Ejemplo:

SELECT * FROM HISTORIAL

The screenshot shows the SSMS interface. In the Object Explorer, a database named 'NELLA1' is selected, and under its 'Tables' node, the 'CURSO' table is expanded to show its columns: CODIGO, DESCRIPCION, FECHA_INICIO, FECHA_FIN, and others. A query window titled 'SQLQuery2.sql' contains the command 'SELECT * FROM HISTORIAL'. The results grid displays 16 rows of data with columns CEDULA, CODIGO, NOTA, and SEDE.

CEDULA	CODIGO	NOTA	SEDE
1	109440300	3	HEREDIA
2	109440300	4	SAN JOSE
3	109440300	5	ALAJUELA
4	309440300	4	SAN JOSE
5	509440300	1	HEREDIA
6	509440300	2	HEREDIA
7	609440300	1	SAN JOSE
8	609440300	3	SAN JOSE
9	609440300	5	HEREDIA
10	709440300	3	SAN JOSE
11	809440300	1	SAN JOSE
12	809440300	2	SAN JOSE
13	809440300	3	SAN JOSE
14	909440300	1	HEREDIA
15	909440300	2	SAN JOSE
16	909440300	5	ALAJUELA

Como se puede ver hay estudiantes que tienen notas mayores a 100, lo cual normalmente no es correcto, si se ve el resultado mostrado por el select, se muestra que la nota más alta es 105, por lo que el update se puede realizar de 2 maneras, asignando 100 al campo nota o restando 5 al campo nota, para ambos casos sería para todos aquellos estudiantes donde la nota sea superior a 100.

UPDATE HISTORIAL

```
SET NOTA = NOTA - 5
WHERE NOTA > 100
```

The screenshot shows the SSMS interface. In the Object Explorer, a database named 'NELLA1' is selected, and under its 'Tables' node, the 'CURSO' table is expanded to show its columns. A query window titled 'SQLQuery2.sql' contains the update command. The results grid shows a message indicating 13 rows affected.

SELECT * FROM HISTORIAL

	CEDULA	CODIGO	NOTA	SEDE
1	109440300	3	100	HEREDIA
2	109440300	4	90	SAN JOSE
3	109440300	5	100	ALAJUELA
4	309440300	4	85	SAN JOSE
5	509440300	1	90	HEREDIA
6	509440300	2	95	HEREDIA
7	609440300	1	85	SAN JOSE
8	609440300	3	100	SAN JOSE
9	609440300	5	85	HEREDIA
10	709440300	3	100	SAN JOSE
11	809440300	1	90	SAN JOSE
12	809440300	2	100	SAN JOSE
13	809440300	3	100	SAN JOSE
14	909440300	1	100	HEREDIA
15	909440300	2	100	SAN JOSE
16	909440300	5	100	ALAJUELA

Después de la actualización, ningún estudiante cuenta con notas superiores a 100.

Actualización condicional compuesta

Para este tipo de actualizaciones, adicional a los operadores de comparación se utilizan los operadores lógicos

Ejemplo

SELECT * FROM HISTORIAL

The screenshot shows the SSMS interface. On the left is the Object Explorer, which is connected to the 'NELLAS' database. It displays the structure of the 'CURSO' table, including columns like CEDULA, CODIGO, NOTA, and SEDE. On the right is a query window titled 'SQLQuery2.sql - NE...nella1\abella (53)*' containing the query 'SELECT * FROM HISTORIAL'. Below the query is a results grid showing 16 rows of data.

	CEDULA	CODIGO	NOTA	SEDE
1	109440300	3	100	HEREDIA
2	109440300	4	90	SAN JOSE
3	109440300	5	100	ALAJUELA
4	309440300	4	85	SAN JOSE
5	509440300	1	90	HEREDIA
6	509440300	2	95	HEREDIA
7	609440300	1	85	SAN JOSE
8	609440300	3	100	SAN JOSE
9	609440300	5	85	HEREDIA
10	709440300	3	100	SAN JOSE
11	809440300	1	90	SAN JOSE
12	809440300	2	100	SAN JOSE
13	809440300	3	100	SAN JOSE
14	909440300	1	100	HEREDIA
15	909440300	2	100	SAN JOSE
16	909440300	5	100	ALAJUELA

This screenshot shows the same SSMS interface as above, but with a different set of data. The results grid shows four rows of data, with the third row (CEDULA 109440300, CODIGO 5) highlighted. The rest of the interface is identical to the first screenshot.

	CEDULA	CODIGO	NOTA	SEDE
1	109440300	3	100	HEREDIA
2	109440300	4	90	SAN JOSE
3	109440300	5	100	ALAJUELA
4	309440300	4	85	SAN JOSE

Se indica que para el estudiante con la cedula 109440300, la nota para el curso 5, no es 100, sino 80, por lo que debe ser actualizada de inmediato.

```
SELECT * FROM HISTORIAL
WHERE CEDULA ='109440300'
AND CODIGO = 5
```

Al ejecutar el select, nos aseguramos que solo se va a modificar lo que se solicita

```
File Edit View Query Project Debug Tools Window Community Help
NewDB Execute
Object Explorer Connect
NELLA1 (SQL Server 10.0.1600 - nella1\nell)
Databases System Databases Database Snapshots CURSO Database Diagrams Tables
System Tables dbo.CURSO Columns
CODIGO (PK, int, no)
DESCRIPCION (varchar)
FECHA_INICIO (date)

SQLQuery2.sql - NE...nella1\nell (53)*
SELECT * FROM HISTORIAL
WHERE CEDULA = '109440300'
AND CODIGO = 5

Results Messages
CEDULA CODIGO NOTA SEDE
1 109440300 5 100 ALAJUELA
```

UPDATE HISTORIAL

```
SET NOTA = 80
WHERE CEDULA = '109440300'
AND CODIGO = 5
```

```
File Edit View Query Project Debug Tools Window Community Help
NewDB Execute
Object Explorer Connect
NELLA1 (SQL Server 10.0.1600 - nella1\nell)
Databases System Databases Database Snapshots CURSO Database Diagrams Tables
System Tables dbo.CURSO Columns

SQLQuery2.sql - NE...nella1\nell (53)*
UPDATE HISTORIAL
SET NOTA = 80
WHERE CEDULA = '109440300'
AND CODIGO = 5

Messages
(1 row(s) affected)
```

Como se puede ver se actualizo solo un campo

```
SELECT * FROM HISTORIAL
```

The screenshot shows a SQL Server Management Studio window titled "SQLQuery2.sql - NE...nella1\abella (53)*". Inside the window, there is a query pane containing the following SQL code:

```
SELECT * FROM HISTORIAL
```

Below the query pane is a results grid. The grid has four columns: CEDULA, CODIGO, NOTA, and SEDE. There are four rows of data:

	CEDULA	CODIGO	NOTA	SEDE
1	109440300	3	100	HEREDIA
2	109440300	4	90	SAN JOSE
3	109440300	5	80	ALAJUELA
4	309440300	4	85	SAN JOSE

Solo se modificó el campo solicitado.

Consultas de Eliminación

Las consultas de eliminación son las que permiten borrar registros completos de una tabla.

Borrado simple

Este tipo de consulta es la que permite borrar todos los datos o registros de una tabla

Ejemplo:

DELETE FROM PERSONA

Borrado condicional simple

En este caso se utiliza la cláusula WHERE y alguno de los operadores de comparación, para establecer una condición de borrado.

Ejemplo

SELECT * FROM CURSO

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

```
SELECT * FROM CURSO
```

The results pane displays the following data:

	CODIGO	DESCRIPCION	FECHA_INICIO	FECHA_FIN
1	1	SQL SERVER 2008	2011-06-05 00:00:00.000	2011-08-28 00:00:00.000
2	2	INTERNET	2011-06-05 00:00:00.000	2011-08-28 00:00:00.000
3	3	ACCES	2011-06-05 00:00:00.000	2011-08-28 00:00:00.000
4	4	WORD 2007	2011-06-05 00:00:00.000	2011-08-28 00:00:00.000
5	5	EXCEL 2007	2011-06-05 00:00:00.000	2011-08-28 00:00:00.000
6	6	EXCEL 2007	2011-06-05 00:00:00.000	2011-08-28 00:00:00.000

Como se puede ver existe dos veces el curso Excel 2007, por lo que uno de ellos se debe eliminar

```
DELETE FROM CURSO  
WHERE CODIGO = 6
```

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

```
DELETE FROM CURSO  
WHERE CODIGO = 6
```

The messages pane displays the following output:

```
(1 row(s) affected)
```

```
SELECT * FROM CURSO
```

SELECT * FROM CURSO				
	CODIGO	DESCRIPCION	FECHA_INICIO	FECHA_FIN
1	1	SQL SERVER 2008	2011-06-05 00:00:00.000	2011-08-28 00:00:00.000
2	2	INTERNET	2011-06-05 00:00:00.000	2011-08-28 00:00:00.000
3	3	ACCES	2011-06-05 00:00:00.000	2011-08-28 00:00:00.000
4	4	WORD 2007	2011-06-05 00:00:00.000	2011-08-28 00:00:00.000
5	5	EXCEL 2007	2011-06-05 00:00:00.000	2011-08-28 00:00:00.000

El registro es eliminado

Borrado condicional compuesto

En este tipo de consulta adicional a los operadores de comparación se utilizan los operadores lógicos.

Ejemplo

```
SELECT * FROM HISTORIAL
```

	CEDULA	CODIGO	NOTA	SEDE
1	109440300	3	100	HEREDIA
2	109440300	4	90	SAN JOSE
3	109440300	5	80	ALAJUELA
4	309440300	4	85	SAN JOSE
5	509440300	1	90	HEREDIA
6	509440300	2	95	HEREDIA
7	609440300	1	85	SAN JOSE
8	609440300	3	100	SAN JOSE
9	609440300	5	85	HEREDIA
10	709440300	3	100	SAN JOSE
11	809440300	1	90	SAN JOSE
12	809440300	2	100	SAN JOSE
13	809440300	3	100	SAN JOSE
14	909440300	1	100	HEREDIA
15	909440300	2	100	SAN JOSE
16	909440300	5	100	ALAJUELA

Indican que el estudiante con cédula 609440300, se retiró del curso de Excel 2007, el cual tiene código 5, como se retiró sin terminarlo se solicita se elimine del histórico solo para este curso , ya que el continua llevando más cursos.

```
DELETE FROM HISTORIAL
WHERE CEDULA ='609440300'
AND CODIGO = 5
```

DELETED	FROM HISTORIAL
WHERE CEDULA ='	609440300'
- AND CODIGO =	5
	(1 row(s) affected)

SELECT * FROM HISTORIAL

	CEDULA	CODIGO	NOTA	SEDE
1	109440300	3	100	HEREDIA
2	109440300	4	90	SAN JOSE
3	109440300	5	80	ALAJUELA
4	309440300	4	85	SAN JOSE
5	509440300	1	90	HEREDIA
6	509440300	2	95	HEREDIA
7	609440300	1	85	SAN JOSE
8	609440300	3	100	SAN JOSE
9	709440300	3	100	SAN JOSE
10	809440300	1	90	SAN JOSE
11	809440300	2	100	SAN JOSE
12	809440300	3	100	SAN JOSE
13	909440300	1	100	HEREDIA
14	909440300	2	100	SAN JOSE
15	909440300	5	100	ALAJUELA

El registro fue eliminado.

Consulta de datos añadidos

Este tipo de consultas son las que se encargan de insertar datos en una tabla se utiliza la cláusula INSERT INTO.

Solo se puede insertar una fila a la vez, por lo que si se requiere insertar en la tabla más de un registro se deben realizar varias sentencias INSERT por separado.

Ejemplo:

Se debe insertar en la tabla Persona a Maria Arollo ya que es una nueva estudiante que se inscribió en el curso de INTERNET

En este caso se deben ejecutar dos insert, el de la tabla PERSONA y el de tabla HISTORIAL

```
INSERT INTO PERSONA
VALUES ('779440300','MARIA','ARROLLO',75,'ALAJUELA')
```

```

INSERT INTO PERSONS
VALUES ('779440300', 'MARIA', 'ARROLLO', 75, 'ALAJUELA')

```

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

```

INSERT INTO PERSONS
VALUES ('779440300', 'MARIA', 'ARROLLO', 75, 'ALAJUELA')

```

Below the query window is a "Messages" pane containing the output:

(1 row(s) affected)

```

INSERT INTO HISTORIAL (CEDULA,CODIGO,NOTA,SEDE)
VALUES ('779440300',002,00,'HEREDIA')

```

```

INSERT INTO HISTORIAL (CEDULA, CODIGO, NOTA, SEDE)
VALUES ('779440300', 002, 00, 'HEREDIA')

```

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

```

INSERT INTO HISTORIAL (CEDULA, CODIGO, NOTA, SEDE)
VALUES ('779440300', 002, 00, 'HEREDIA')

```

Below the query window is a "Messages" pane containing the output:

(1 row(s) affected)

La cláusula `INSERT INTO` se puede utilizar también con un select.

Ejemplo:

En ocasiones anteriores no se han tenido los respaldos suficientes, por lo que se solicita un respaldo de la tabla de histórico.

Para este efecto se debe crear una tabla que contenga los mismos campos de la tabla de origen.
Creación de la tabla de respaldo

```

CREATE TABLE HISTORIAL_RESPALDO
(
CEDULA INT NOT NULL,
CODIGO INT NOT NULL,
NOTA INT NOT NULL,
SEDE VARCHAR(25)
)

```

```
CREATE TABLE HISTORIAL_RESPALDO
(
    CEDULA INT NOT NULL,
    CODIGO INT NOT NULL,
    NOTA INT NOT NULL,
    SEDE VARCHAR(25)
)
```

Messages
Command(s) completed successfully.

```
INSERT INTO HISTORIAL_RESPALDO
SELECT * FROM HISTORIAL
```

```
INSERT INTO HISTORIAL_RESPALDO
SELECT * FROM HISTORIAL
```

Messages
(17 row(s) affected)

SELECT INTO

El SELECT INTO crea una nueva tabla a partir de una consulta.

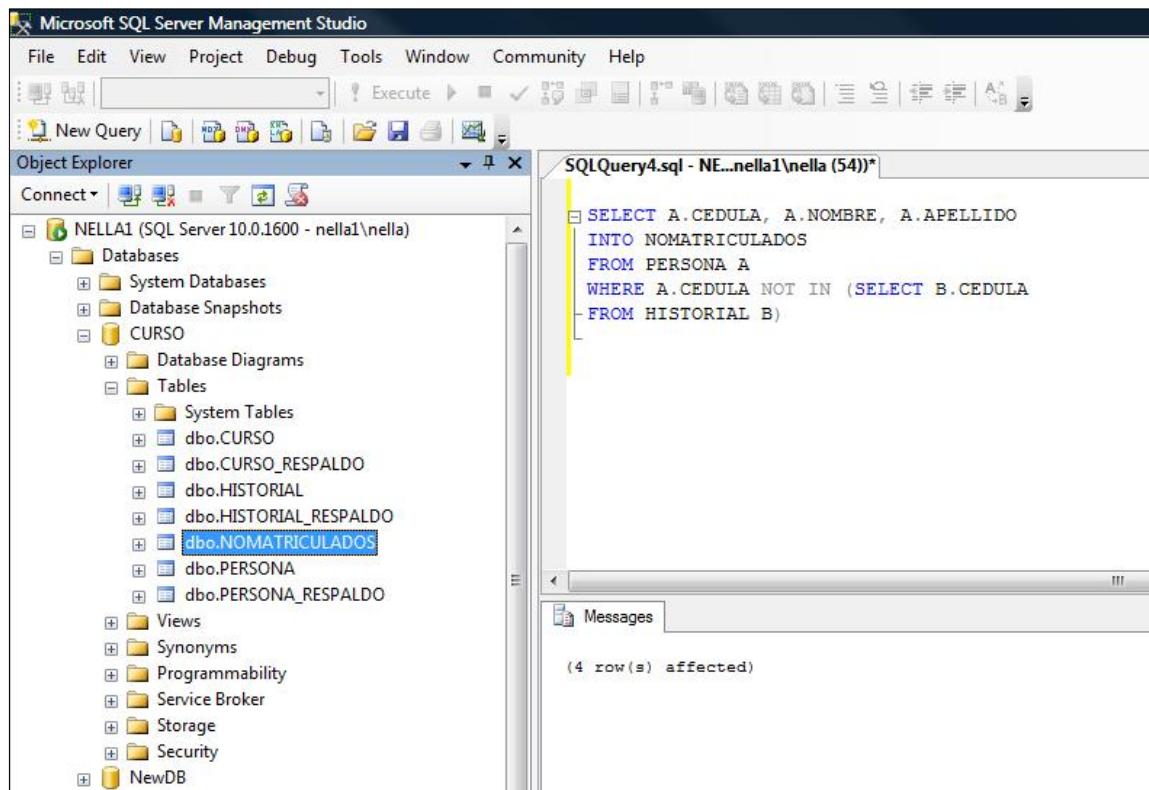
No permite crear una tabla con particiones, ya que las columnas de la tabla resultante son los campos seleccionados en la consulta que dio origen a la nueva tabla.

Los índices, las restricciones y los desencadenadores definidos en la tabla de origen no se transfieren a la nueva tabla, ni se pueden especificar en la instrucción SELECT...INTO. Si se requieren estos objetos, se deben crear después de ejecutar SELECT...INTO.

Ejemplo:

Se requiere crear una tabla con solo las personas que en este momento no se encuentran matriculadas en ningún curso.

```
SELECT A.CEDULA, A.NOMBRE, A.APELLIDO
INTO NOMATRICULADOS
FROM PERSONA A
WHERE A.CEDULA NOT IN (SELECT B.CEDULA
FROM HISTORIAL B)
```



Como se puede ver en la base de datos CURSO se creó la tabla NOMATRICULADOS con las columnas especificadas en la consulta.

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer on the left, under the database 'CURSO', there is a table named 'NOMATRICULADOS'. In the center, a query window titled 'SQLQuery4.sql' contains the following SQL code:

```
SELECT * FROM NOMATRICULADOS
```

The results pane on the right displays the data from the 'NOMATRICULADOS' table:

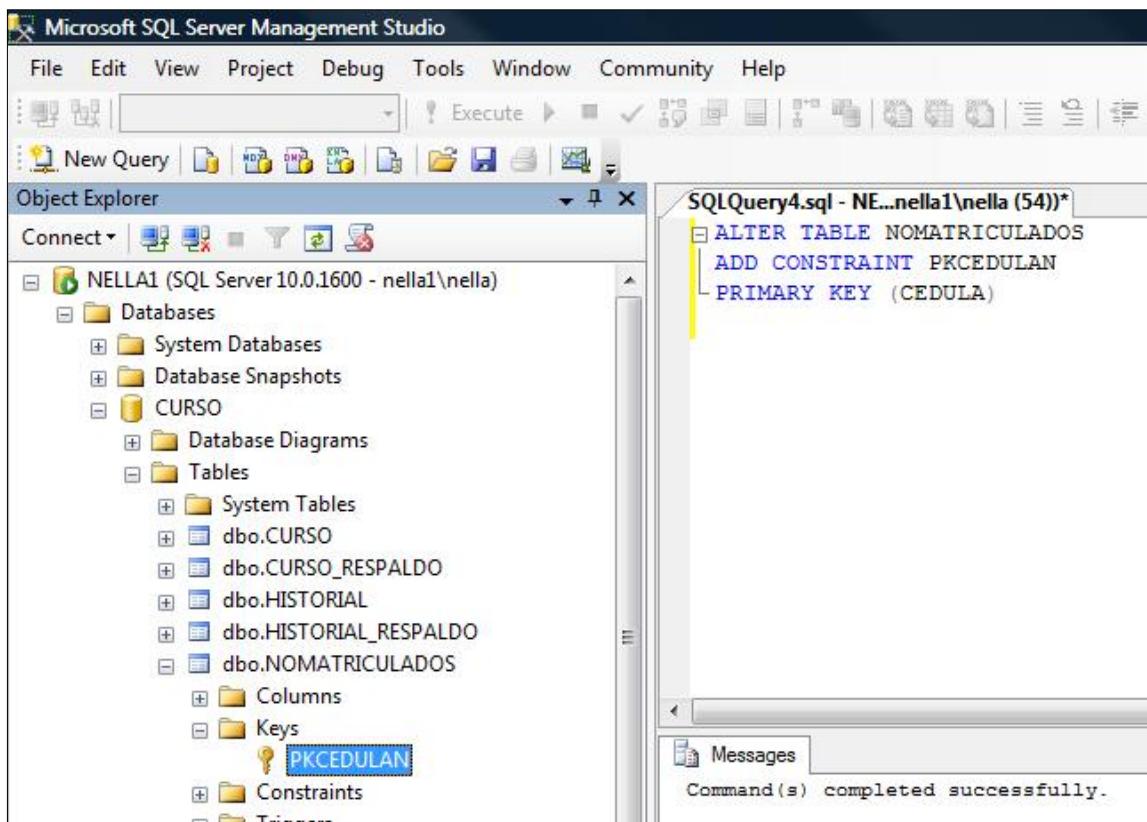
	CEDULA	NOMBRE	APELLIDO
1	209440300	JULIO	MONGE
2	409440300	MARIO	MORA
3	519440300	JULIA	DIAZ
4	779440300	MARIA	ARROLLO

En el caso de agregar algún constraint, se debe hacer por medio de un Alter o en el caso de agregar un índice por medio de un Create.

Ejemplo:

Agregar el campo CEDULA como llave primaria en la tabla NOMATRICULADOS.

```
ALTER TABLE NOMATRICULADOS
ADD CONSTRAINT PKCEDULAN
PRIMARY KEY (CEDULA)
```



Práctica # 4 Consultas

Objetivo: el estudiante reconozco los diferentes tipos de consulta que existen

Con base a la base de datos CURSO y las tablas que la componen realizar la siguiente práctica

1. Modificar la dirección de todas aquellas personas que viven en PAVAS y actualizarlas con SAN JOSE.
2. Actualice la fecha de inicio para el curso con código 2 a 06/06/2011 y la fecha fin para este mismo curso en 21/07/2011
3. Aumente en 5 todas aquellas notas que sean menores de 80 y mayores o iguales a 70
4. Se debe eliminar del historial al estudiante con cédula 779440300 , en el curso 2
5. Agregar la siguiente información a las tablas PERSONA e HISTORIAL según corresponda

Tabla PERSONA

Cédula: 78880300

Nombre: JOSE

Apellido: ARROLLO

Edad :50

Dirección: ALAJUELA

Tabla HISTORIAL

CEDULA: 78880300

CODIGO: 002

NOTA:0

SEDE:SAN JOSE

6. Crear por medio de la cláusula select into la tabla MATRICULADOS, donde se guarde la información de la cedula, nombre apellido código de curso y nombre de curso en el cual está matriculado cada persona.
7. Mostrar la cantidad de semanas que tarda el curso de semanas que tarda el curso de Access.
8. Muestre el día de la semana en el que inicia el curso de SQL.

Nota : Ver solución en el [Anexo IV](#)

La cláusula CONSTRAINT

Es una palabra clave opcional que indica el principio de una definición de una restricción PRIMARY KEY, NOT NULL, UNIQUE, FOREIGN KEY O CHECK.

Se utiliza en las instrucciones ALTER TABLE y CREATE TABLE para crear o eliminar índices.

La cláusula CONSTRAINT debe aparecer inmediatamente después del campo indexado.

Sintaxis

Para indices de campos únicos

```
CONSTRAINT nombre {PRIMARY KEY|UNIQUE|REFERENCES table externa  
[(campo externo 1,campo externo2)]}
```

Para indices de campos multiples :

```
CONSTRAINT nombre {PRIMARY KEY( primario1[primario2[,...]])|  
UNIQUE(único1[,único2[...]])|  
FOREIGN KEY(ref1[,ref2[,...]])REFERENCES table externa [(campo externo1  
,campo externo2[,...])]}
```

primario#: Es el nombre del campo o campos que forman el índice primero.

único#: Es el nombre del campo o los campos que forman el índice de clave único.

ref#: Es el nombre del campo o los campos que forman el índice externo (hacen referencia a campos de otra tabla).

tabla externa: Es el nombre de la tabla que contiene el campo o los campos referenciados en ref#.

campos externos : Es el nombre del campo o de los campos de la tabla externa especificados por ref1 ,ref2.....,refN

Tipo de índice	Descripción
UNIQUE	Impide la duplicación de claves alternativas (no principales) y asegura que se cree un índice para aumentar el rendimiento. Se permiten valores nulos. Una tabla puede tener varias restricciones UNIQUE
PRIMARY KEY	Identifica de forma exclusiva cada una de las filas; asegura que los usuarios no escriban valores duplicados y que se cree un índice para aumentar el rendimiento. No se permiten valores nulos. Permite la declaración de la llave primaria.
FOREIGN KEY	Define una columna o combinación de columnas cuyos valores coinciden con la clave principal de la misma u otra tabla. Permite la declaración de la llave foránea.
CHECK	Especifica los valores de los datos que se aceptan en una columna en función de los valores de otras columnas de la misma tabla. Permite establecer condiciones a un campo y validarlos.

NULL O NOT NULL	Son palabras claves que determinan si se permiten o no valores NULL en la columna. NULL no es estrictamente una restricción, pero puede especificarse de la misma forma que un NOT NULL
-----------------	---

Creación de Índices

Los índices se crean por medio de las instrucciones:

CREATE TABLE: A la hora de crear una tabla

Sintaxis

```

CREATE TABLE tabla
(campo1 tipo (tamaño) índice 1,
Campo2 tipo (tamaño) índice 2,...,)
CONSTRAINT nombreRestricción PRIMARY KEY (nombre de campo)
CONSTRAINT nombreRestricción FOREIGN KEY (nombre de campo)
REFERENCES tabla 1(nombre del campo de tabla1))

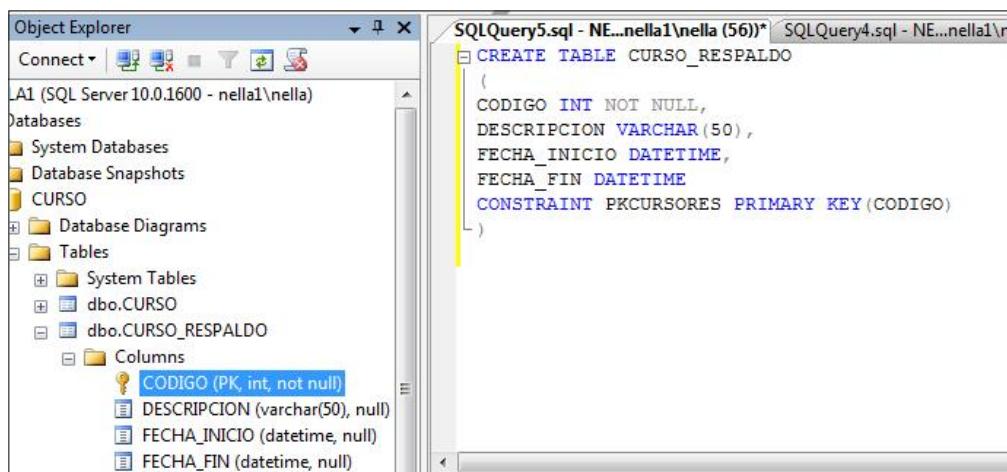
```

Ejemplo:

```

CREATE TABLE CURSO_RESPALDO
(
CODIGO INT NOT NULL,
DESCRIPCION VARCHAR(50),
FECHA_INICIO DATETIME,
FECHA_FIN DATETIME
CONSTRAINT PKCURSORES PRIMARY KEY(CODIGO)
)

```



CREATE INDEX: cuando la tabla ya ha sido definida.

```
CREATE [ UNIQUE ] [ CLUSTERED | NONCLUSTERED ]
INDEX NOMBRE DE INDICE
ON <object> ( column [ ASC | DESC ] [ ,...n ] )
```

Ejemplo:

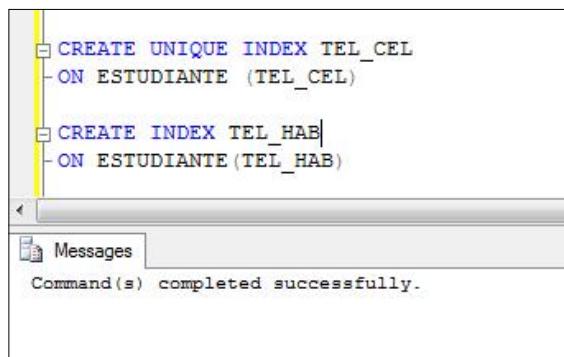
Para este ejemplo se crea en la base de datos de NewDB la tabla Estudiante

```
USE NewDB
CREATE TABLE ESTUDIANTE
(
CARNET INT NOT NULL,
NOMBRE VARCHAR (50) NOT NULL,
APELLIDO VARCHAR(50) NOT NULL,
EDAD INT NOT NULL,
DIRECCION VARCHAR(50) NULL,
TEL_CEL INT NOT NULL,
TEL_HAB INT NOT NULL,
CONSTRAINT PKCARNET PRIMARY KEY(CARNET)
)
```

Se procede a crear 4 índices

```
CREATE UNIQUE INDEX TEL_CEL
ON ESTUDIANTE (TEL_CEL)
```

```
CREATE INDEX TEL_HAB
ON ESTUDIANTE(TEL_HAB)
```



Se procede a insertar información en la tabla

The screenshot shows a table with student information and a corresponding error message dialog from Microsoft SQL Server Management Studio.

Table Data:

	CARNET	NOMBRE	APELLIDO	EDAD	DIRECCION	TEL_CEL	TEL_HAB
	12345678	MARIA	SALAS	25	HEREDIA	88888888	22222222
.	91011121	LUIS	ARIAS	20	SANJOSE	88888888	22222222
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Error Message Dialog:

No row was updated.
The data in row 2 was not committed.
Error Source: .Net SqlClient Data Provider.
Error Message: Cannot insert duplicate key row in object 'dbo.ESTUDIANTE' with unique index 'TEL_CEL'.
The statement has been terminated.

Correct the errors and retry or press ESC to cancel the change(s).

Buttons:

Aceptar

Para el segundo registro se escribe el mismo número de celular que en el registro anterior sin embargo no lo permite, debido a que este campo se declaro como único, aunque no sea llave primaria

Como se puede ver el campo TEL_HAB a pesar de ser índice si acepta que se repita

The screenshot shows the same table after attempting to insert a new row with the same value in the TEL_HAB column as the second row.

Table Data:

	CARNET	NOMBRE	APELLIDO	EDAD	DIRECCION	TEL_CEL	TEL_HAB
	12345678	MARIA	SALAS	25	HEREDIA	88888888	22222222
.	91011121	LUIS	ARIAS	20	SANJOSE	88888887	22222222
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Eliminar y Añadir campos e índices

La eliminación o agregado de campos e índices se puede realizar por medio de la instrucción ALTER TABLE

ALTER TABLE: cuando la tabla ya ha sido creada y por alguna razón en la instrucción CREATE TABLE no se incluyó el CONSTRAINT.

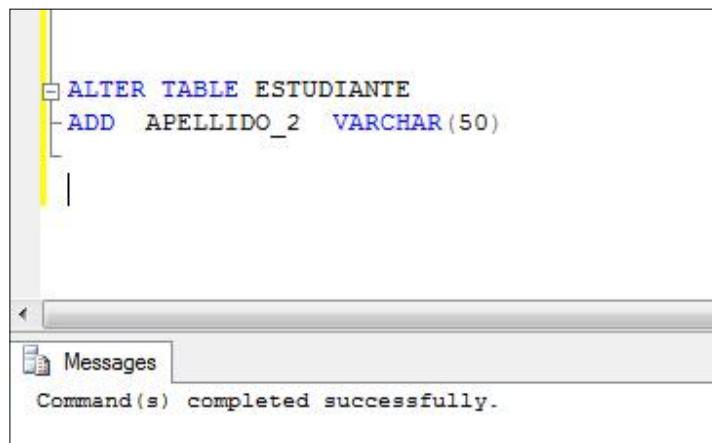
Permite agregar un nuevo campo a una tabla existente.

```
ALTER TABLE Nombre de Tabla  
ADD Nombre de Columna Tipo de  
Dato ;
```

Ejemplo :

Agrega la columna APELLIDO_2 a la tabla Estudiante

```
ALTER TABLE ESTUDIANTE  
ADD APELLIDO_2 VARCHAR(50)
```



The screenshot shows a SQL Server Management Studio window. In the main pane, there is a tree view under 'ALTER TABLE ESTUDIANTE' with the command 'ADD APELLIDO_2 VARCHAR(50)'. Below the tree, there is a 'Messages' tab. The message content is: 'Command(s) completed successfully.'

Agregar constraint por medio del ALTER**Cláusula**

```
ALTER TABLE nombre de tabla
ADD CONSTRAINT
Nombre de llave primaria PRIMARY
KEY(Campo de llave primera )
ON [PRIMARY];
```

```
ALTER TABLE nombre de tabla hija
ADD CONSTRAINT nombre de llave foránea
FOREIGN KEY (columna o columnas que
conforman la llave foránea)
REFERENCES nombre de tabla padre (Campo o
campos de llave primaria de tabla padre);
```

Modificar la columna DIRECCION, si una dirección no es digitada que automáticamente se llene con la palabra DESCONOCIDA y validar que la edad del estudiante no sea mayor a 25 años ni menor de 15

```
ALTER TABLE ESTUDIANTE
ADD CONSTRAINT DIRECCION DEFAULT 'DESCONOCIDA' FOR DIRECCION
```

```
ALTER TABLE ESTUDIANTE
ADD CONSTRAINT EDAD CHECK (EDAD BETWEEN 15 AND 25 )
```

```
ALTER TABLE ESTUDIANTE
ADD CONSTRAINT DIRECCION DEFAULT 'DESCONOCIDA' FOR DIRECCION

ALTER TABLE ESTUDIANTE
ADD CONSTRAINT EDAD CHECK (EDAD BETWEEN 15 AND 25 )

Messages
Command(s) completed successfully.
```

Se procede a incluir datos

The screenshot shows a table with columns: CARNET, NOMBRE, APELLIDO, EDAD, DIRECCION, TEL_CEL, TEL_HAB, and APELLIDO_2. The last row has values: 44444444, CARLOS, CASTRO, 26, NULL, 99999999, 66666666, RAMIREZ. A red error icon is present in the EDAD column of this row. An error dialog box from Microsoft SQL Server Management Studio is displayed, stating: "No row was updated. The data in row 3 was not committed. Error Source: .Net SqlClient Data Provider. Error Message: The INSERT statement conflicted with the CHECK constraint "EDAD". The conflict occurred in database "NewDB", table "dbo.ESTUDIANTE", column 'EDAD'. The statement has been terminated." A "Aceptar" button is at the bottom of the dialog.

	CARNET	NOMBRE	APELLIDO	EDAD	DIRECCION	TEL_CEL	TEL_HAB	APELLIDO_2
	12345678	MARIA	SALAS	25	HEREDIA	88888888	22222222	NULL
	91011121	LUIS	ARIAS	20	SANJOSE	88888887	22222222	NULL
.	44444444	CARLOS	CASTRO	26	NULL	99999999	66666666	RAMIREZ
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

En este caso se está validando el campo EDAD, a la hora de crear el constraint se indicó que el campo de EDAD debe de estar en un rango de 15 a 25, pero como se incluyó 26, muestra el mensaje de error.

The screenshot shows a table with columns: CARNET, NOMBRE, APELLIDO, EDAD, DIRECCION, TEL_CEL, TEL_HAB, and APELLIDO_2. The last row has values: 11111111, SOFI, ALL, 13, DESCONOCIDA, 78987777, 44444444, BLACK. A red error icon is present in the EDAD column of this row. An error dialog box from Microsoft SQL Server Management Studio is displayed, stating: "No row was updated. The data in row 4 was not committed. Error Source: .Net SqlClient Data Provider. Error Message: The INSERT statement conflicted with the CHECK constraint "EDAD". The conflict occurred in database "NewDB", table "dbo.ESTUDIANTE", column 'EDAD'. The statement has been terminated." A "Aceptar" button is at the bottom of the dialog.

	CARNET	NOMBRE	APELLIDO	EDAD	DIRECCION	TEL_CEL	TEL_HAB	APELLIDO_2
	12345678	MARIA	SALAS	25	HEREDIA	88888888	22222222	NULL
	91011121	LUIS	ARIAS	20	SANJOSE	88888887	22222222	NULL
.	44444444	CARLOS	CASTRO	25	DESCONOCIDA	99999999	66666666	RAMIREZ
.	11111111	SOFI	ALL	13	NULL	78987777	44444444	BLACK
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

En este otro caso se incluyó una edad de 13 que tampoco cumple con el rango dado

	CARNET	NOMBRE	APELLIDO	EDAD	DIRECCION	TEL_CEL	TEL_HAB	APELLIDO_2
	12345678	MARIA	SALAS	25	HEREDIA	88888888	22222222	NULL
	91011121	LUIS	ARIAS	20	SANJOSE	88888887	22222222	NULL
	44444444	CARLOS	CASTRO	25	DESCONOCIDA	99999999	66666666	RAMIREZ
*	11111111	SOFI	ALL	15	DESCONOCIDA	78987777	44444444	BLACK
	22222222	JOSE	LOPEZ	17	DESCONOCIDA	87888788	26666666	LOPEZ
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Como se puede ver en los últimos tres registros incluidos no se ingreso información en el campo DIRECCION por lo que inserto por defecto DESCONOCIDA.

Borrado de constraint por medio de Alter

```
ALTER TABLE nombre de tabla
DROP CONSTRAINT nombre del constraint;
```

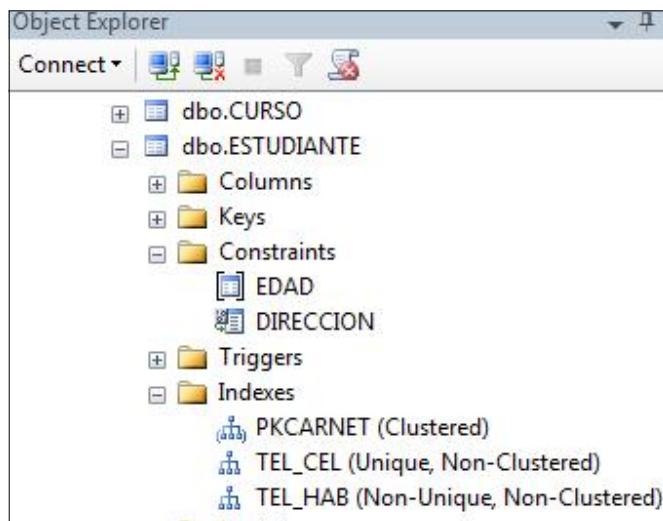
Borrado de un índice

Los índices se eliminan por medio del DROP

```
DROP INDEX nombre de índice
ON nombre de tabla
```

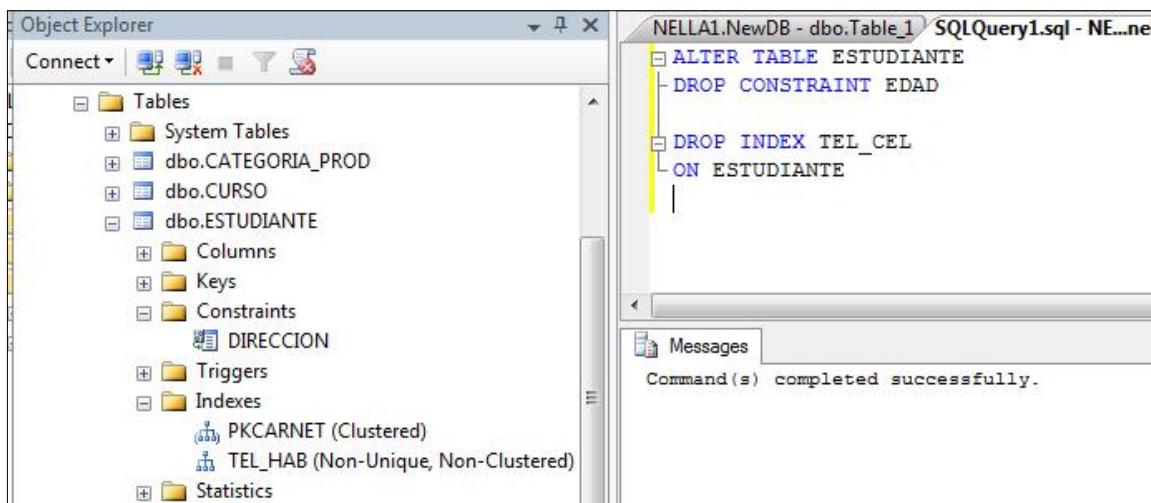
Ejemplo:

Borrar cada uno de los índices y constraints creados en el punto anterior



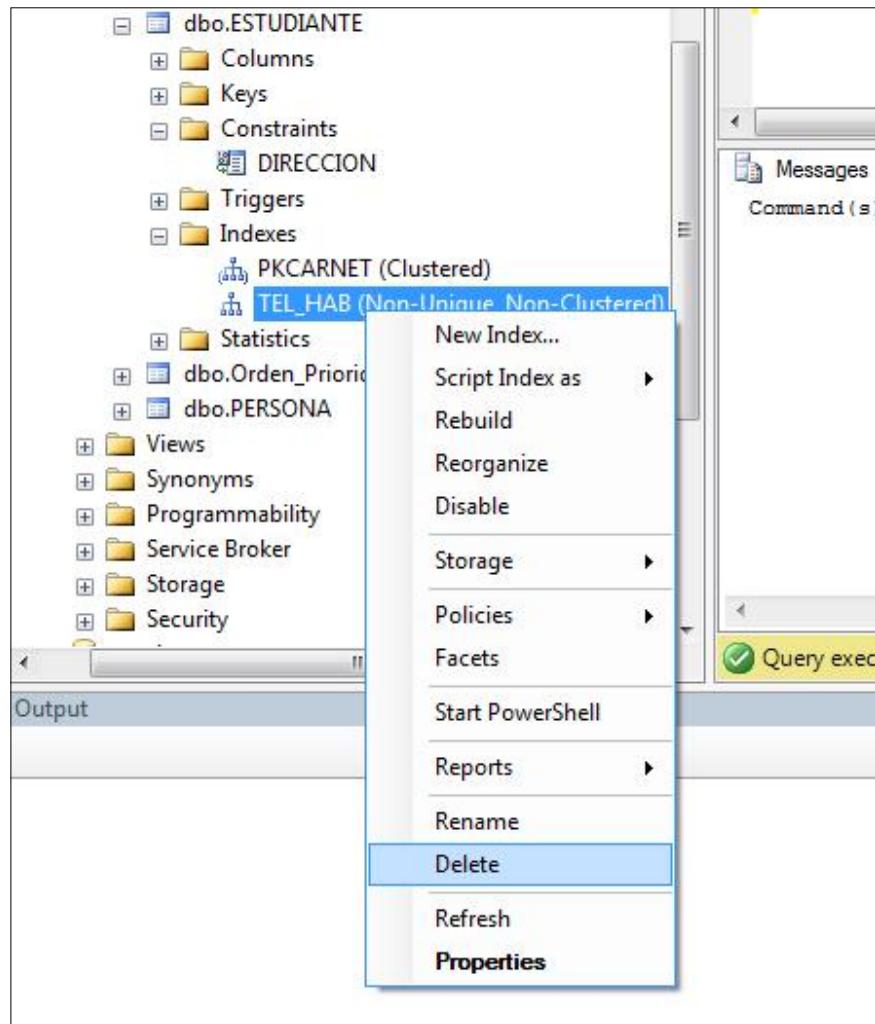
```
ALTER TABLE ESTUDIANTE  
DROP CONSTRAINT EDAD
```

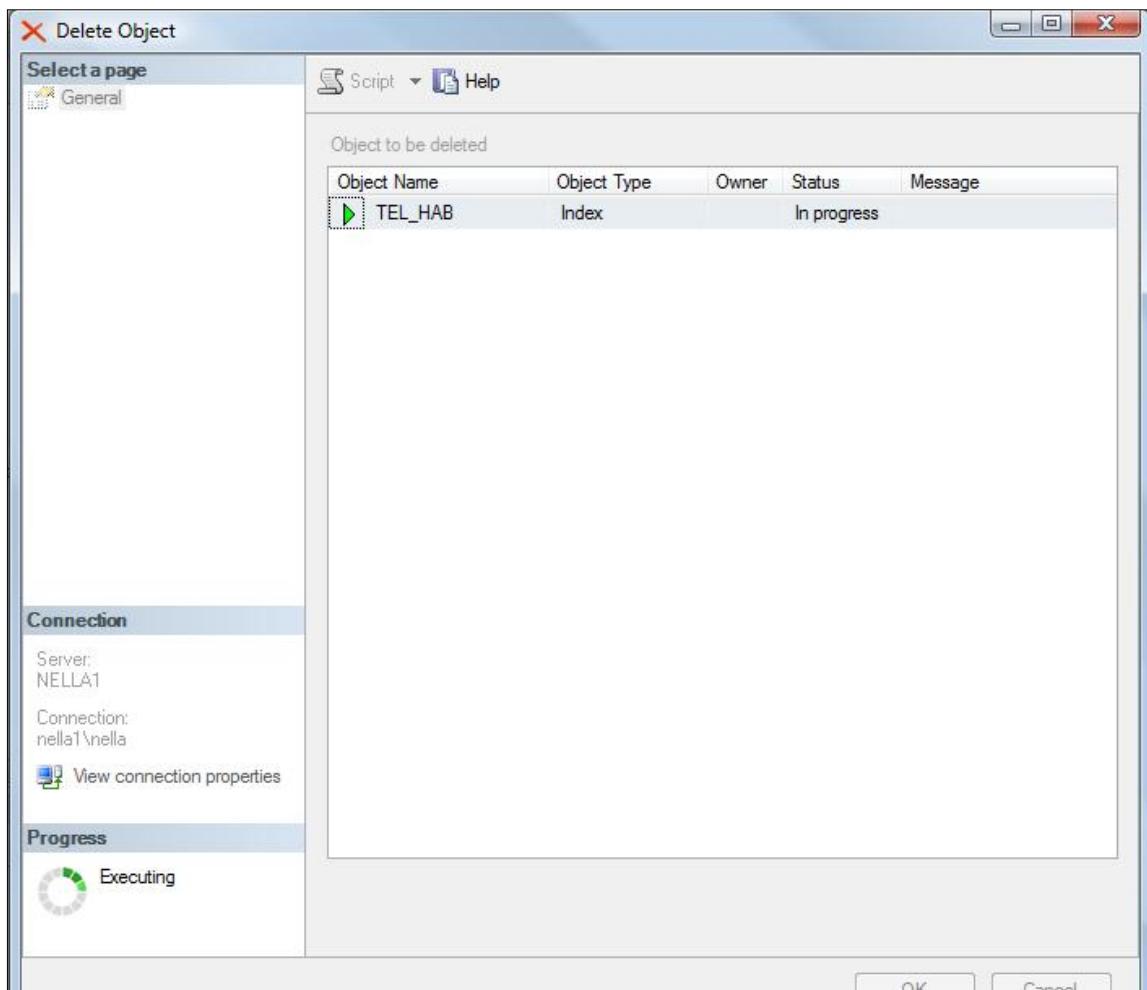
```
DROP INDEX TEL_CEL  
ON ESTUDIANTE
```



Como se puede ver el constraint EDAD y el índice TEL_CEL se eliminaron de la tabla ESTUDIANTE.

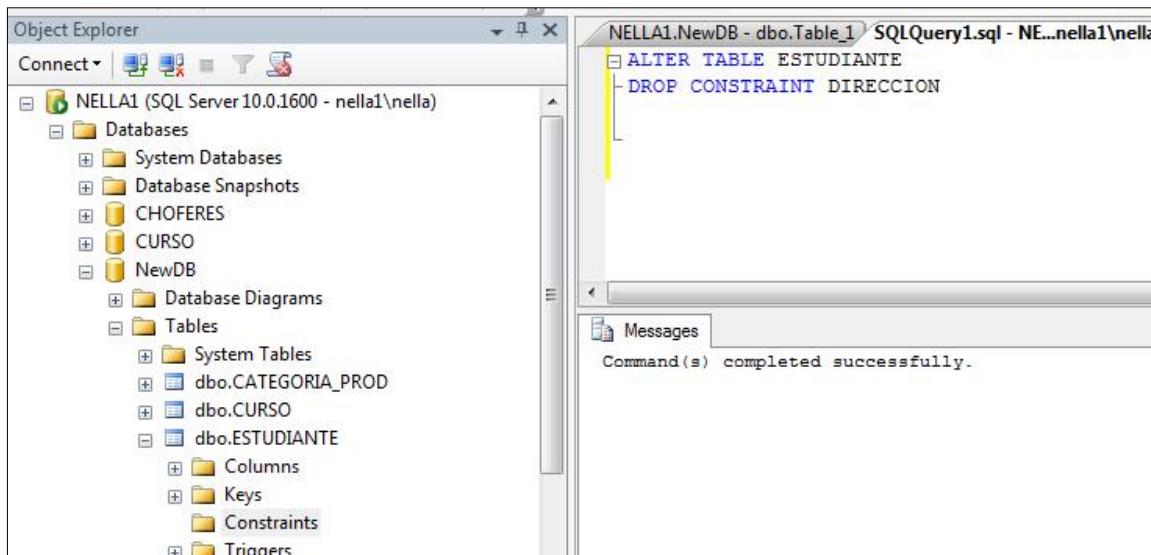
Tanto el índice como el constraint se pueden eliminar dando clic derecho sobre el nombre de cada uno de ellos y seleccionando la opción Delete





Se da clic sobre OK y el objeto es eliminado

```
ALTER TABLE ESTUDIANTE  
DROP CONSTRAINT DIRECCION
```



Borrado de la columna de una tabla por medio del ALTER

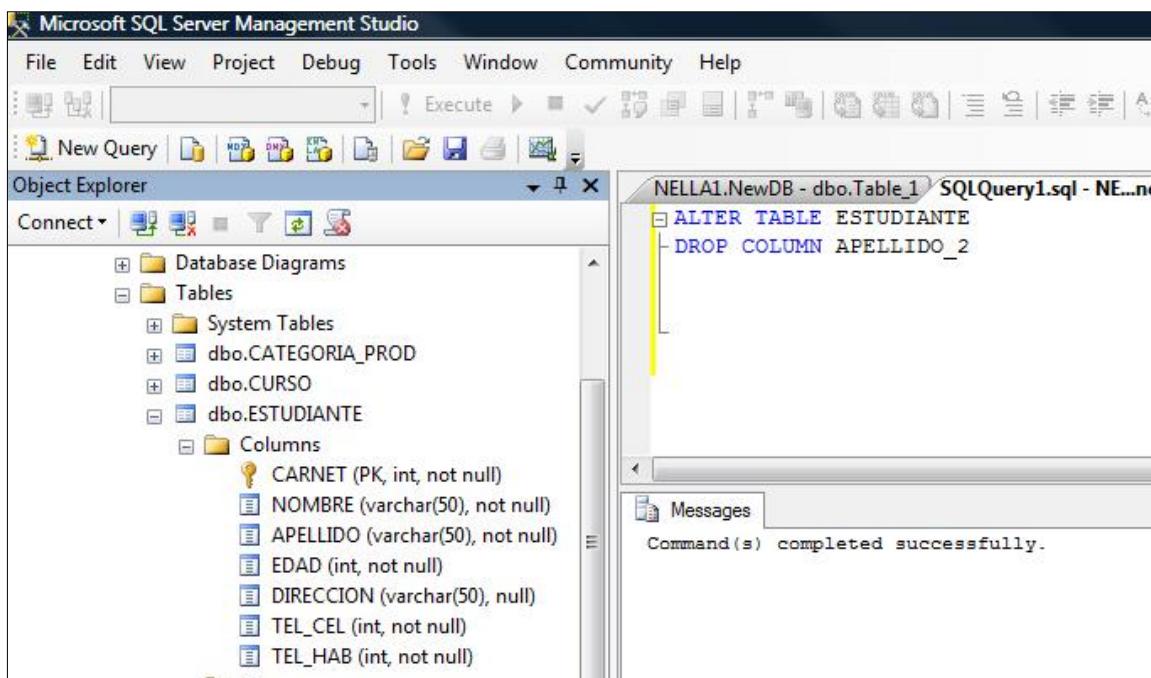
Cláusula

```
ALTER TABLE nombre de tabla
DROP COLUMN nombre de columna;
```

Ejemplo:

Borrar de la tabla ESTUDIANTE, la columna APELLIDO_2

```
ALTER TABLE ESTUDIANTE
DROP COLUMN APELLIDO_2
```



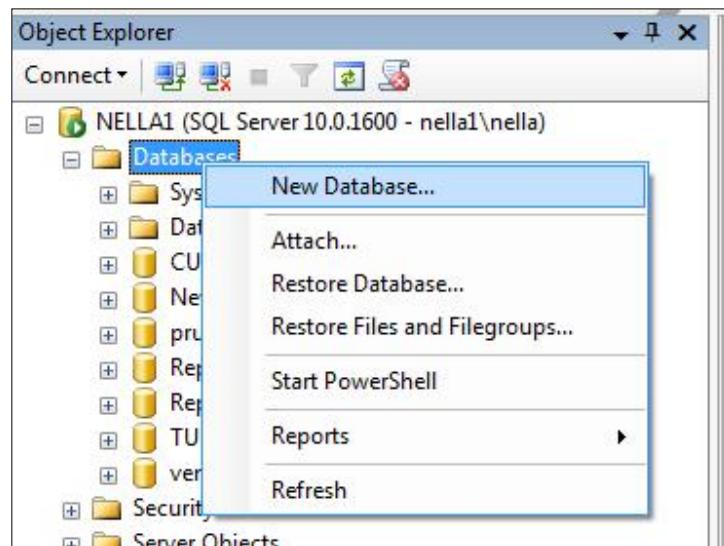
Creación de una base de datos

Anteriormente se explicó que por medio de código como el siguiente se puede crear una base de datos

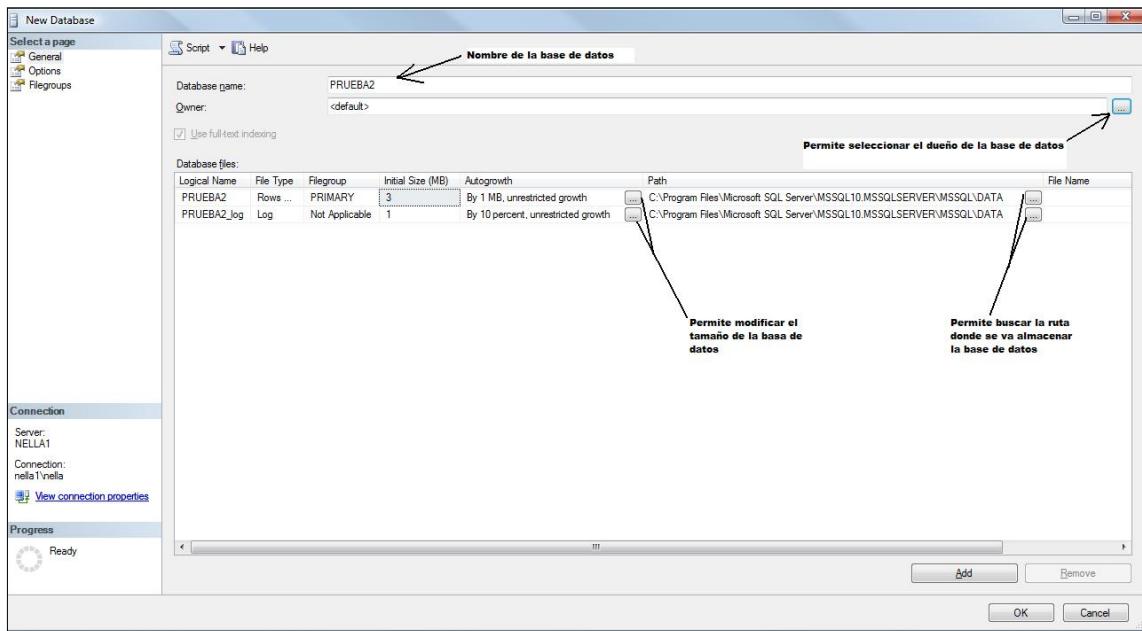
```
USE master
GO
CREATE DATABASE CURSO
ON
(NAME = 'c:\cursosql1\CURSO_dat',
FILENAME = 'c:\cursosql1\CURSO.mdf',
SIZE = 4MB,
MAXSIZE = 10MB,
FILEGROWTH = 1)
GO
```

Sin embargo una de las facilidades que ofrece Microsoft SQL Server, es que al igual que las tablas se pueden crear bases de datos de forma gráfica o manual.

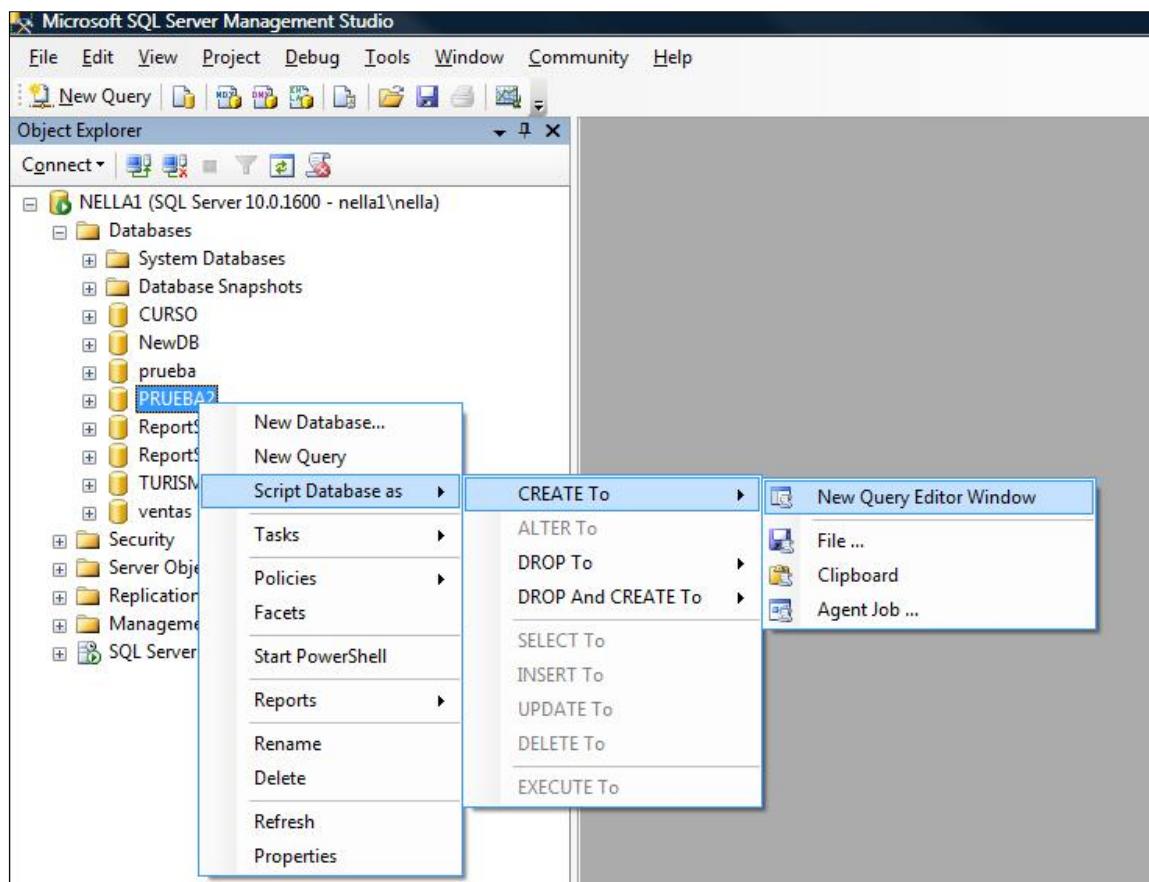
En el Explorador de Objetos (Object Explorer), se da clic derecho sobre la **carpeta Base de datos(Data Base)** y se selecciona la opción **Nueva Base de Datos(New Database)**



Se muestra la siguiente pantalla



Una vez creada la base de datos si se desea ver el query que la creo, se da clic derecho sobre el nombre de la base de datos y se selecciona la opción **Script de la Base de Datos (Script Database as), Crear a (CREATE TO), Editor de Query (New Query Editor Windows)**



```
SQLQuery3.sql - NE...nella1\ella (54))  
GO  
  
ALTER DATABASE [PRUEBA2] SET ALLOW_SNAPSHOT_ISOLATION OFF  
GO  
  
ALTER DATABASE [PRUEBA2] SET PARAMETERIZATION SIMPLE  
GO  
  
ALTER DATABASE [PRUEBA2] SET READ_COMMITTED_SNAPSHOT OFF  
GO  
  
ALTER DATABASE [PRUEBA2] SET HONOR_BROKER_PRIORITY OFF  
GO  
  
ALTER DATABASE [PRUEBA2] SET READ_WRITE  
GO  
  
ALTER DATABASE [PRUEBA2] SET RECOVERY FULL  
GO  
  
ALTER DATABASE [PRUEBA2] SET MULTI_USER  
GO  
  
ALTER DATABASE [PRUEBA2] SET PAGE_VERIFY CHECKSUM  
GO  
  
ALTER DATABASE [PRUEBA2] SET DB_CHAINING OFF  
GO
```

Sub Consultas

Las sub consultas son consultas de select anidadas o select dentro de otro select, esta otra consulta select puede aparecer en la lista de selección o en las cláusulas WHERE o HAVING.

En este tipo de consultas se utilizan las siguientes condiciones

EXISTS: Ciento si una sub consulta devuelve como mínimo un registro

IN: Pertenencia a un conjunto de valores o ser miembro de una sub consulta

Any: Compara el valor con cada valor devuelto por una subconsulta retornando cierto si uno o cualquiera de ellos cumple la condición, debe ir precedido de =, <,>,>= o <=

ALL: Compara el valor con cada valor devuelto por una subconsulta retornando cierto si todos ellos cumplen con la condición debe ir precedido de =, <,>,>= o <=

Ejemplo:

Se necesita saber cuál de las personas actualmente no están inscritos en algún curso.

Operador IN

```
SELECT NOMBRE , APELLIDO FROM PERSONA
WHERE CEDULA NOT IN (SELECT CEDULA FROM HISTORIAL)
```

	NOMBRE	APELLIDO
1	JULIO	MONGE
2	MARIO	MORA
3	JULIA	DIAZ

Muestre los estudiantes que tienen notas entre 90 y 100

```
SELECT NOMBRE , APELLIDO FROM PERSONA
WHERE CEDULA IN (SELECT CEDULA FROM HISTORIAL
WHERE NOTA BETWEEN 90 AND 100)
```

```
SELECT NOMBRE , APELLIDO FROM PERSONA  
WHERE CEDULA IN (SELECT CEDULA FROM HISTORIAL  
WHERE NOTA BETWEEN 90 AND 100)
```

	NOMBRE	APELLIDO
1	JUANA	PAZ
2	JUAN	PEREZ
3	ANA	ARIAS
4	TERESA	PAZ
5	JOSE	ARGUEDAZ

Operadores EXISTS y NOT EXISTS

Mostrar el nombre y apellido de todas aquellas personas que tengan el curso 1 matriculado y la nota sea mayor a 80.

```
SELECT NOMBRE,APELLIDO FROM PERSONA A  
WHERE EXISTS(SELECT * FROM HISTORIAL B  
WHERE A.CEDULA = B.CEDULA  
AND B.CODIGO = 1  
AND B.NOTA > 80)
```

```

SELECT NOMBRE, APELLIDO FROM PERSONA A
WHERE EXISTS(SELECT * FROM HISTORIAL B
WHERE A.CEDULA = B.CEDULA
AND B.CODIGO = 1
AND B.NOTA > 80)

```

Results

	NOMBRE	APELLIDO
1	JOSE	ARGUEDAZ

Mostrar el nombre, apellido y cedula de aquellas personas que no tienen cursos matriculados

```

SELECT NOMBRE, APELLIDO, CEDULA FROM PERSONA A
WHERE NOT EXISTS(SELECT * FROM HISTORIAL B
WHERE A.CEDULA = B.CEDULA)

```

```

SELECT NOMBRE, APELLIDO, CEDULA FROM PERSONA A
WHERE NOT EXISTS(SELECT * FROM HISTORIAL B
WHERE A.CEDULA = B.CEDULA)

```

Results

	NOMBRE	APELLIDO	CEDULA
1	JULIO	MONGE	209440300
2	MARIO	MORA	409440300
3	JULIA	DIAZ	519440300

Consultas de referencias cruzadas

Las consultas de referencias cruzadas muestran todas las combinaciones posibles de todas las filas o registros de las tablas combinadas.

En este tipo de combinación no se requiere tener una columna en común.

Cuando se utilizan combinaciones cruzadas, SQL Server genera un producto cartesiano en el que el número de filas del conjunto de resultados es igual al número de filas de la primera tabla multiplicado por el número de filas de la segunda tabla.

Para las consultas de referencia cruzada se utiliza el **CROSS JOIN**

Ejemplo:

```
SELECT CODIGO,DESCRICION
FROM CURSO
CROSS JOIN HISTORIAL
```

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

```
SELECT CODIGO,DESCRICION
FROM CURSO
CROSS JOIN HISTORIAL
```

Below the query window is a 'Messages' window displaying the following error message:

Msg 209, Level 16, State 1, Line 1
Ambiguous column name 'CODIGO'.

En este caso el error se da porque tanto en la tabla CURSO como en la tabla HISTORIAL, existe el campo CODIGO, en este caso hacemos uso de los alias.

Ejemplo:

```
SELECT A.CODIGO,DESCRICION
FROM CURSO A
CROSS JOIN HISTORIAL
```

```

SELECT A.CODIGO, DESCRIPCION
FROM CURSO A
CROSS JOIN HISTORIAL

```

	CODIGO	DESCRIPCION
1	1	SQL SERVER 2008
2	1	SQL SERVER 2008
3	1	SQL SERVER 2008
4	1	SQL SERVER 2008
5	1	SQL SERVER 2008
6	1	SQL SERVER 2008
7	1	SQL SERVER 2008
8	1	SQL SERVER 2008
9	1	SQL SERVER 2008
10	1	SQL SERVER 2008
11	1	SQL SERVER 2008
12	1	SQL SERVER 2008
13	1	SQL SERVER 2008
14	1	SQL SERVER 2008
15	1	SQL SERVER 2008
16	1	SQL SERVER 2008
17	1	SQL SERVER 2008
18	2	INTERNET
19	2	INTERNET
20	2	INTERNET
21	2	INTERNET
22	2	INTERNET
23	2	INTERNET

Consultas de Unión Internas

La UNION, es un tipo de consulta interna que se puede considerar como una suma, se sumaría en una tabla de resultado el conjunto de columnas de dos tablas según el criterio aplicado
T1 {1,2,3,4,5,6} y T2 {1.3.5.7}

T1 UNION T2 = {1,2,3,4,5,6,7}

Las dos tablas a unir deben tener el mismo número de columnas y el mismo tipo de dato por columna.

```
SELECT *FROM HISTORIAL
UNION
SELECT * FROM HISTORIAL_RESPALDO
```

Ejemplo:

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

```
SELECT *FROM HISTORIAL
UNION
SELECT * FROM HISTORIAL_RESPALDO
```

The results grid displays 17 rows of data with the following columns:

	CEDULA	CODIGO	NOTA	SEDE
1	109440300	3	90	HEREDIA
2	109440300	4	80	SAN JOSE
3	109440300	5	90	ALAJUELA
4	309440300	4	75	SAN JOSE
5	509440300	1	80	HEREDIA
6	509440300	2	85	HEREDIA
7	609440300	1	75	SAN JOSE
8	609440300	3	90	SAN JOSE
9	609440300	5	75	HEREDIA
10	709440300	3	90	SAN JOSE
11	779440300	2	0	HEREDIA
12	809440300	1	80	SAN JOSE
13	809440300	2	95	SAN JOSE
14	809440300	3	95	SAN JOSE
15	909440300	1	90	HEREDIA
16	909440300	2	95	SAN JOSE
17	909440300	5	90	ALAJUELA

Las consultas internas combinan tablas mediante la comparación de los valores de las columnas que son comunes a ambas tablas.

Se devuelven las filas o registros que cumplen la condición.

Para este tipo de consultas se utiliza JOIN o INNER JOIN

Cláusula

```
SELECT *
FROM Table1
[INNER] JOIN Table2
ON Table1.column = Table2.column;
```

Ejemplo:

Mostrar el nombre, apellido y cédula de todas las personas que están actualmente inscritos en algún curso

```
SELECT NOMBRE, APELLIDO, PERSONA.CEDULA
FROM PERSONA
JOIN HISTORIAL
ON PERSONA.CEDULA = HISTORIAL.CEDULA
```

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

```
SELECT NOMBRE, APELLIDO, PERSONA.CEDULA
FROM PERSONA
JOIN HISTORIAL
ON PERSONA.CEDULA = HISTORIAL.CEDULA
```

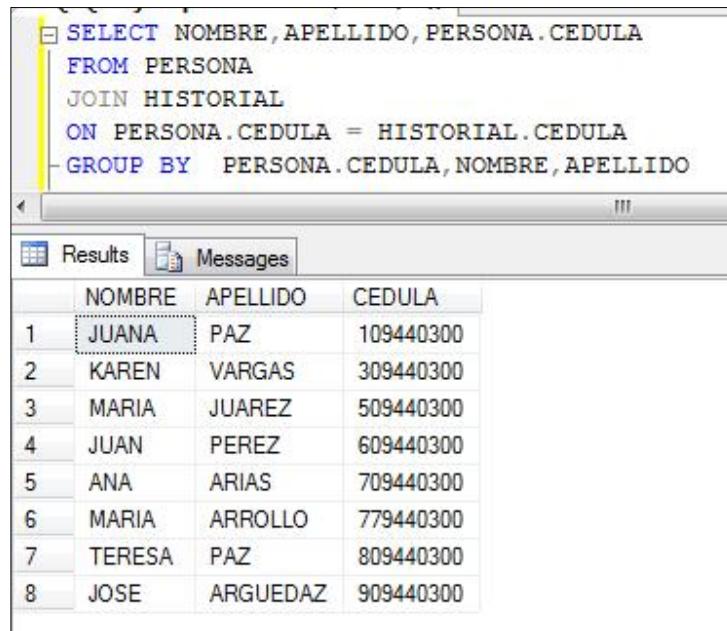
Below the query window is a results grid. The grid has three columns: NOMBRE, APELLIDO, and CEDULA. The data is as follows:

	NOMBRE	APELLIDO	CEDULA
1	JUANA	PAZ	109440300
2	JUANA	PAZ	109440300
3	JUANA	PAZ	109440300
4	KAREN	VARGAS	309440300
5	MARIA	JUAREZ	509440300
6	MARIA	JUAREZ	509440300
7	JUAN	PEREZ	609440300
8	JUAN	PEREZ	609440300
9	JUAN	PEREZ	609440300
10	ANA	ARIAS	709440300
11	MARIA	ARROLLO	779440300
12	TERESA	PAZ	809440300
13	TERESA	PAZ	809440300
14	TERESA	PAZ	809440300
15	JOSE	ARGUE...	909440300
16	JOSE	ARGUE...	909440300
17	JOSE	ARGUE...	909440300

Como se puede ver muestra duplicados en la consulta
En este caso podemos utilizar un GROUP BY

```
SELECT NOMBRE, APELLIDO, PERSONA.CEDULA
FROM PERSONA
JOIN HISTORIAL
ON PERSONA.CEDULA = HISTORIAL.CEDULA
```

GROUP BY PERSONA.CEDULA,NOMBRE,APELLIDO



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

```

SELECT NOMBRE, APELLIDO, PERSONA.CEDULA
FROM PERSONA
JOIN HISTORIAL
ON PERSONA.CEDULA = HISTORIAL.CEDULA
GROUP BY PERSONA.CEDULA, NOMBRE, APELLIDO
    
```

Below the code, there are two tabs: "Results" and "Messages". The "Results" tab displays the following data:

	NOMBRE	APELLIDO	CEDULA
1	JUANA	PAZ	109440300
2	KAREN	VARGAS	309440300
3	MARIA	JUAREZ	509440300
4	JUAN	PEREZ	609440300
5	ANA	ARIAS	709440300
6	MARIA	ARROLLO	779440300
7	TERESA	PAZ	809440300
8	JOSE	ARGUEDAZ	909440300

El INNER JOIN funciona de la misma manera que el JOIN así que si lo utilizamos el resultado debería ser el mismo.

Ejemplo:

```

SELECT NOMBRE, APELLIDO, A.CEDULA
FROM PERSONA A
INNER JOIN HISTORIAL B
ON A.CEDULA = B.CEDULA
GROUP BY A.CEDULA, NOMBRE, APELLIDO
    
```

Nótese que en este caso se están utilizando alias.

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

```
SELECT NOMBRE, APELLIDO, A.CEDULA
FROM PERSONA A
INNER JOIN HISTORIAL B
ON A.CEDULA = B.CEDULA
GROUP BY A.CEDULA, NOMBRE, APELLIDO
```

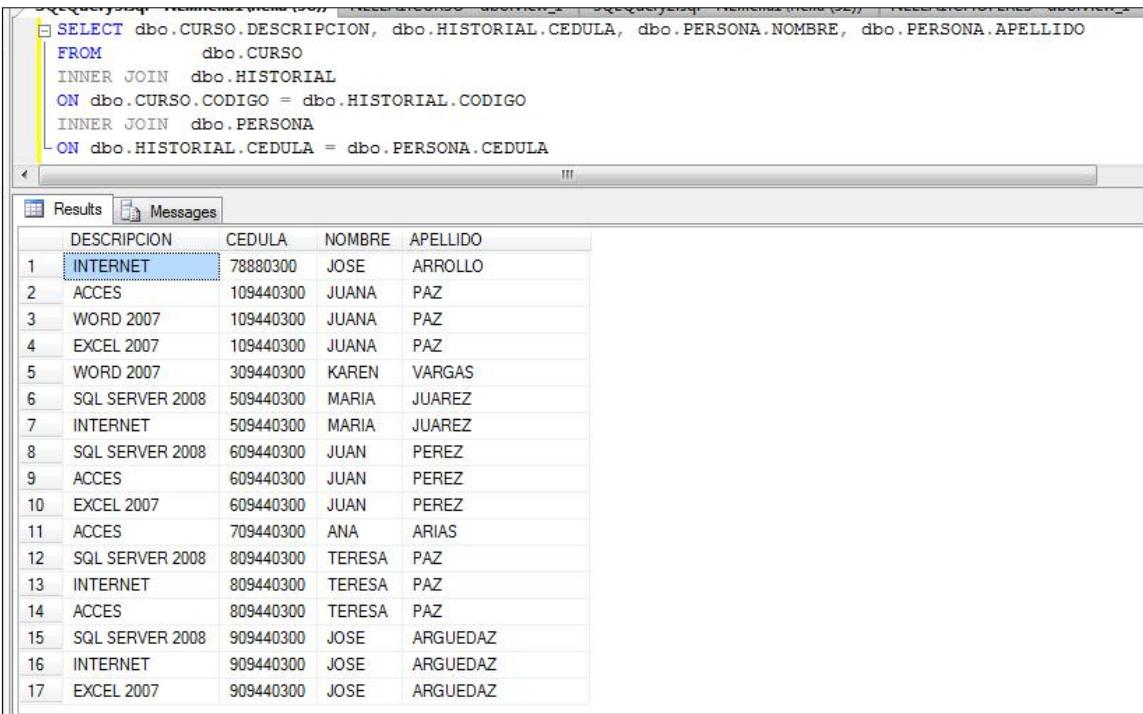
Below the query window is a results grid with three columns: NOMBRE, APELLIDO, and CEDULA. The data is as follows:

	NOMBRE	APELLIDO	CEDULA
1	JUANA	PAZ	109440300
2	KAREN	VARGAS	309440300
3	MARIA	JUAREZ	509440300
4	JUAN	PEREZ	609440300
5	ANA	ARIAS	709440300
6	MARIA	ARROLLO	779440300
7	TERESA	PAZ	809440300
8	JOSE	ARGUEDAZ	909440300

Como se puede ver el resultado es el mismo que al utilizar el JOIN.

Si se quisiera obtener la información de las personas que están matriculadas por curso sería de la siguiente manera sin los alias

```
SELECT      dbo.CURSO.DESCRIPCION,      dbo.HISTORIAL.CEDULA,      dbo.PERSONA.NOMBRE,
dbo.PERSONA.APELLIDO
FROM        dbo.CURSO
INNER JOIN  dbo.HISTORIAL
ON         dbo.CURSO.CODIGO = dbo.HISTORIAL.CODIGO
INNER JOIN  dbo.PERSONA
ON         dbo.HISTORIAL.CEDULA = dbo.PERSONA.CEDULA
```



The screenshot shows a SQL Server Management Studio window. At the top, there is a query editor pane containing the following T-SQL code:

```

SELECT dbo.CURSO.DESCRIPCION, dbo.HISTORIAL.CEDULA, dbo.PERSONA.NOMBRE, dbo.PERSONA.APELLIDO
FROM      dbo.CURSO
INNER JOIN dbo.HISTORIAL
ON dbo.CURSO.CODIGO = dbo.HISTORIAL.CODIGO
INNER JOIN dbo.PERSONA
ON dbo.HISTORIAL.CEDULA = dbo.PERSONA.CEDULA
  
```

Below the query editor is a results grid titled "Results". The grid displays 17 rows of data with four columns: DESCRIPCION, CEDULA, NOMBRE, and APELLIDO. The data is as follows:

	DESCRIPCION	CEDULA	NOMBRE	APELLIDO
1	INTERNET	78880300	JOSE	ARROLLO
2	ACCES	109440300	JUANA	PAZ
3	WORD 2007	109440300	JUANA	PAZ
4	EXCEL 2007	109440300	JUANA	PAZ
5	WORD 2007	309440300	KAREN	VARGAS
6	SQL SERVER 2008	509440300	MARIA	JUAREZ
7	INTERNET	509440300	MARIA	JUAREZ
8	SQL SERVER 2008	609440300	JUAN	PEREZ
9	ACCES	609440300	JUAN	PEREZ
10	EXCEL 2007	609440300	JUAN	PEREZ
11	ACCES	709440300	ANA	ARIAS
12	SQL SERVER 2008	809440300	TERESA	PAZ
13	INTERNET	809440300	TERESA	PAZ
14	ACCES	809440300	TERESA	PAZ
15	SQL SERVER 2008	909440300	JOSE	ARGUEDAZ
16	INTERNET	909440300	JOSE	ARGUEDAZ
17	EXCEL 2007	909440300	JOSE	ARGUEDAZ

Con alias

```

SELECT A.DESCRIPCION, B.CEDULA, C.NOMBRE, C.APELLIDO
FROM      dbo.CURSO A
INNER JOIN dbo.HISTORIAL B
ON A.CODIGO = B.CODIGO
INNER JOIN dbo.PERSONA C
ON B.CEDULA = C.CEDULA
  
```

```
SELECT A.DESCRIPCION, B.CEDULA, C.NOMBRE, C.APELLIDO
FROM      dbo.CURSO A
INNER JOIN  dbo.HISTORIAL B
ON A.CODIGO = B.CODIGO
INNER JOIN  dbo.PERSONA C
ON B.CEDULA = C.CEDULA
```

	DESCRIPCION	CEDULA	NOMBRE	APELLIDO
1	INTERNET	78880300	JOSE	ARROLLO
2	ACCES	109440300	JUANA	PAZ
3	WORD 2007	109440300	JUANA	PAZ
4	EXCEL 2007	109440300	JUANA	PAZ
5	WORD 2007	309440300	KAREN	VARGAS
6	SQL SERVER 2008	509440300	MARIA	JUAREZ
7	INTERNET	509440300	MARIA	JUAREZ
8	SQL SERVER 2008	609440300	JUAN	PEREZ
9	ACCES	609440300	JUAN	PEREZ
10	EXCEL 2007	609440300	JUAN	PEREZ
11	ACCES	709440300	ANA	ARIAS
12	SQL SERVER 2008	809440300	TERESA	PAZ
13	INTERNET	809440300	TERESA	PAZ
14	ACCES	809440300	TERESA	PAZ
15	SQL SERVER 2008	909440300	JOSE	ARGUEDAZ
16	INTERNET	909440300	JOSE	ARGUEDAZ
17	EXCEL 2007	909440300	JOSE	ARGUEDAZ

El resultado es el mismo

Para poder realizar una consulta en forma gráfica se da clic derecho sobre la carpeta Vistas (Views), Nueva vista(New View) y se agregan las tablas y los campos

The screenshot shows the SSMS interface. At the top, three tables are listed in the Object Explorer: CURSO, HISTORIAL, and PERSONA. The CURSO table has columns * (All Columns), CODIGO, DESCRIPCION, FECHA_INICIO, and FECHA_FIN. The HISTORIAL table has columns * (All Columns), CEDULA, CODIGO, NOTA, and SEDE. The PERSONA table has columns * (All Columns), CEDULA, NOMBRE, APELLIDO, and EDAD. Below the Object Explorer, a query grid displays the selected columns: DESCRIPCION, CEDULA, NOMBRE, and APELLIDO. The query itself is:

```

SELECT
  dbo.CURSO.DESCRIPCION,
  dbo.HISTORIAL.CEDULA,
  dbo.PERSONA.NOMBRE,
  dbo.PERSONA.APELLIDO
FROM
  dbo.CURSO INNER JOIN
    dbo.HISTORIAL ON dbo.CURSO.CODIGO = dbo.HISTORIAL.CODIGO INNER JOIN
  dbo.PERSONA ON dbo.HISTORIAL.CEDULA = dbo.PERSONA.CEDULA
  
```

At the bottom of the query grid, there is a preview of the results:

	DESCRIPCION	CEDULA	NOMBRE	APELLIDO
▶	INTERNET	78880300	JOSE	ARROLLO
	ACCES	109440300	JUANA	PAZ
	WORD 2007	109440300	JUANA	PAZ
	EXCEL 2007	109440300	JUANA	PAZ

Below the preview, it says "1 of 17" and "Cell is Read Only."

Consultas de Unión Externa

Las uniones externas izquierda y derecha combinan filas de dos tablas que cumplen una condición de combinación, más las filas de la tabla izquierda o derecha que no la cumplen.

Las uniones externas izquierda y derecha se utilizan cuando se necesita una lista completa de datos de una de las tablas más la información que cumpla la condición.

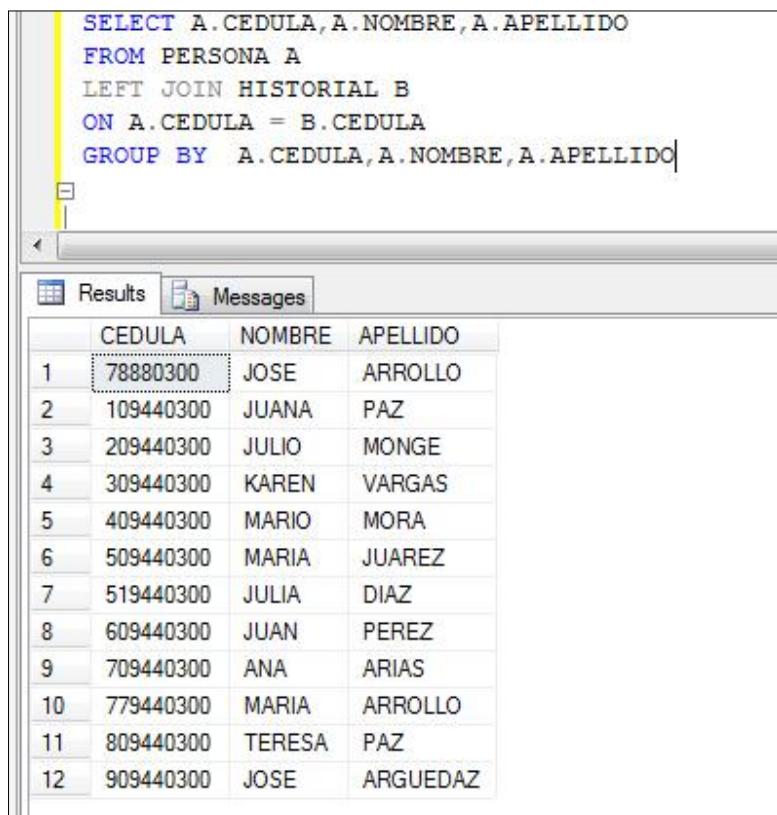
Este tipo de uniones no se utilizan a menudo, porque en algunos casos los resultados dados no son los esperados en estos casos se deben realizar varias pruebas o cambiar el orden de las tablas, en ocasiones utilizar otro tipo de consulta.

LEFT JOIN: Unión a la izquierda, toma los datos de la primera tabla especificada.

```

SELECT A.CEDULA,A.NOMBRE,A.APELLIDO
FROM PERSONA A
  
```

```
LEFT JOIN HISTORIAL B
ON A.CEDULA = B.CEDULA
GROUP BY A.CEDULA,A.NOMBRE,A.APELLIDO
```



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

```
SELECT A.CEDULA, A.NOMBRE, A.APELLIDO
FROM PERSONA A
LEFT JOIN HISTORIAL B
ON A.CEDULA = B.CEDULA
GROUP BY A.CEDULA, A.NOMBRE, A.APELLIDO
```

The results grid displays 12 rows of data:

	CEDULA	NOMBRE	APELLIDO
1	78880300	JOSE	ARROLLO
2	109440300	JUANA	PAZ
3	209440300	JULIO	MONGE
4	309440300	KAREN	VARGAS
5	409440300	MARIO	MORA
6	509440300	MARIA	JUAREZ
7	519440300	JULIA	DIAZ
8	609440300	JUAN	PEREZ
9	709440300	ANA	ARIAS
10	779440300	MARIA	ARROLLO
11	809440300	TERESA	PAZ
12	909440300	JOSE	ARGUEDAZ

Muestra los datos de la tabla Persona, coincidan o no con la tabla Historial.

Se puede utilizar el OUTER, da el mismo resultado

```
SELECT A.CEDULA,A.NOMBRE,A.APELLIDO
FROM PERSONA A
LEFT OUTER JOIN HISTORIAL B
ON A.CEDULA = B.CEDULA
GROUP BY A.CEDULA,A.NOMBRE,A.APELLIDO
```

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

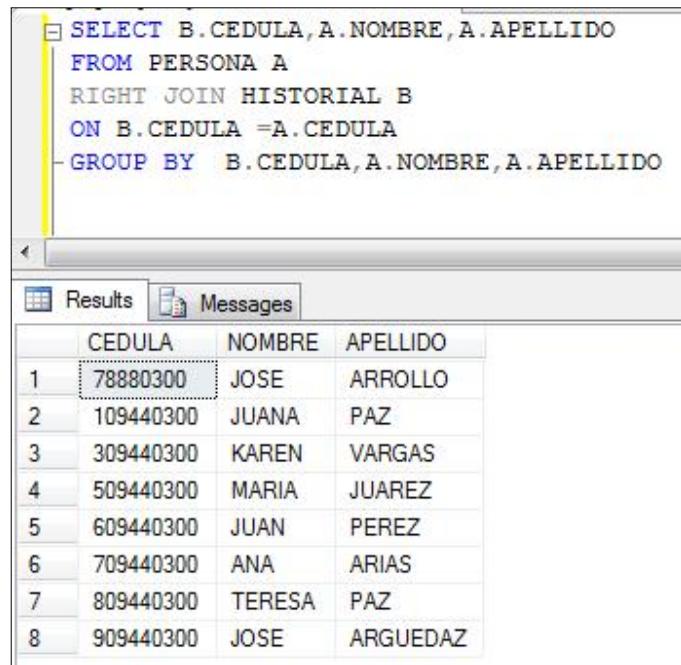
```
SELECT A.CEDULA, A.NOMBRE, A.APELLIDO  
FROM PERSONA A  
LEFT OUTER JOIN HISTORIAL B  
ON A.CEDULA = B.CEDULA  
GROUP BY A.CEDULA, A.NOMBRE, A.APELLIDO
```

The results grid displays 12 rows of data:

	CEDULA	NOMBRE	APELLIDO
1	78880300	JOSE	ARROLLO
2	109440300	JUANA	PAZ
3	209440300	JULIO	MONGE
4	309440300	KAREN	VARGAS
5	409440300	MARIO	MORA
6	509440300	MARIA	JUAREZ
7	519440300	JULIA	DIAZ
8	609440300	JUAN	PEREZ
9	709440300	ANA	ARIAS
10	779440300	MARIA	ARROLLO
11	809440300	TERESA	PAZ
12	909440300	JOSE	ARGUEDAZ

RIGHT JOIN: Unión a la derecha, toma los datos de la segunda tabla especificada.

```
SELECT B.CEDULA,A.NOMBRE,A.APELLIDO  
FROM PERSONA A  
RIGHT JOIN HISTORIAL B  
ON B.CEDULA =A.CEDULA  
GROUP BY B.CEDULA,A.NOMBRE,A.APELLIDO
```



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

```

SELECT B.CEDULA, A.NOMBRE, A.APELLIDO
FROM PERSONA A
RIGHT JOIN HISTORIAL B
ON B.CEDULA = A.CEDULA
GROUP BY B.CEDULA, A.NOMBRE, A.APELLIDO

```

Below the code, there are two tabs: "Results" and "Messages". The "Results" tab displays the following data:

	CEDULA	NOMBRE	APELLIDO
1	78880300	JOSE	ARROLLO
2	109440300	JUANA	PAZ
3	309440300	KAREN	VARGAS
4	509440300	MARIA	JUAREZ
5	609440300	JUAN	PEREZ
6	709440300	ANA	ARIAS
7	809440300	TERESA	PAZ
8	909440300	JOSE	ARGUEDAZ

Muestra los datos de la tabla Historial, coincidan o no con la tabla Persona.

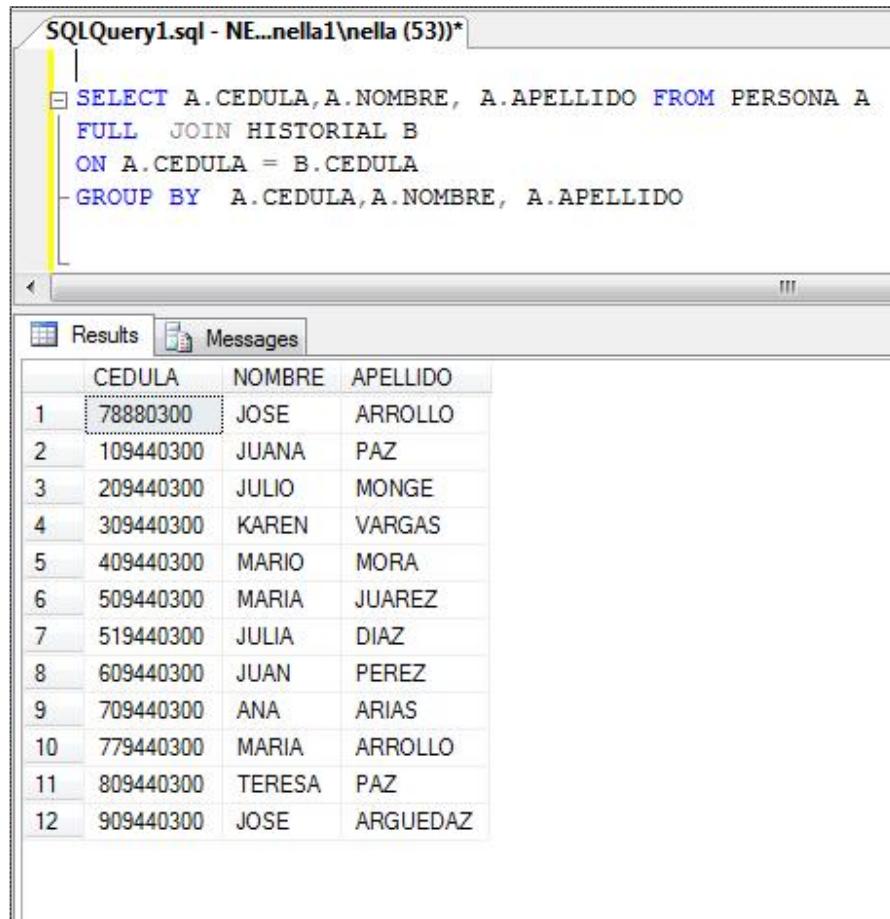
FULL OUTER JOIN O FULL JOIN: Combinación externa completa, incluye todas las filas de ambas tablas, con independencia de que la otra tabla tenga o no un valor coincidente.

Ejemplo 1:

```

SELECT A.CEDULA,A.NOMBRE, A.APELLIDO FROM PERSONA A
FULL JOIN HISTORIAL B
ON A.CEDULA = B.CEDULA
GROUP BY A.CEDULA,A.NOMBRE, A.APELLIDO

```



The screenshot shows a SQL Server Management Studio window. The top pane displays a query named 'SQLQuery1.sql' with the following T-SQL code:

```
SELECT A.CEDULA, A.NOMBRE, A.APELLIDO FROM PERSONA A
FULL JOIN HISTORIAL B
ON A.CEDULA = B.CEDULA
GROUP BY A.CEDULA, A.NOMBRE, A.APELLIDO
```

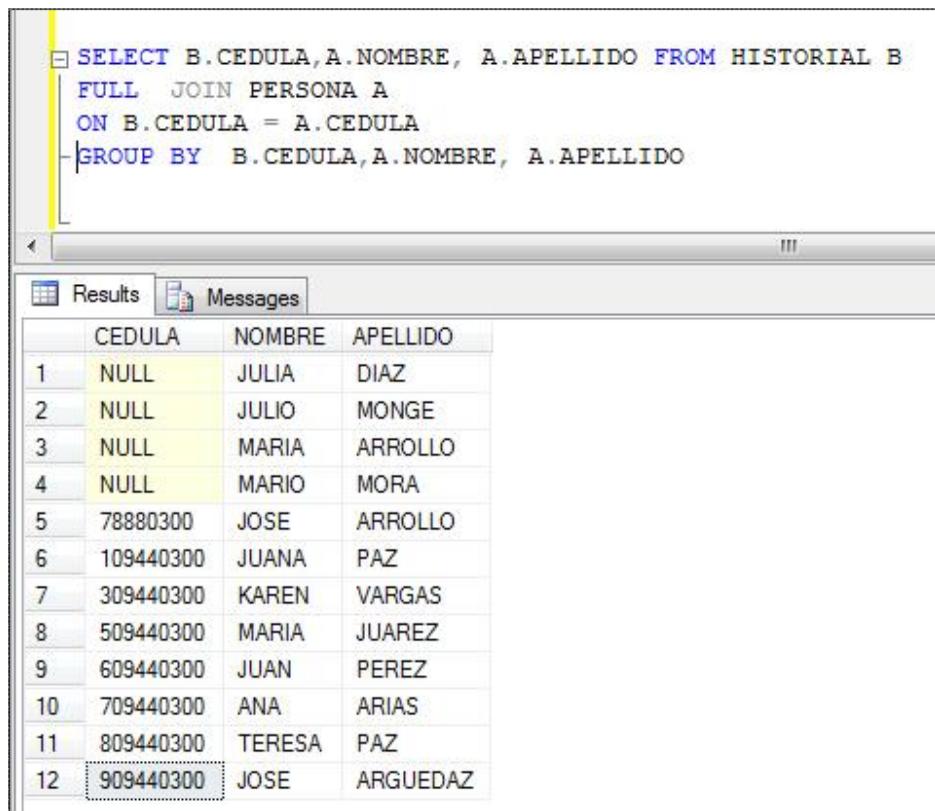
The bottom pane shows the results of the query, which is a grid with three columns: CEDULA, NOMBRE, and APELLIDO. The data consists of 12 rows, each numbered from 1 to 12.

	CEDULA	NOMBRE	APELLIDO
1	78880300	JOSE	ARROLLO
2	109440300	JUANA	PAZ
3	209440300	JULIO	MONGE
4	309440300	KAREN	VARGAS
5	409440300	MARIO	MORA
6	509440300	MARIA	JUAREZ
7	519440300	JULIA	DIAZ
8	609440300	JUAN	PEREZ
9	709440300	ANA	ARIAS
10	779440300	MARIA	ARROLLO
11	809440300	TERESA	PAZ
12	909440300	JOSE	ARGUEDAZ

Muestra la información de la tabla Persona, tanto los que coinciden o no con la tabla Historial.

Ejemplo 2

```
SELECT B.CEDULA,A.NOMBRE, A.APELLIDO FROM HISTORIAL B
FULL JOIN PERSONA A
ON B.CEDULA = A.CEDULA
GROUP BY B.CEDULA,A.NOMBRE, A.APELLIDO
```



The screenshot shows a SQL query window with a results grid. The query is:

```

SELECT B.CEDULA, A.NOMBRE, A.APELLIDO FROM HISTORIAL B
FULL JOIN PERSONA A
ON B.CEDULA = A.CEDULA
GROUP BY B.CEDULA, A.NOMBRE, A.APELLIDO

```

The results grid has columns CEDULA, NOMBRE, and APELLIDO. The data is:

	CEDULA	NOMBRE	APELLIDO
1	NULL	JULIA	DIAZ
2	NULL	JULIO	MONGE
3	NULL	MARIA	ARROLLO
4	NULL	MARIO	MORA
5	78880300	JOSE	ARROLLO
6	109440300	JUANA	PAZ
7	309440300	KAREN	VARGAS
8	509440300	MARIA	JUAREZ
9	609440300	JUAN	PEREZ
10	709440300	ANA	ARIAS
11	809440300	TERESA	PAZ
12	909440300	JOSE	ARGUEDAZ

Muestra los datos de la tabla Historial, y la información de la tabla Persona que no coincide con la de Historial la muestra en NULL.

En una consulta externa completa se puede utilizar la cláusula WHERE, para devolver solo los datos coincidentes.

Si tomamos los 2 ejemplos anteriores tenemos lo siguiente

Ejemplo 1:

Devuelve las personas que tienen cursos matriculados

```

SELECT A.CEDULA,A.NOMBRE, A.APELLIDO FROM PERSONA A
FULL JOIN HISTORIAL B
ON A.CEDULA = B.CEDULA
WHERE B.CEDULA IS NOT NULL
GROUP BY A.CEDULA,A.NOMBRE, A.APELLIDO

```

```

SELECT A.CEDULA, A.NOMBRE, A.APELLIDO FROM PERSONA A
FULL JOIN HISTORIAL B
ON A.CEDULA = B.CEDULA
WHERE B.CEDULA IS NOT NULL
GROUP BY A.CEDULA, A.NOMBRE, A.APELLIDO

```

Results

	CEDULA	NOMBRE	APELLIDO
1	78880300	JOSE	ARROLLO
2	109440300	JUANA	PAZ
3	309440300	KAREN	VARGAS
4	509440300	MARIA	JUAREZ
5	609440300	JUAN	PEREZ
6	709440300	ANA	ARIAS
7	809440300	TERESA	PAZ
8	909440300	JOSE	ARGUEDAZ

Devuelve las personas que no tienen cursos matriculados

```

SELECT A.CEDULA, A.NOMBRE, A.APELLIDO FROM PERSONA A
FULL JOIN HISTORIAL B
ON B.CEDULA = A.CEDULA
WHERE B.CEDULA IS NULL
GROUP BY A.CEDULA, A.NOMBRE, A.APELLIDO

```

```

SELECT A.CEDULA, A.NOMBRE, A.APELLIDO FROM PERSONA A
FULL JOIN HISTORIAL B
ON B.CEDULA = A.CEDULA
WHERE B.CEDULA IS NULL
GROUP BY A.CEDULA, A.NOMBRE, A.APELLIDO

```

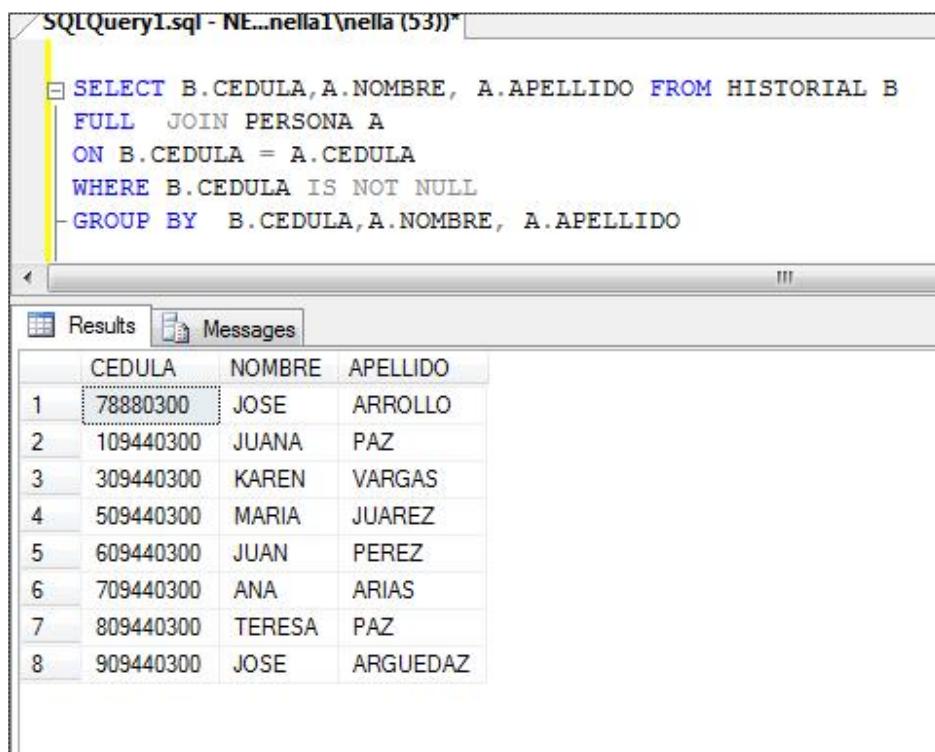
Results

	CEDULA	NOMBRE	APELLIDO
1	209440300	JULIO	MONGE
2	409440300	MARIO	MORA
3	519440300	JULIA	DIAZ
4	779440300	MARIA	ARROLLO

Ejemplo 2:

Devuelve las personas que tienen cursos matriculados

```
SELECT B.CEDULA,A.NOMBRE, A.APELLIDO FROM HISTORIAL B  
FULL JOIN PERSONA A  
ON B.CEDULA = A.CEDULA  
WHERE B.CEDULA IS NOT NULL  
GROUP BY B.CEDULA,A.NOMBRE, A.APELLIDO
```



The screenshot shows a SQL Server Management Studio window titled "SQLQuery1.sql - NE...nella1\abella (53)*". The query pane contains the T-SQL code provided above. The results pane is visible below, showing a grid of data with three columns: CEDULA, NOMBRE, and APELLIDO. The data consists of 8 rows, each representing a person with their ID, name, and last name.

	CEDULA	NOMBRE	APELLIDO
1	78880300	JOSE	ARROLLO
2	109440300	JUANA	PAZ
3	309440300	KAREN	VARGAS
4	509440300	MARIA	JUAREZ
5	609440300	JUAN	PEREZ
6	709440300	ANA	ARIAS
7	809440300	TERESA	PAZ
8	909440300	JOSE	ARGUEDAZ

Devuelve las personas que no tienen cursos matriculados

```
SELECT B.CEDULA,A.NOMBRE, A.APELLIDO FROM HISTORIAL B  
FULL JOIN PERSONA A  
ON B.CEDULA = A.CEDULA  
WHERE B.CEDULA IS NULL  
GROUP BY B.CEDULA,A.NOMBRE, A.APELLIDO
```

```

SQLQuery1.sql - NE...nella1\abella (53)*
SELECT B.CEDULA, A.NOMBRE, A.APELLIDO FROM HISTORIAL B
  FULL JOIN PERSONA A
    ON B.CEDULA = A.CEDULA
   WHERE B.CEDULA IS NULL
  GROUP BY B.CEDULA, A.NOMBRE, A.APELLIDO
  
```

	CEDULA	NOMBRE	APELLIDO
1	NULL	JULIA	DIAZ
2	NULL	JULIO	MONGE
3	NULL	MARIA	ARROLLO
4	NULL	MARIO	MORA

Como se puede ver al tomar el campo Cedula de la tabla Historial, va a mostrar un NULL, porque este dato no existe en la tabla Historial, debe ser tomado de la tabla Persona, tal y como se realizó en el ejemplo 1

De igual forma la cláusula WHERE se puede utilizar en las consultas con LEFT JOIN y RIGHT JOIN

Ejemplo:

Mostrar las personas que tienen cursos matriculados

```

SELECT B.CEDULA,A.NOMBRE, A.APELLIDO
FROM HISTORIAL B
LEFT JOIN PERSONA A
ON B.CEDULA = A.CEDULA
WHERE B.CEDULA IS NOT NULL
GROUP BY B.CEDULA,A.NOMBRE, A.APELLIDO
  
```

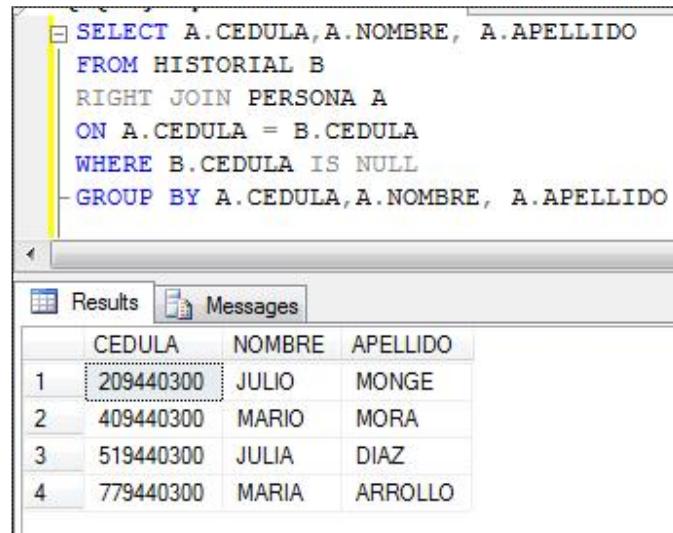
```
SELECT B.CEDULA, A.NOMBRE, A.APELLIDO  
FROM HISTORIAL B  
LEFT JOIN PERSONA A  
ON B.CEDULA = A.CEDULA  
WHERE B.CEDULA IS NOT NULL  
GROUP BY B.CEDULA, A.NOMBRE, A.APELLIDO
```

	CEDULA	NOMBRE	APELLIDO
1	78880300	JOSE	ARROLLO
2	109440300	JUANA	PAZ
3	309440300	KAREN	VARGAS
4	509440300	MARIA	JUAREZ
5	609440300	JUAN	PEREZ
6	709440300	ANA	ARIAS
7	809440300	TERESA	PAZ
8	909440300	JOSE	ARGUEDAZ

En este caso se utiliza el **LEFT JOIN**, porque lo que se quiere es extraer la información de la tabla Historial.

Mostrar las personas que no tienen cursos matriculados

```
SELECT A.CEDULA,A.NOMBRE, A.APELLIDO  
FROM HISTORIAL B  
RIGHT JOIN PERSONA A  
ON A.CEDULA = B.CEDULA  
WHERE B.CEDULA IS NULL  
GROUP BY A.CEDULA,A.NOMBRE, A.APELLIDO
```



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

```

SELECT A.CEDULA, A.NOMBRE, A.APELLIDO
FROM HISTORIAL B
RIGHT JOIN PERSONA A
ON A.CEDULA = B.CEDULA
WHERE B.CEDULA IS NULL
GROUP BY A.CEDULA, A.NOMBRE, A.APELLIDO

```

Below the code is a results grid with three columns: CEDULA, NOMBRE, and APELLIDO. The data is as follows:

	CEDULA	NOMBRE	APELLIDO
1	209440300	JULIO	MONGE
2	409440300	MARIO	MORA
3	519440300	JULIA	DIAZ
4	779440300	MARIA	ARROLLO

En este caso se utiliza **RIGHT JOIN**, porque lo que se quiere es extraer la información de la tabla Persona.

Se pueden realizar consultas un poco más extensas, con varios JOIN, por ejemplo para en el caso de extraer todas aquellas personas que no tienen cursos matriculados, una forma de hacerlo sería la siguiente

```

SELECT A.CEDULA,A.NOMBRE, A.APELLIDO
FROM PERSONA A
LEFT JOIN HISTORIAL B
ON A.CEDULA = B.CEDULA
WHERE A.CEDULA NOT IN (SELECT B.CEDULA
FROM PERSONA A
RIGHT JOIN HISTORIAL B
ON A.CEDULA = B.CEDULA
GROUP BY B.CEDULA)
GROUP BY A.CEDULA,A.NOMBRE, A.APELLIDO

```

```
SELECT A.CEDULA, A.NOMBRE, A.APELLIDO
FROM PERSONA A
LEFT JOIN HISTORIAL B
ON A.CEDULA = B.CEDULA
WHERE A.CEDULA NOT IN (SELECT B.CEDULA
FROM PERSONA A
RIGHT JOIN HISTORIAL B
ON A.CEDULA = B.CEDULA
GROUP BY B.CEDULA)
GROUP BY A.CEDULA, A.NOMBRE, A.APELLIDO
```

The screenshot shows a SQL query being run in a Microsoft SQL Server Management Studio (SSMS) window. The query is designed to find people from the 'PERSONA' table who have no entries in the 'HISTORIAL' table. It uses a LEFT JOIN to include all rows from 'PERSONA' and a WHERE clause to filter out rows where the 'CEDULA' is found in the 'B' table. An alternative RIGHT JOIN approach is shown in the WHERE clause's subquery, which yields the same result. The results are grouped by 'CEDULA', 'NOMBRE', and 'APELLIDO'. The results grid displays four rows of data:

	CEDULA	NOMBRE	APELLIDO
1	209440300	JULIO	MONGE
2	409440300	MARIO	MORA
3	519440300	JULIA	DIAZ
4	779440300	MARIA	ARROLLO

Como se puede ver el resultado es el mismo

Práctica# 5 para la casa

Objetivo: Medir el conocimiento de los estudiantes, según los temas vistos en clase

Desarrolle los siguientes puntos:

1. Creación de la siguiente base de datos en SQL Server, el nombre de la base de datos es choferes.

Tablas

Chofer

Num_chofer int,
 Nombre varchar(30)
 Dirección varchar(50)
 Fecha-inicio datetime
 Salario float

Camión

Num_camion int
 Marca varchar(20)
 Tipo varchar(20)
 Año int
 Capacidad int

Envío

Num_chofer int
 Num_camion int
 Fecha datetime
 Peso float

Recuerde definir llaves primarias y foráneas según lo visto en clase

2. Insertar 5 registros en cada una de las entidades con los siguientes datos

Num chofer	Nombre	Dirección	Fecha inicio	Salario
1	Luis	San José	10/01/1996	120000
2	Samuel	Heredia	25/10/1994	155000
3	Jesús	San José	20/05/2001	110000
4	Marcos	Majuela	15/02/2000	120200
5	Miguel	Heredia	30/06/1999	140000

Num	Marca	Tipo	Año	Capacidad

camión				
11	Hino	Cisterna	1995	10
12	Hiace	Pick up	1998	4
13	Hino	Cisterna	2000	15
14	Ford	Pick up	1999	10
15	Hino	Carga	1998	5

3. Incluya envíos basado en estas tablas donde las fechas oscilen entre 01/01/1994 al 30/01/2002

Para este punto se van a incluir los siguientes datos

NUM_CHOFE	NUM_CAMION	FECHA	PESO
1	11	01/01/1994	25000
1	12	07/16/1995	25600
2	13	12/23/1996	75000
2	14	01/30/1997	25800
3	15	12/03/2000	15900
3	11	11/25/2001	45800
4	12	06/16/2000	23000
4	13	09/13/2001	45100
5	14	05/19/2002	63200
5	11	01/20/2002	44500
5	13	01/22/2002	44500
5	12	01/23/2002	44500
5	15	01/24/2002	44500
4	12	01/24/2002	44500
4	13	01/24/2002	44500
4	11	05/18/2002	43500
4	15	07/18/2002	25500
4	13	08/18/2002	20500

4. Responder en SQL a las siguientes consultas:

- Dar una lista de los nombres de choferes que viven en San José y que tienen un salario superior de ₡110000.
- Dar una lista de los camiones cisternas que hicieron un trayecto antes del 5 de octubre de 2001

- c. Dar una lista de los choferes que ingresaron después del 1 de enero de 1999 y que hicieron un envío superior de 20 toneladas en un camión Hino en el año 2000 Y 2001
- d. Dar la lista de los camiones agrupados por marca y año
- e. Dar una lista de los nombres de los choferes que han hecho envío con todos los camiones de la compañía.

Nota: para dar la solución a esta consulta favor investigar en la ayuda de SQL sobre la función CONVERT, se necesita de count , el distinct .

Tomar en cuenta que la tabla ENVIO no cuenta con llave primaria por lo que tanto el chofer como el camión se pueden repetir

Se puede realizar una concatenación de campos para lo cual va utilizar +”+

De la siguiente manera

SELECT EXPRESION CHAR +”+EXPRESION CHAR; por lo que necesitara de la función CONVERT, esta se vio en clase.

- f. Dar una lista del peso total de la mercadería transportada en 2002.
- g. Crear una tabla de los nombres de los choferes que usaron un camión Hino durante el año 2000, la nueva tabla debe tener el nombre del chofer, la marca y el tipo del camión.
- h. Crear una tabla de los camiones que son del año 1998 y que además realizaron un envío durante 2001, la nueva tabla debe tener Marca, Tipo , fecha de envío y año .

Práctica # 6 Práctica General para examen

Objetivo: Evacuar las dudas que los estudiantes con respecto a los temas vistos en clase antes de la realización del examen para el curso de SQL I.

Crear una base de datos llamada Turismo, que contiene las siguientes tablas:

Turista

Numero_turista	Int NOT NULL (PK)
Nombre	varchar(30)
APELLIDO_1	VARCHAR (30)
APELLIDO_2	VARCHAR(30)
Pais	varchar(20)
Edad	Int

Lugar

Codigo_lugar	int (PK)
Nombre	varchar(30)
Tipo_lugar	varchar(20)
Continente	varchar(7)

Viaje

Num_viaje	int
Num_turista	int (Fk)
Cod_lugar	int (Fk)
Fecha_salida	datetime
Fecha_llegada	datetime
Ciudad_salida	varchar(20)

Todos los campos son NOT NULL a excepción del APELLIDO_2

Realice las siguientes operaciones en la base de datos creada:

1. Adicionar una nueva columna en la tabla de VIAJE que se llama Estadía.
2. Suprimir una columna fecha_llegada en la tabla viaje.
3. Insertar 10 turistas en la tabla turistas, validar por medio de un constraint que ningún turista puede tener menos de 5 años ni más de 99 y que si el segundo apellido queda en blanco escribir NO LO DIO.
4. Insertar 10 registro en la tabla Lugar, asignando códigos de 10 en 10 (usar todos los continentes: América, Asia, África, Europa, Oceanía).
5. Usar la instrucción para actualizar, de tal forma que se deben incrementar los códigos de lugar según el continente de la siguiente forma:

Continente	Codigo_lugar
América	Codigo_lugar + 100
Africa	Codigo_lugar + 200

Asia	Codigo_lugar + 300
Europa	Codigo_lugar + 400
Oceanía	Codigo_lugar + 500

6. Eliminar los registros de la tabla cuyo continente es Oceanía.
7. Dar la lista de los nombre de todos los turistas.
8. Dar la lista de los diferentes tipos de lugar y de los continentes en donde se encuentran.
9. Dar la lista de los turistas que son costarricense, canadienses, panameños o jamaiquinos.
10. Dar el código de lugar más grande
11. Dar el número de sitios diferentes que pueden visitarse.
12. Para cada continente, dar una lista por continente y el código de lugares localizados en ese continente.
13. Dar la lista de aquellos tipos de lugar de los cuales existen al menos 3 del mismo tipo.
14. Dar la lista de los códigos de lugar y sus nombres en orden descendente.
15. Adicionar como llave primaria los campos NUMERO_VIAJE y NUMERO_TURISTA como llave primaria de la tabla VIAJE.
16. Inserte 4 registros en la tabla VIAJE utilizando como referencia la información de las tablas TURISTA y LUGAR.
17. Muestre los turistas que se encuentran de viaje y los que no están de viaje, utilice join y select anidado respectivamente.

Nota : Solución [Anexo VI](#)