

## **Práctica Funciones SQL Server**

En primer lugar, crearemos una base de datos llamada **escuela**, y dentro de ella, las siguientes tablas:

```
CREATE TABLE Estudiantes (  
    EstudianteID INT,  
    Nombre NVARCHAR(100),  
    Edad INT,  
    Carrera NVARCHAR(100),  
    Primary key (EstudianteID)  
);
```

```
CREATE TABLE Cursos (  
    CursoID INT,  
    Nombre NVARCHAR(100),  
    Creditos INT,  
    Profesor NVARCHAR(100),  
    Primary key (CursoID)  
);
```

```
CREATE TABLE Inscripciones (  
    InscripcionID INT,  
    EstudianteID INT,  
    CursoID INT,  
    Primary key (InscripcionID),  
    FOREIGN KEY (EstudianteID) REFERENCES Estudiantes(EstudianteID),  
    FOREIGN KEY (CursoID) REFERENCES Cursos(CursoID)  
);
```

A continuación, insertaremos los siguientes datos:

```
INSERT INTO Estudiantes (EstudianteID, Nombre, Edad, Carrera) VALUES (1, 'Juan Pérez', 20,  
'Ingeniería');  
  
INSERT INTO Estudiantes (EstudianteID, Nombre, Edad, Carrera) VALUES (2, 'María García', 22,  
'Medicina');
```

```
INSERT INTO Estudiantes (EstudianteID, Nombre, Edad, Carrera) VALUES (3, 'Carlos López', 21, 'Ingeniería');

INSERT INTO Estudiantes (EstudianteID, Nombre, Edad, Carrera) VALUES (4, 'Ana Martínez', 23, 'Medicina');

INSERT INTO Estudiantes (EstudianteID, Nombre, Edad, Carrera) VALUES (5, 'Luis Rodríguez', 24, 'Ingeniería');


INSERT INTO Cursos (CursoID, Nombre, Creditos, Profesor) VALUES (1, 'Introducción a la Programación', 4, 'Profesor A');

INSERT INTO Cursos (CursoID, Nombre, Creditos, Profesor) VALUES (2, 'Anatomía Humana', 3, 'Profesor B');

INSERT INTO Cursos (CursoID, Nombre, Creditos, Profesor) VALUES (3, 'Álgebra Lineal', 5, 'Profesor C');

INSERT INTO Cursos (CursoID, Nombre, Creditos, Profesor) VALUES (4, 'Medicina Interna', 4, 'Profesor D');

INSERT INTO Cursos (CursoID, Nombre, Creditos, Profesor) VALUES (5, 'Estructuras de Datos', 4, 'Profesor E');

INSERT INTO Inscripciones (InscripcionID, EstudianteID, CursoID) VALUES (1, 1, 1); -- Juan Pérez inscrito en Introducción a la Programación

INSERT INTO Inscripciones (InscripcionID, EstudianteID, CursoID) VALUES (2, 2, 2); -- María García inscrita en Anatomía Humana

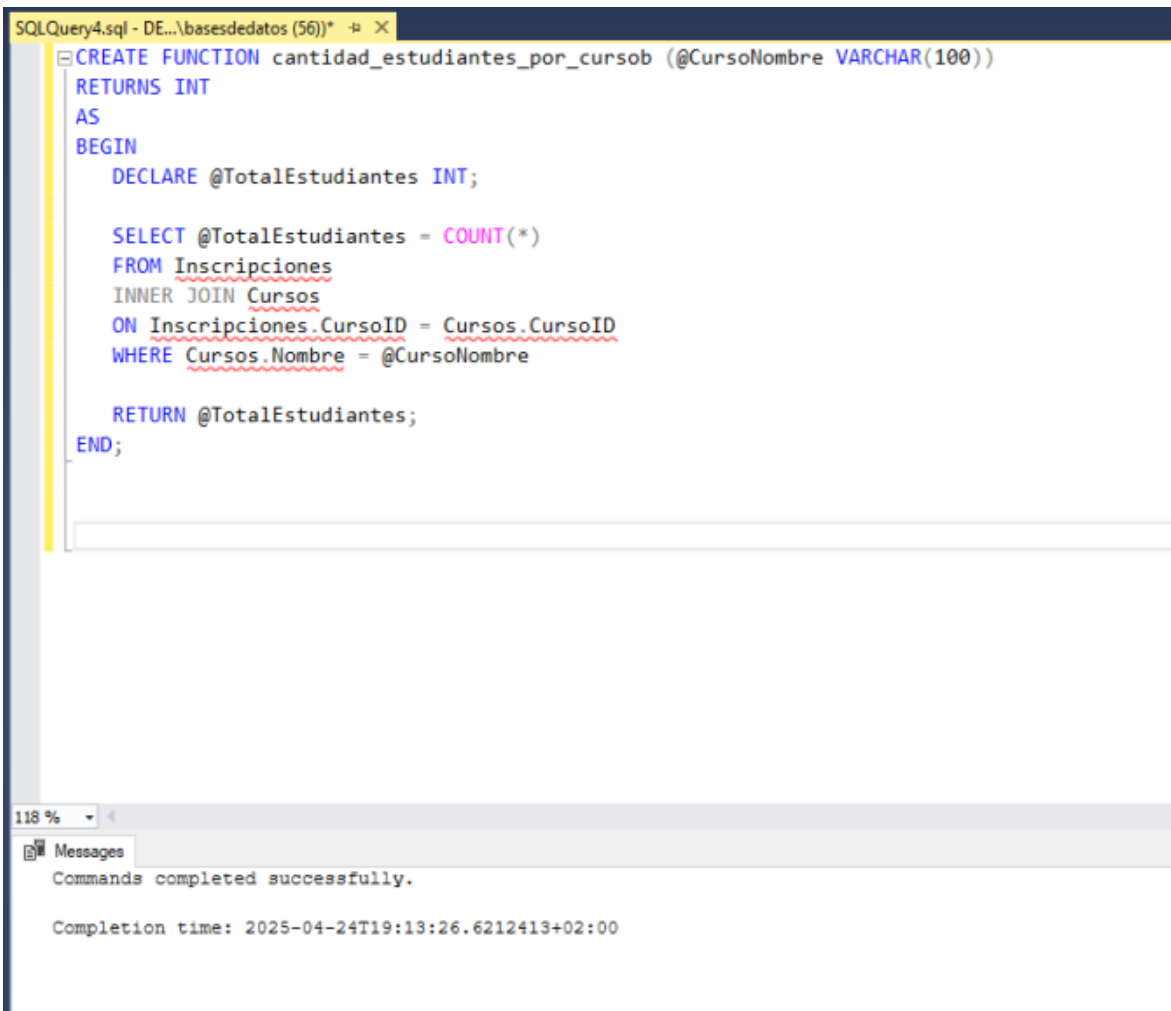
INSERT INTO Inscripciones (InscripcionID, EstudianteID, CursoID) VALUES (3, 3, 3); -- Carlos López inscrito en Álgebra Lineal

INSERT INTO Inscripciones (InscripcionID, EstudianteID, CursoID) VALUES (4, 4, 4); -- Ana Martínez inscrita en Medicina Interna

INSERT INTO Inscripciones (InscripcionID, EstudianteID, CursoID) VALUES (5, 5, 1); -- Luis Rodríguez inscrito en Introducción a la Programación
```

Por último, crearemos las funciones necesarias para obtener la siguiente información de la base de datos:

1. **Función para obtener la cantidad de estudiantes por curso:** Esta función recibe el nombre del curso y devuelve el número de estudiantes inscritos.



The screenshot displays the SQL Server Enterprise Manager interface. The top pane shows a SQL query window titled 'SQLQuery4.sql - DE...\basesdedatos (56))' containing the following T-SQL code:

```
CREATE FUNCTION cantidad_estudiantes_por_cursob (@CursoNombre VARCHAR(100))
RETURNS INT
AS
BEGIN
    DECLARE @TotalEstudiantes INT;

    SELECT @TotalEstudiantes = COUNT(*)
    FROM Inscripciones
    INNER JOIN Cursos
    ON Inscripciones.CursoID = Cursos.CursoID
    WHERE Cursos.Nombre = @CursoNombre

    RETURN @TotalEstudiantes;
END;
```

The bottom pane shows the 'Messages' tab with the following output:

```
Commands completed successfully.

Completion time: 2025-04-24T19:13:26.6212413+02:00
```

2. **Función para calcular la cantidad total de créditos de un estudiante:** Esta función recibe el id de un estudiante y calcula la suma de los créditos de los cursos en los que está inscrito.

```
SQLQuery1.sql - DE...\basesdedatos (64)) *  X
CREATE FUNCTION total_creditos_estudiante(@EstudianteID INT)
RETURNS INT
AS
BEGIN
    DECLARE @SumaCreditos INT;

    SELECT @SumaCreditos = SUM(Cursos.Creditos)
    FROM Inscripciones
    INNER JOIN Cursos ON Inscripciones.CursoID = Cursos.CursoID
    WHERE Inscripciones.EstudianteID = @EstudianteID;

    RETURN @SumaCreditos;
END;
```

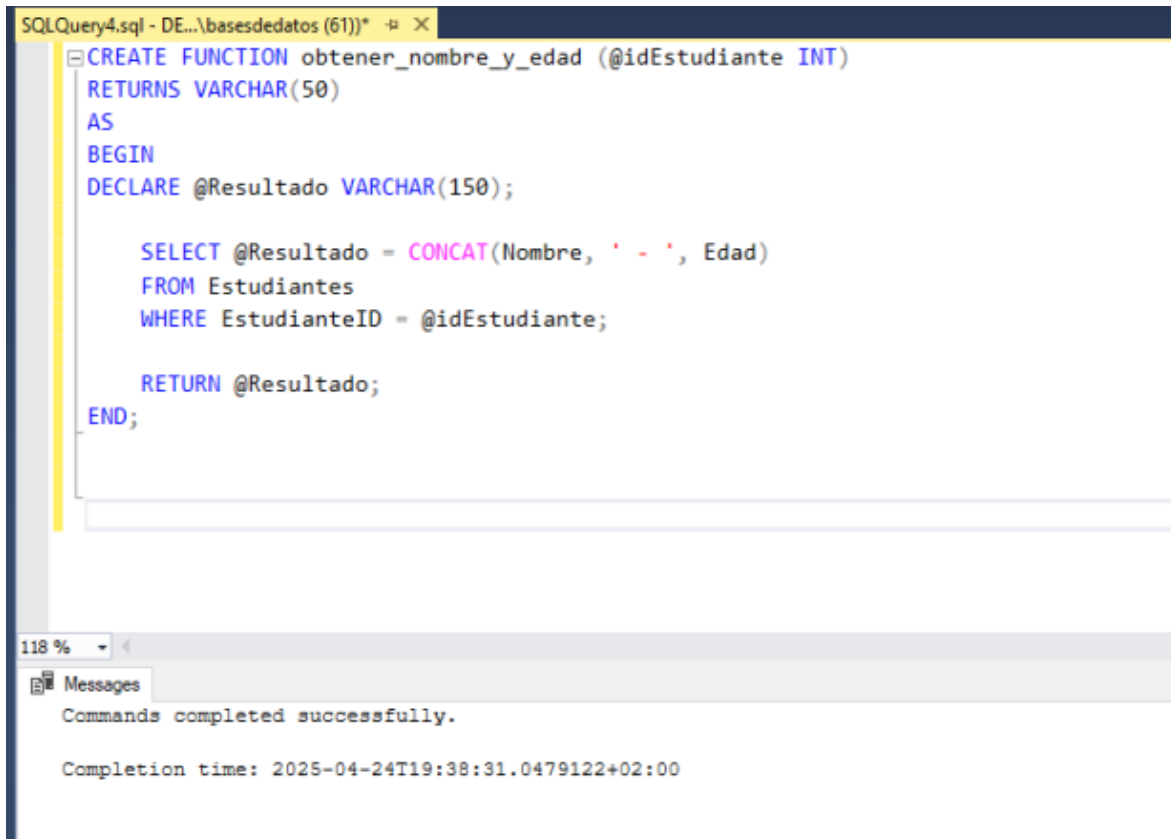
118 %

Messages

Commands completed successfully.

Completion time: 2025-04-24T19:26:15.2396374+02:00

**3. Función para obtener el nombre concatenado con la edad:** Esta función recibe el Id del estudiante y devuelve el nombre concatenado con la edad



The screenshot displays the SQL Server Enterprise Manager interface. The top pane shows a query window titled 'SQLQuery4.sql - DE...\basesdedatos (61))' containing the following T-SQL code to create a function:

```
CREATE FUNCTION obtener_nombre_y_edad (@idEstudiante INT)
RETURNS VARCHAR(50)
AS
BEGIN
    DECLARE @Resultado VARCHAR(150);

    SELECT @Resultado = CONCAT(Nombre, ' - ', Edad)
    FROM Estudiantes
    WHERE EstudianteID = @idEstudiante;

    RETURN @Resultado;
END;
```

The bottom pane, labeled 'Messages', shows the execution result: 'Commands completed successfully.' and the completion time: '2025-04-24T19:38:31.0479122+02:00'.

**4. Función para obtener el promedio de edad de los estudiantes por curso:** Esta función recibe el nombre del curso, y devuelve el promedio de edad de los estudiantes.

```
SQLQuery4.sql - DE...\basesdedatos (61)) * - + X
CREATE FUNCTION promedio_edad_por_curso(@CursoNombre VARCHAR(100))
RETURNS DECIMAL(10, 2)
AS
BEGIN
    DECLARE @Promedio DECIMAL(10, 2);

    SELECT @Promedio = AVG(Estudiantes.Edad)
    FROM Inscripciones
    INNER JOIN Cursos ON Inscripciones.CursoID = Cursos.CursoID
    INNER JOIN Estudiantes ON Inscripciones.EstudianteID = Estudiantes.EstudianteID
    WHERE Cursos.Nombre = @CursoNombre;

    RETURN @Promedio;
END;
```

118 %

Messages

Commands completed successfully.

Completion time: 2025-04-24T19:55:48.1519256+02:00

**5. Función para obtener los nombres de los alumnos inscritos en un curso:** Esta función recibe el id del curso y devuelve un listado con los nombres de los alumnos inscritos.

```
SQLQuery4.sql - DE...\basesdedatos (61)) * - + X
CREATE FUNCTION nombres_alumnos_por_curso (@CursoID INT)
RETURNS VARCHAR(MAX)
AS
BEGIN
    DECLARE @Nombres VARCHAR(MAX);

    SELECT @Nombres = CONCAT(Estudiantes.Nombre , ', ')
    FROM Inscripciones
    INNER JOIN Estudiantes ON Inscripciones.EstudianteID = Estudiantes.EstudianteID
    WHERE Inscripciones.CursoID = @CursoID;

    RETURN @Nombres;
END;
```

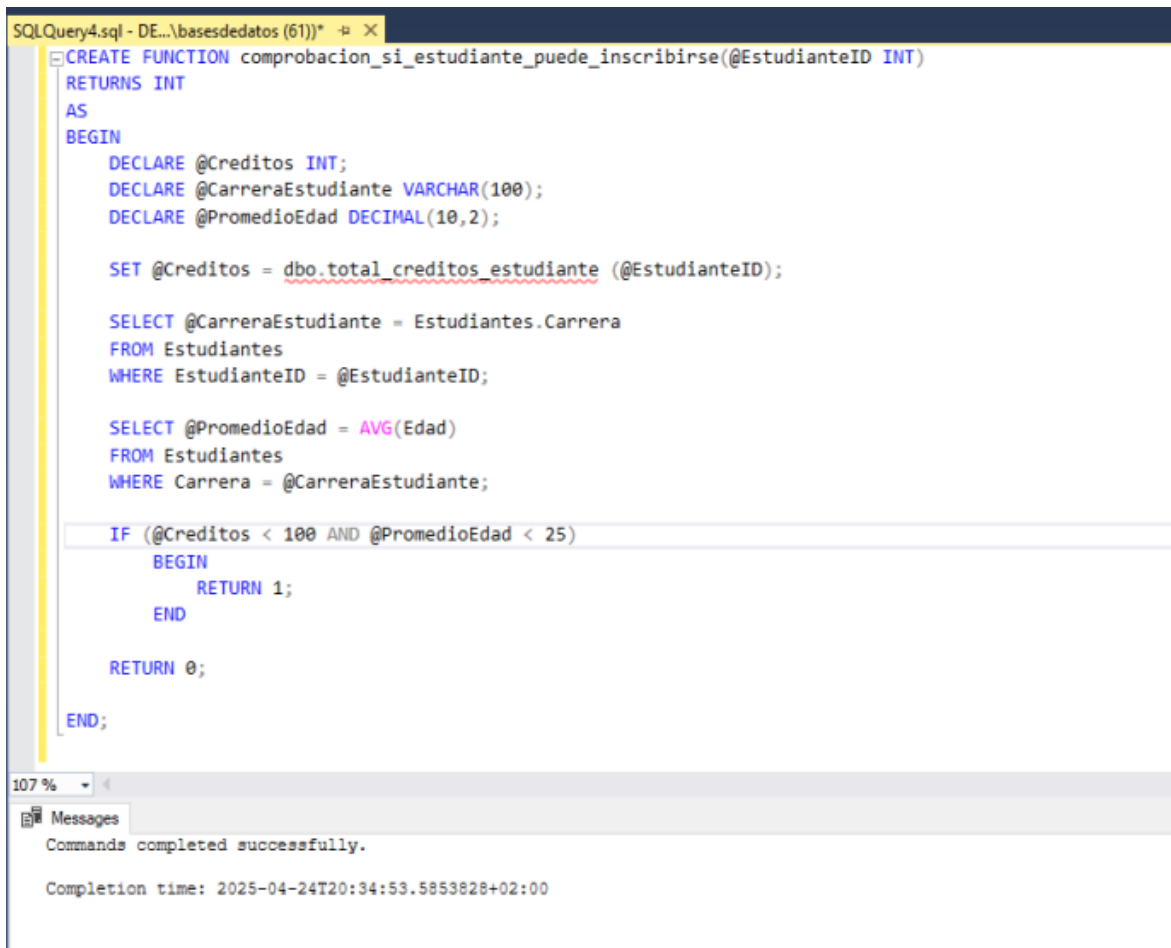
118 %

Messages

Commands completed successfully.

Completion time: 2025-04-24T20:02:25.1194887+02:00

6. Supongamos ahora que queremos crear una función que determine si un estudiante puede inscribirse en un nuevo curso según los siguientes criterios: que tenga menos de 100 créditos acumulados y que el promedio de edad de los estudiantes de su carrera sea menor de 25 años. La función devolverá 1 si el estudiante puede matricularse, o 0 en caso contrario.

The image is a screenshot of the SQL Server Enterprise Manager interface. The top pane shows a SQL query window titled 'SQLQuery4.sql - DE... \basesdedatos (61))' with a blue header bar. The query text is as follows:

```
CREATE FUNCTION comprobacion_si_estudiante_puede_inscribirse(@EstudianteID INT)
RETURNS INT
AS
BEGIN
    DECLARE @Credito INT;
    DECLARE @CarreraEstudiante VARCHAR(100);
    DECLARE @PromedioEdad DECIMAL(10,2);

    SET @Credito = dbo.total_creditos_estudiante (@EstudianteID);

    SELECT @CarreraEstudiante = Estudiantes.Carrera
    FROM Estudiantes
    WHERE EstudianteID = @EstudianteID;

    SELECT @PromedioEdad = AVG(Edad)
    FROM Estudiantes
    WHERE Carrera = @CarreraEstudiante;

    IF (@Credito < 100 AND @PromedioEdad < 25)
    BEGIN
        RETURN 1;
    END

    RETURN 0;
END;
```

The bottom pane shows the 'Messages' tab with the text 'Commands completed successfully.' and 'Completion time: 2025-04-24T20:34:53.5853828+02:00'. The zoom level is set to 107%.

Crear una función que permita realizar lo anterior y que use para ello, algunas de las otras funciones que hemos definido en los ejercicios previos.