

## Consultas avanzadas en SQL Server - Carlos José Torre García

### 1. Consultas de agregación

#### 1. Total de ingresos por curso (SUM)

```
SELECT c.NombreCurso, SUM(p.Monto) AS IngresosTotales
FROM Cursos c
JOIN Matriculas m ON c.CursoID = m.CursoID
JOIN Pagos p ON m.MatriculaID = p.MatriculaID
GROUP BY c.NombreCurso;
```

	NombreCurso	IngresosTotales
1	Arte Clásico	1800.00
2	Arte Moderno	800.00
3	Biología Celular	1700.00
4	Biología Molecular	700.00
5	Economía Avanzada	1900.00
6	Economía Básica	900.00
7	Estadística Aplicada	1100.00
8	Física Aplicada	500.00
9	Física Cuántica	1500.00

#### 2. Número de estudiantes por curso (COUNT)

```
SELECT c.NombreCurso, COUNT(DISTINCT m.EstudianteID) AS EstudiantesMatriculados
FROM Cursos c
JOIN Matriculas m ON c.CursoID = m.CursoID
GROUP BY c.NombreCurso;
```

	NombreCurso	EstudiantesMatriculados
1	Arte Clásico	1
2	Arte Moderno	1
3	Biología Celular	1
4	Biología Molecular	1
5	Economía Avanzada	1
6	Economía Básica	1
7	Estadística Aplicada	1
8	Física Aplicada	1

#### 3. Edad promedio de los estudiantes (AVG)

```
SELECT AVG(DATEDIFF(YEAR, e.FechaNacimiento, GETDATE())) AS EdadPromedio
FROM Estudiantes e;
```

	EdadPromedio
1	34

#### 4. Fecha de pago más temprana y más tardía (MIN, MAX)

```
SELECT MIN(p.FechaPago) AS FechaPrimerPago, MAX(p.FechaPago) AS FechaUltimoPago
FROM Pagos p;
```

	FechaPrimerPago	FechaUltimoPago
1	2025-01-01	2025-01-20

## 5. Monto total de pagos (SUM)

```
SELECT SUM(p.Monto) AS MontoTotalPagado
FROM Pagos p;
```

	MontoTotalPagado
1	21000.00

## 2. Agrupacion de datos

## 6. Ejemplo de por Estudiantes por grupo de edad:

```
SELECT CASE
    WHEN DATEDIFF(YEAR, e.FechaNacimiento, GETDATE()) BETWEEN 18 AND 25 THEN '18-25'
    WHEN DATEDIFF(YEAR, e.FechaNacimiento, GETDATE()) BETWEEN 26 AND 35 THEN '26-35'
    WHEN DATEDIFF(YEAR, e.FechaNacimiento, GETDATE()) BETWEEN 36 AND 45 THEN '36-45'
    ELSE '46+' END AS RangoEdad,
    COUNT(e.EstudianteID) AS NumeroEstudiantes
FROM Estudiantes e
GROUP BY CASE
    WHEN DATEDIFF(YEAR, e.FechaNacimiento, GETDATE()) BETWEEN 18 AND 25 THEN '18-25'
    WHEN DATEDIFF(YEAR, e.FechaNacimiento, GETDATE()) BETWEEN 26 AND 35 THEN '26-35'
    WHEN DATEDIFF(YEAR, e.FechaNacimiento, GETDATE()) BETWEEN 36 AND 45 THEN '36-45'
```

	RangoEdad	NumeroEstudiantes
1	18-25	1
2	26-35	10
3	36-45	9

### 3. SUBCONSULTAS

#### 7. Ejemplo de Estudiantes que han pagado más de 1000 soles

```
go
SELECT e.Nombres + ' ' + e.Apellidos AS Estudiante
FROM Estudiantes e
WHERE e.EstudianteID IN (
    SELECT m.EstudianteID
    FROM Matriculas m
    JOIN Pagos p ON m.MatriculaID = p.MatriculaID
    GROUP BY m.EstudianteID
    HAVING SUM(p.Monto) > 1000
)
```

68 %

Results Messages

	Estudiante
1	Miguel Fernandez
2	Lucia Torres
3	Juan Diaz
4	Carmen Sanchez
5	Pedro Jimenez
6	Elena Ruiz
7	Fernando Morales
8	Patricia Ortiz
9	Jorge Castro
10	Isabel Vega

### 4. CONSULTAS DE UNION:

Tipos de uniones (INNER JOIN, LEFT JOIN, RIGHT JOIN, FULL JOIN)

#### 8. INNER JOIN

Obtiene los estudiantes junto con los cursos en los que están matriculados.

```
go
SELECT E.Nombres, E.Apellidos, C.NombreCurso
FROM Estudiantes E
INNER JOIN Cursos C ON E.EstudianteID = M.EstudianteID
INNER JOIN Matriculas M ON M.MatriculaID = C.MatriculaID
```

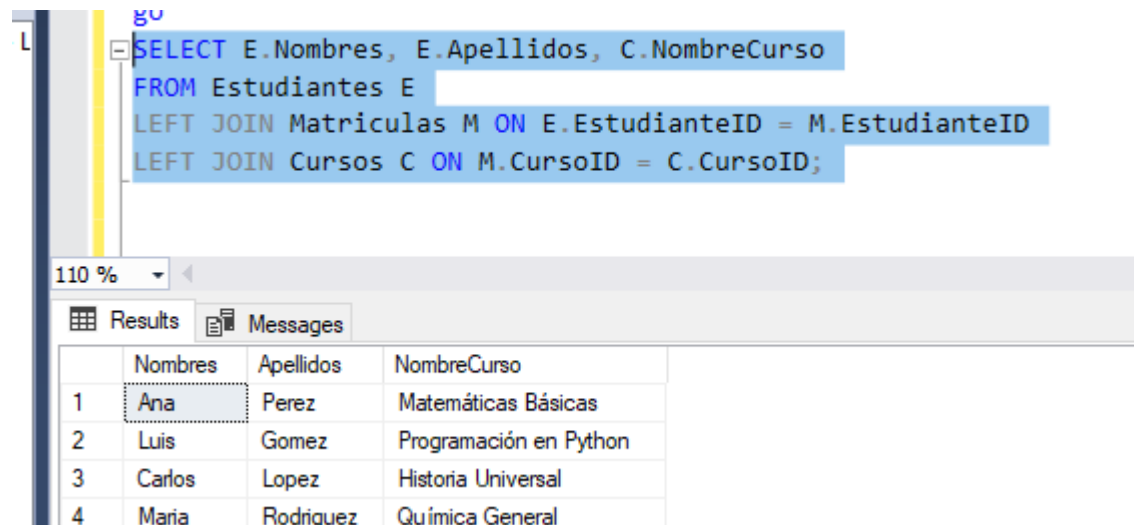
110 %

Results Messages

	Nombres	Apellidos	NombreCurso
1	Ana	Perez	Matemáticas Básicas
2	Luis	Gomez	Programación en Python
3	Carlos	Lopez	Historia Universal
4	Marta	Rodriguez	Química General
5	Jose	Martinez	Física Aplicada
6	Laura	Garcia	Literatura Española
7	David	Hernandez	Biología Molecular
8	Sofia	Lopez	Arte Moderno
9	Daniel	Gonzalez	Economía Básica
10	Andrea	Ramirez	Geografía Mundial
11	Miguel	Fernandez	Estadística Aplicada

## 9. LEFT JOIN

Muestra todos los estudiantes y los cursos en los que están matriculados, incluyendo a aquellos que no están matriculados en ningún curso.



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

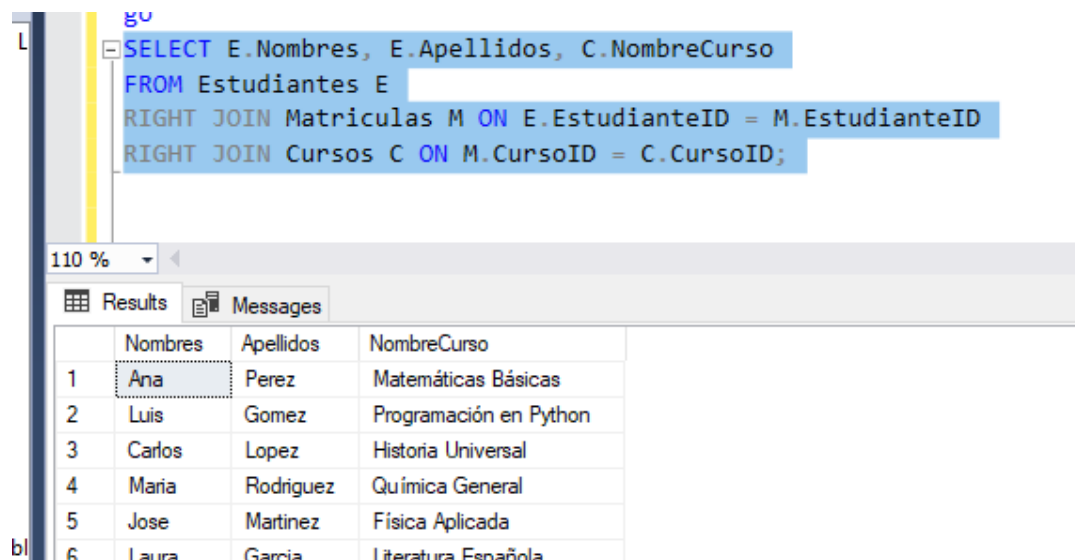
```
SELECT E.Nombres, E.Apellidos, C.NombreCurso
FROM Estudiantes E
LEFT JOIN Matriculas M ON E.EstudianteID = M.EstudianteID
LEFT JOIN Cursos C ON M.CursoID = C.CursoID;
```

The query is executed, and the results are displayed in a table with the following data:

	Nombres	Apellidos	NombreCurso
1	Ana	Perez	Matemáticas Básicas
2	Luis	Gomez	Programación en Python
3	Carlos	Lopez	Historia Universal
4	Maria	Rodriguez	Química General

## 10. RIGHT JOIN

Muestra todos los cursos y los estudiantes matriculados en ellos, incluyendo aquellos cursos sin estudiantes matriculados.



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

```
SELECT E.Nombres, E.Apellidos, C.NombreCurso
FROM Estudiantes E
RIGHT JOIN Matriculas M ON E.EstudianteID = M.EstudianteID
RIGHT JOIN Cursos C ON M.CursoID = C.CursoID;
```

The query is executed, and the results are displayed in a table with the following data:

	Nombres	Apellidos	NombreCurso
1	Ana	Perez	Matemáticas Básicas
2	Luis	Gomez	Programación en Python
3	Carlos	Lopez	Historia Universal
4	Maria	Rodriguez	Química General
5	Jose	Martinez	Física Aplicada
6	Laura	Garcia	Literatura Española

## 11. FULL JOIN

Muestra todos los estudiantes y todos los cursos, combinando los registros cuando hay coincidencias.

```
go
- L SELECT E.Nombres, E.Apellidos, C.NombreCurso
FROM Estudiantes E
FULL JOIN Matriculas M ON E.EstudianteID = M.EstudianteID
FULL JOIN Cursos C ON M.CursoID = C.CursoID;
```

110 %

Results Messages

	Nombres	Apellidos	NombreCurso
1	Ana	Perez	Matemáticas Básicas
2	Luis	Gomez	Programación en Python
3	Carlos	Lopez	Historia Universal
4	Maria	Rodriguez	Química General
5	Jose	Martinez	Física Aplicada
6	Laura	Garcia	Literatura Española

## 12. MULTIPLES UNIONES

Obtenemos los estudiantes, los cursos en los que están matriculados y los docentes que imparten esos cursos.

```
SELECT
E.Nombres AS Estudiante,
E.Apellidos AS Apellidos,
C.NombreCurso,
D.Nombres AS Docente
FROM
Estudiantes E
INNER JOIN
Matriculas M ON E.EstudianteID = M.EstudianteID
INNER JOIN
Cursos C ON M.CursoID = C.CursoID
LEFT JOIN
Docentes D ON C.DocenteID = D.DocenteID;
```

62 %

Results Messages

	Estudiante	Apellidos	NombreCurso	Docente
1	Ana	Perez	Matemáticas Básicas	Carlos
2	Luis	Gomez	Programación en Python	Ana
3	Carlos	Lopez	Historia Universal	Luis
4	Maria	Rodriguez	Química General	Maria
5	Jose	Martinez	Física Aplicada	Jose
6	Laura	Garcia	Literatura Española	Laura

## 6. SUBCONSULTA DE UNIONES

### 13. Subconsulta en la cláusula SELECT

Calcula el número de estudiantes matriculados en cada curso.

```
go
SELECT C.NombreCurso,
       (SELECT COUNT(*) FROM Matriculas M WHERE M.CursoID = C.CursoID) AS NumEstudiantes
FROM Cursos C;
```

1 %

Results Messages

	NombreCurso	NumEstudiantes
1	Matemáticas Básicas	1
2	Programación en Python	1
3	Historia Universal	1
4	Química General	1
5	Física Aplicada	1

### 14. Subconsulta en la cláusula WHERE

Obtiene los estudiantes que están matriculados en cursos con más de 10 semanas de duración.

```
go
SELECT E.Nombres, E.Apellidos
FROM Estudiantes E
WHERE E.EstudianteID IN (
    SELECT M.EstudianteID
    FROM Matriculas M
    INNER JOIN Cursos C ON M.CursoID = C.CursoID
    WHERE C.DuracionSemanas > 10
)
```

91 %

Results Messages

	Nombres	Apellidos
1	Luis	Gomez
2	Jose	Martinez
3	Sofia	Lopez
4	Miguel	Fernandez
5	Pedro	Jimenez
6	Patricia	Ortiz

## 15. Subconsulta correlacionada

Obtiene los estudiantes que han pagado más que el promedio de todos los pagos.

```
SELECT E.Nombres, E.Apellidos
FROM Estudiantes E
WHERE EXISTS (
    SELECT 1
    FROM Matriculas M
    INNER JOIN Pagos P ON M.MatriculaID = P.MatriculaID
    WHERE M.EstudianteID = E.EstudianteID
    GROUP BY M.EstudianteID
    HAVING SUM(P.Monto) > (SELECT AVG(P.Monto) FROM Pagos P)
```

91 %

Results Messages

	Nombres	Apellidos
1	Miguel	Fernandez
2	Lucia	Torres
3	Juan	Diaz
4	Carmen	Sanchez
5	Pedro	Jimenez

## 16. Consultas con funciones de ventana

### Funciones Over:

Las funciones de ventana permiten realizar cálculos sobre un conjunto de filas relacionadas sin colapsar los resultados. Se utilizan dentro de la cláusula **OVER()**. Ejemplos comunes incluyen **ROW\_NUMBER()**, **RANK()**, **DENSE\_RANK()**, **SUM()**, **AVG()**, entre otras.

```
SELECT
    E.Nombres,
    E.Apellidos,
    C.NombreCurso,
    ROW_NUMBER() OVER (ORDER BY E.Apellidos) AS Fila
FROM
    Estudiantes E
    INNER JOIN Matriculas M ON E.EstudianteID = M.EstudianteID
    INNER JOIN Cursos C ON M.CursoID = C.CursoID;
```

83 %

Results Messages

	Nombres	Apellidos	NombreCurso	Fila
1	Jorge	Castro	Economía Avanzada	1
2	Juan	Diaz	Historia del Arte	2
3	Miguel	Fernandez	Estadística Aplicada	3
4	Laura	Garcia	Literatura Española	4
5	Luis	Gomez	Programación en Python	5
6	Daniel	Gonzalez	Economía Básica	6
7	David	Hernandez	Biología Molecular	7
8	Pedro	Jimenez	Física Cuántica	8
9	Sofia	Lopez	Arte Moderno	9

## 17 Particiones de ventana

La cláusula **PARTITION BY** divide el conjunto de resultados en particiones a las que se aplica la función de ventana de forma independiente.

```
SELECT
    E.Nombres,
    E.Apellidos,
    C.NombreCurso,
    DENSE_RANK() OVER (PARTITION BY C.NombreCurso ORDER BY E.Apellidos) AS Rango
FROM
    Estudiantes E
    INNER JOIN Matriculas M ON E.EstudianteID = M.EstudianteID
    INNER JOIN Cursos C ON M.CursoID = C.CursoID;
```

83 %

Results Messages

	Nombres	Apellidos	NombreCurso	Rango
1	Patricia	Ortiz	Arte Clásico	1
2	Sofia	Lopez	Arte Moderno	1
3	Fernando	Morales	Biología Celular	1
4	David	Hernandez	Biología Molecular	1
5	Jorge	Castro	Economía Avanzada	1
6	Daniel	Gonzalez	Economía Básica	1

## 18 Marcos de ventana

Los marcos de ventana definen un subconjunto de filas dentro de la partición para aplicar la función de ventana. Se utilizan con la cláusula **ROWS BETWEEN**.

```
SELECT
    E.Nombres,
    E.Apellidos,
    C.NombreCurso,
    AVG(P.Monto) OVER (
        PARTITION BY C.NombreCurso
        ORDER BY P.FechaPago
        ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW
    ) AS PromedioAcumulado
FROM
    Estudiantes E
    INNER JOIN Matriculas M ON E.EstudianteID = M.EstudianteID
    INNER JOIN Cursos C ON M.CursoID = C.CursoID
    INNER JOIN Pagos P ON M.MatriculaID = P.MatriculaID;
```

83 %

Results Messages

	Nombres	Apellidos	NombreCurso	PromedioAcumulado
1	Patricia	Ortiz	Arte Clásico	1800.000000
2	Sofia	Lopez	Arte Moderno	800.000000
3	Fernando	Morales	Biología Celular	1700.000000
4	David	Hernandez	Biología Molecular	700.000000
5	Jorge	Castro	Economía Avanzada	1900.000000
6	Daniel	Gonzalez	Economía Básica	900.000000
7	Miguel	Fernandez	Estadística Aplicada	1100.000000
8	Jose	Martinez	Física Aplicada	500.000000