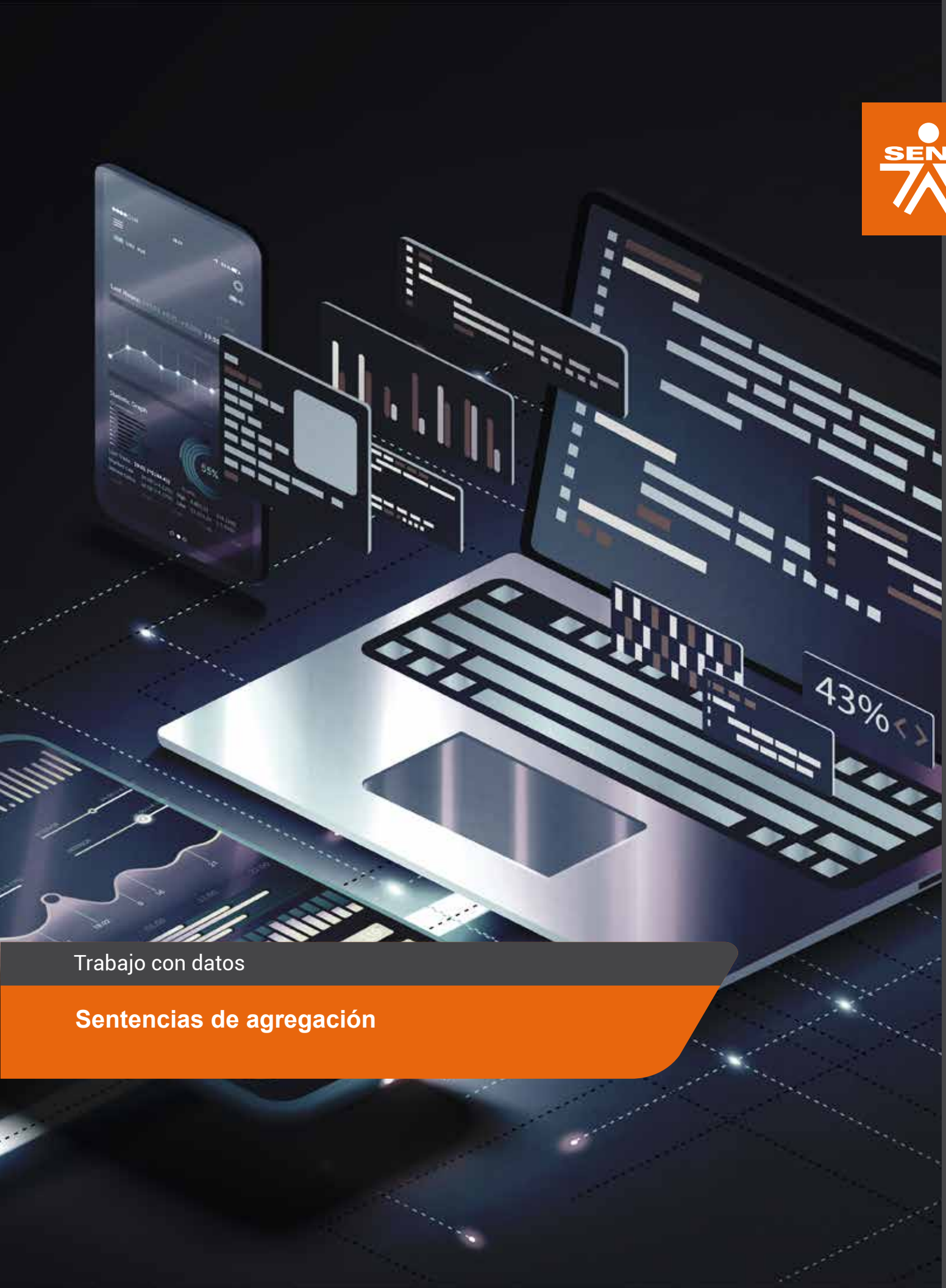


Trabajo con datos

# Sentencias de agregación

Servicio Nacional de Aprendizaje SENA

## Sentencias de agregación

Las sentencias de agregación nos permiten realizar cálculos específicos dentro de nuestra base de datos; en otras palabras, son funciones que ya se encuentran programadas en la base de datos y que solo hace falta llamarlas para aplicarlas en nuestra base de datos. Es importante que tengamos en cuenta que estas funciones no afectan los datos almacenados en las tablas de la base de datos, sino que utilizan los datos en modo de lectura, para poder mostrar al usuario el resultado esperado.

### max

Esta función realiza un recorrido de un campo de la tabla, teniendo en cuenta que trabaja con valores numéricos y permite establecer cuál es el valor con valor máximo de ese grupo de datos; por eso es necesario tener en cuenta, precisamente, ese aspecto que los valores tienen que ser de tipo numéricos, para que pueda funcionar de manera correcta. A continuación, veremos un ejemplo.

```
use dbs_Calificaciones
go
select * from tbl_Calificaciones
select max(tbl_Calificaciones.Definitiva) from tbl_Calificaciones
```

130 %

Results | Messages

| | PKN_Registro | FKId_tbl_Estudiante | Nota1 | Nota2 | Nota3 | Definitiva | FKCodigo_tbl_Estado | FKCodigo_tbl_Asignatura | Fecha |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 123 | 4,2 | 0 | 4,5 | 4,23 | 1 | ING3 | 2022-08-24 |
| 2 | 2 | 123 | 2,5 | 3 | 2 | 2,45 | 2 | ING1 | 2022-08-24 |
| 3 | 3 | 123 | 4,6 | 4,6 | 46 | 4,6 | 1 | ING4 | 2022-09-12 |
| 4 | 4 | 123 | 1,8 | 1,8 | 1,8 | 1,8 | 2 | ING2 | 2022-09-12 |

| | (No column name) |
|---|---|
| 1 | 4,6 |

### min

Esta función nos permite realizar un recorrido en un conjunto de valores almacenados en una tabla y establecer cuál es el valor mínimo que se encuentra en ese conjunto de datos; es importante tener en cuenta que también es necesario establecer que dichos valores son de tipo numérico para realizar de manera correcta su ejecución; a continuación, veremos un ejemplo.

```
use dbs_Calificaciones
go
select * from tbl_Calificaciones
select min(tbl_Calificaciones.Definitiva) from tbl_Calificaciones
```

130 %

Results | Messages

| | PKN_Registro | FKId_tbl_Estudiante | Nota1 | Nota2 | Nota3 | Definitiva | FKCodigo_tbl_Estado | FKCodigo_tbl_Asignatura | Fecha |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 123 | 4,2 | 0 | 4,5 | 4,23 | 1 | ING3 | 2022-08-24 |
| 2 | 2 | 123 | 2,5 | 3 | 2 | 2,45 | 2 | ING1 | 2022-08-24 |
| 3 | 3 | 123 | 4,6 | 4,6 | 46 | 4,6 | 1 | ING4 | 2022-09-12 |
| 4 | 4 | 123 | 1,8 | 1,8 | 1,8 | 1,8 | 2 | ING2 | 2022-09-12 |

| | (No column name) |
|---|---|
| 1 | 1,8 |

## avg

La función AVG permite calcular el promedio general de un conjunto de datos, es decir, realiza la sumatoria de los valores y luego los divide por la cantidad de registros que coincidan con la condición o se encuentren en el conjunto de datos; esta instrucción es de suma importancia cuando se desean saber los valores promedios de cierta cantidad de datos; a continuación, veremos un ejemplo particular donde se desea saber el promedio de notas del estudiante con documento de identificación 123.

```
use dbs_Calificaciones
go
select * from tbl_Calificaciones
SELECT AVG(tbl_Calificaciones.Definitiva) FROM tbl_Calificaciones
```

130 %

Results | Messages

| | PKN_Registro | FKId_tbl_Estudiante | Nota1 | Nota2 | Nota3 | Definitiva | FKCodigo_tbl_Estado | FKCodigo_tbl_Asignatura | Fecha |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 123 | 4,2 | 0 | 4,5 | 4,23 | 1 | ING3 | 2022-08-24 |
| 2 | 2 | 123 | 2,5 | 3 | 2 | 2,45 | 2 | ING1 | 2022-08-24 |
| 3 | 3 | 123 | 4,6 | 4,6 | 46 | 4,6 | 1 | ING4 | 2022-09-12 |
| 4 | 4 | 123 | 1,8 | 1,8 | 1,8 | 1,8 | 2 | ING2 | 2022-09-12 |

| | (No column name) |
|---|---|
| 1 | 3,27 |

## group by

La función GROUP BY se utiliza cuando deseamos realizar una operación de agrupamiento, teniendo en cuenta la coincidencia de los datos que pueden llegar a ser iguales o del mismo tipo, dentro de los registros de la tabla; es decir, aquellos que se encuentran asociados al mismo campo que se desea mostrar; esta función es muy importante cuando se desea mostrar la asociación de los registros en una base de datos y adicional permite realizar operaciones de agregación adicionales como contar, sumar o multiplicar; a continuación, veremos un ejemplo de esto.

| codigo | nombre | rubro | precio | tipoplato |
|---|---|---|---|---|
| 1 | 1 | milanesa y fritas | 40000 | Principal |
| 2 | 2 | arroz primavera | 35000 | Principal |
| 3 | 3 | arroz con pollo | 15000 | Ejecutivo |
| 4 | 4 | Papas fritas | 2000 | Acompañamiento |
| NULL | NULL | NULL | NULL | NULL |

```
use dbs_ejemploCrossJoin
go
SELECT COUNT(*), tipoplato
FROM comidas
GROUP BY tipoplato
```

130 %

Results | Messages

| | (No column name) | tipoplato |
|---|---|---|
| 1 | 1 | Acompañamiento |
| 2 | 1 | Ejecutivo |
| 3 | 2 | Principal |

## having

La cláusula HAVING se agregó a SQL, porque la palabra clave WHERE no se puede usar con funciones agregadas; para este caso específico, esta cláusula nos permite tener la posibilidad de agregar funciones que realicen cálculos dentro de las condiciones que se establezcan en la consulta; esto genera una ventaja al momento de realizar una condición amarrada a un proceso de cálculo de operación; a continuación, veremos un ejemplo de esto.

```
use dbs_ejemploCrossJoin
go
SELECT COUNT(codigo) as 'Cantidad de platos', tipoplato as 'Tipo de plato'
FROM comidas
GROUP BY tipoplato
HAVING COUNT(codigo) >= 1
```

130 %

Results | Messages

| | Cantidad de platos | Tipo de plato |
|---|---|---|
| 1 | 2 | Acompañamiento |
| 2 | 1 | Ejecutivo |
| 3 | 2 | Principal |

## order by

La función ORDER BY está relacionada con el ordenamiento de los datos que se encuentran dentro de una tabla; muchas veces se requiere que los datos que se muestran al usuario se encuentren con un orden específico, el cual puede ser de manera ascendente o descendente; es decir, de menor a mayor o de mayor a menor; en estos casos, se requiere que el campo que afecta a esta función debe ser un campo numérico para poder realizar el ordenamiento de manera correcta; a continuación, veremos un ejemplo de lo que planteamos a anteriormente.

```
use dbs_ejemploCrossJoin
go
select * from comidas
SELECT rubro, tipoplato
FROM comidas
order by precio ASC
```

130 %

Results | Messages

| | codigo | nombre | rubro | precio | tipoplato |
|---|---|---|---|---|---|
| 1 | 1 | 1 | milanesa y fritas | 40000 | Principal |
| 2 | 2 | 2 | arroz primavera | 35000 | Principal |
| 3 | 3 | 3 | arroz con pollo | 15000 | Ejecutivo |
| 4 | 4 | 4 | Papas fritas | 2000 | Acompañamiento |
| 5 | 5 | 5 | Ensalada | 1500 | Acompañamiento |

| | rubro | tipoplato |
|---|---|---|
| 1 | Ensalada | Acompañamiento |
| 2 | Papas fritas | Acompañamiento |
| 3 | arroz con pollo | Ejecutivo |
| 4 | arroz primavera | Principal |
| 5 | milanesa y fritas | Principal |

*Orden ascendente*

```
use dbs_ejemploCrossJoin
go
select * from comidas
SELECT rubro, tipoplato
FROM comidas
order by precio DESC
```

130 %

Results | Messages

| | codigo | nombre | rubro | precio | tipoplato |
|---|---|---|---|---|---|
| 1 | 1 | 1 | milanesa y fritas | 40000 | Principal |
| 2 | 2 | 2 | arroz primavera | 35000 | Principal |
| 3 | 3 | 3 | arroz con pollo | 15000 | Ejecutivo |
| 4 | 4 | 4 | Papas fritas | 2000 | Acompañamiento |
| 5 | 5 | 5 | Ensalada | 1500 | Acompañamiento |

| | rubro | tipoplato |
|---|---|---|
| 1 | milanesa y fritas | Principal |
| 2 | arroz primavera | Principal |
| 3 | arroz con pollo | Ejecutivo |
| 4 | Papas fritas | Acompañamiento |
| 5 | Ensalada | Acompañamiento |

*Orden descendente*