# Contar registros (count)

Existen en SQL Server funciones que nos permiten contar registros, calcular sumas, promedios, obtener valores máximos y mínimos. Estas funciones se denominan funciones de agregado y operan sobre un conjunto de valores (registros), no con datos individuales y devuelven un único valor.

Imaginemos que nuestra tabla "libros" contiene muchos registros. Para averiguar la cantidad sin necesidad de contarlos manualmente usamos la función "count()":

select count(\*)

from libros;

La función "count()" cuenta la cantidad de registros de una tabla, incluyendo los que tienen valor nulo.

También podemos utilizar esta función junto con la cláusula "where" para una consulta más específica. Queremos saber la cantidad de libros de la editorial "Planeta":

select count(\*)

from libros

where editorial='Planeta';

Para contar los registros que tienen precio (sin tener en cuenta los que tienen valor nulo), usamos la función "count()" y en los paréntesis colocamos el nombre del campo que necesitamos contar:

select count(precio)

from libros;

Note que "count(\*)" retorna la cantidad de registros de una tabla (incluyendo los que tienen valor "null") mientras que "count(precio)" retorna la cantidad de registros en los cuales el campo "precio" no es nulo. No es lo mismo. "count(\*)" cuenta registros, si en lugar de un asterisco colocamos como argumento el nombre de un campo, se contabilizan los registros cuyo valor en ese campo NO es nulo.

**Ejercicio:**

Trabajamos con la tabla "libros" de una librería.

Eliminamos la tabla si existe:

if object\_id ('libros') is not null

drop table libros;

Creamos la tabla:

create table libros(

codigo int identity,

titulo varchar(40) not null,

autor varchar(20) default 'Desconocido',

editorial varchar(20),

precio decimal(6,2),

primary key(codigo)

);

Ingresamos algunos registros:

insert into libros

values('El aleph','Borges','Emece',15.90);

insert into libros

values('Antología poética','J. L. Borges','Planeta',null);

insert into libros

values('Alicia en el pais de las maravillas','Lewis Carroll',null,19.90);

insert into libros

values('Matematica estas ahi','Paenza','Siglo XXI',15);

insert into libros

values('Martin Fierro','Jose Hernandez',default,40);

insert into libros

values('Aprenda PHP','Mario Molina','Nuevo siglo',null);

insert into libros

values('Uno','Richard Bach','Planeta',20);

Averiguemos la cantidad de libros usando la función "count()":

select count(\*)

from libros;

Note que incluye todos los libros aunque tengan valor nulo en algún campo.

Contamos los libros de editorial "Planeta":

select count(\*)

from libros

where editorial='Planeta';

Contamos los registros que tienen precio (sin tener en cuenta los que tienen valor nulo), usando la función "count(precio)":

select count(precio)

from libros;

**Código:**

if object\_id ('libros') is not null

drop table libros;

create table libros(

codigo int identity,

titulo varchar(40) not null,

autor varchar(20) default 'Desconocido',

editorial varchar(20),

precio decimal(6,2),

primary key(codigo)

);

insert into libros

values('El aleph','Borges','Emece',15.90);

insert into libros

values('Antología poética','J. L. Borges','Planeta',null);

insert into libros

values('Alicia en el pais de las maravillas','Lewis Carroll',null,19.90);

insert into libros

values('Matematica estas ahi','Paenza','Siglo XXI',15);

insert into libros

values('Martin Fierro','Jose Hernandez',default,40);

insert into libros

values('Aprenda PHP','Mario Molina','Nuevo siglo',null);

insert into libros

values('Uno','Richard Bach','Planeta',20);

select count(\*)

from libros;

select count(\*)

from libros

where editorial='Planeta';

select count(precio)

from libros;

**Problema:**

Trabaje con la tabla llamada "medicamentos" de una farmacia.

1- Elimine la tabla, si existe:

if object\_id('medicamentos') is not null

drop table medicamentos;

2- Cree la tabla con la siguiente estructura:

create table medicamentos(

codigo int identity,

nombre varchar(20),

laboratorio varchar(20),

precio decimal(6,2),

cantidad tinyint,

fechavencimiento datetime not null,

numerolote int default null,

primary key(codigo)

);

3- Ingrese algunos registros:

insert into medicamentos

values('Sertal','Roche',5.2,1,'2005-02-01',null);

insert into medicamentos

values('Buscapina','Roche',4.10,3,'2006-03-01',null);

insert into medicamentos

values('Amoxidal 500','Bayer',15.60,100,'2007-05-01',null);

insert into medicamentos

values('Paracetamol 500','Bago',1.90,20,'2008-02-01',null);

insert into medicamentos

values('Bayaspirina',null,2.10,null,'2009-12-01',null);

insert into medicamentos

values('Amoxidal jarabe','Bayer',null,250,'2009-12-15',null);

4- Muestre la cantidad de registros empleando la función "count(\*)" (6 registros)

5- Cuente la cantidad de medicamentos que tienen laboratorio conocido (5 registros)

6- Cuente la cantidad de medicamentos que tienen precio distinto a "null" y que tienen cantidad distinto a "null", disponer alias para las columnas.

7- Cuente la cantidad de remedios con precio conocido, cuyo laboratorio comience con "B" (2 registros)

8- Cuente la cantidad de medicamentos con número de lote distitno de "null" (0 registros)

**Solución:**

if object\_id('medicamentos') is not null

drop table medicamentos;

create table medicamentos(

codigo int identity,

nombre varchar(20),

laboratorio varchar(20),

precio decimal(6,2),

cantidad tinyint,

fechavencimiento datetime not null,

numerolote int default null,

primary key(codigo)

);

insert into medicamentos

values('Sertal','Roche',5.2,1,'2005-02-01',null);

insert into medicamentos

values('Buscapina','Roche',4.10,3,'2006-03-01',null);

insert into medicamentos

values('Amoxidal 500','Bayer',15.60,100,'2007-05-01',null);

insert into medicamentos

values('Paracetamol 500','Bago',1.90,20,'2008-02-01',null);

insert into medicamentos

values('Bayaspirina',null,2.10,null,'2009-12-01',null);

insert into medicamentos

values('Amoxidal jarabe','Bayer',null,250,'2009-12-15',null);

select count(\*)

from medicamentos;

select count(laboratorio)

from medicamentos;

select count(precio) as 'Con precio',

count(cantidad) as 'Con cantidad'

from medicamentos;

select count(precio)

from medicamentos

where laboratorio like 'B%';

select count(numerolote) from medicamentos;

# Contar registros (count\_big)

Retorna la cantidad de registros. Es similar a la función "count(\*)", la diferencia es que "count\_big" retorna un valor "bigint" y "count", un "int".

"count\_big(\*)" cuenta la cantidad de registros de una tabla, incluyendo los valores nulos y duplicados.

"count\_big(CAMPO)" retorna la cantidad de registros cuyo valor en el campo especificado entre paréntesis no es nulo.

"count\_big(distinct CAMPO)" retorna la cantidad de registros cuyo valor en el campo especificado no es nulo, sin considerar los repetidos.

Averiguemos la cantidad de libros usando la función "count\_big()":

select count\_big(\*)

from libros;

Note que incluye todos los libros aunque tengan valor nulo en algún campo.

Contamos los libros de editorial "Planeta":

select count\_big(\*)

from libros

where editorial='Planeta';

Contamos los registros que tienen precio (sin tener en cuenta los que tienen valor nulo):

select count\_big(precio)

from libros;

Contamos las editoriales (sin repetir):

select count\_big(distinct editorial)

from libros;

**Ejercicio:**

Trabajamos con la tabla "libros" de una librería.

Eliminamos la tabla si existe:

if object\_id ('libros') is not null

drop table libros;

Creamos la tabla:

create table libros(

codigo int identity,

titulo varchar(40) not null,

autor varchar(20) default 'Desconocido',

editorial varchar(20),

precio decimal(6,2),

primary key(codigo)

);

Ingresamos algunos registros:

insert into libros values('El aleph','Borges','Emece',15.90);

insert into libros values('Antología poética','Borges','Planeta',null);

insert into libros values('Alicia en el pais de las maravillas','Lewis Carroll',null,19.90);

insert into libros values('Matematica estas ahi','Paenza','Siglo XXI',15);

insert into libros values('Martin Fierro','Jose Hernandez',default,40);

insert into libros values('Aprenda PHP','Mario Molina','Nuevo siglo',null);

insert into libros values('Uno','Richard Bach','Planeta',20);

Contamos los libros usando "count\_big()":

select count\_big(\*)

from libros;

Note que incluye todos los libros aunque tengan valor nulo en algún campo.

Contamos los libros de editorial "Planeta":

select count\_big(\*)

from libros

where editorial='Planeta';

Averiguamos la cantidad de libros que tienen precio:

select count\_big(precio)

from libros;

Contamos las editoriales (sin repetir):

select count\_big(distinct editorial)

from libros;

**Código:**

if object\_id ('libros') is not null

drop table libros;

create table libros(

codigo int identity,

titulo varchar(40) not null,

autor varchar(20) default 'Desconocido',

editorial varchar(20),

precio decimal(6,2),

primary key(codigo)

);

insert into libros values('El aleph','Borges','Emece',15.90);

insert into libros values('Antología poética','Borges','Planeta',null);

insert into libros values('Alicia en el pais de las maravillas','Lewis Carroll',null,19.90);

insert into libros values('Matematica estas ahi','Paenza','Siglo XXI',15);

insert into libros values('Martin Fierro','Jose Hernandez',default,40);

insert into libros values('Aprenda PHP','Mario Molina','Nuevo siglo',null);

insert into libros values('Uno','Richard Bach','Planeta',20);

select count\_big(\*)

from libros;

select count\_big(\*)

from libros

where editorial='Planeta';

select count\_big(precio)

from libros;

select count\_big(distinct editorial)

from libros;

**Problema:**

Trabaje con la tabla llamada "medicamentos" de una farmacia.

1- Elimine la tabla, si existe:

if object\_id('medicamentos') is not null

drop table medicamentos;

2- Cree la tabla con la siguiente estructura:

create table medicamentos(

codigo int identity,

nombre varchar(20),

laboratorio varchar(20),

precio decimal(6,2),

cantidad tinyint,

fechavencimiento datetime not null,

numerolote int default null,

primary key(codigo)

);

3- Ingrese algunos registros:

insert into medicamentos

values('Sertal','Roche',5.2,1,'2005-02-01',null);

insert into medicamentos

values('Buscapina','Roche',4.10,3,'2006-03-01',null);

insert into medicamentos

values('Amoxidal 500','Bayer',15.60,100,'2007-05-01',null);

insert into medicamentos

values('Paracetamol 500','Bago',1.90,20,'2008-02-01',null);

insert into medicamentos

values('Bayaspirina',null,2.10,null,'2009-12-01',null);

insert into medicamentos

values('Amoxidal jarabe','Bayer',null,250,'2009-12-15',null);

4- Muestre la cantidad de registros empleando la función "count\_big(\*)" (6 registros)

5- Cuente la cantidad de laboratorios distintos (3 registros)

6- Cuente la cantidad de medicamentos que tienen precio y cantidad distinto de "null" (5 y 5)

**Solución:**

if object\_id('medicamentos') is not null

drop table medicamentos;

create table medicamentos(

codigo int identity,

nombre varchar(20),

laboratorio varchar(20),

precio decimal(6,2),

cantidad tinyint,

fechavencimiento datetime not null,

numerolote int default null,

primary key(codigo)

);

insert into medicamentos

values('Sertal','Roche',5.2,1,'2005-02-01',null);

insert into medicamentos

values('Buscapina','Roche',4.10,3,'2006-03-01',null);

insert into medicamentos

values('Amoxidal 500','Bayer',15.60,100,'2007-05-01',null);

insert into medicamentos

values('Paracetamol 500','Bago',1.90,20,'2008-02-01',null);

insert into medicamentos

values('Bayaspirina',null,2.10,null,'2009-12-01',null);

insert into medicamentos

values('Amoxidal jarabe','Bayer',null,250,'2009-12-15',null);

select count\_big(\*)

from medicamentos;

select count\_big(distinct laboratorio)

from medicamentos;

select count\_big(precio) as 'Con precio',

count\_big(cantidad) as 'Con cantidad'

from medicamentos;

# Funciones de agrupamiento (count - sum - min - max - avg)

Hemos visto que SQL Server tiene funciones que nos permiten contar registros, calcular sumas, promedios, obtener valores máximos y mínimos, las funciones de agregado.

Ya hemos aprendido una de ellas, "count()", veamos otras.

Se pueden usar en una instrucción "select" y combinarlas con la cláusula "group by".

Todas estas funciones retornan "null" si ningún registro cumple con la condición del "where", excepto "count" que en tal caso retorna cero.

El tipo de dato del campo determina las funciones que se pueden emplear con ellas.

Las relaciones entre las funciones de agrupamiento y los tipos de datos es la siguiente:

- count: se puede emplear con cualquier tipo de dato.

- min y max: con cualquier tipo de dato.

- sum y avg: sólo en campos de tipo numérico.

La función "sum()" retorna la suma de los valores que contiene el campo especificado. Si queremos saber la cantidad total de libros que tenemos disponibles para la venta, debemos sumar todos los valores del campo "cantidad":

select sum(cantidad)

from libros;

Para averiguar el valor máximo o mínimo de un campo usamos las funciones "max()" y "min()" respectivamente.

Queremos saber cuál es el mayor precio de todos los libros:

select max(precio)

from libros;

Entonces, dentro del paréntesis de la función colocamos el nombre del campo del cuál queremos el máximo valor.

La función "avg()" retorna el valor promedio de los valores del campo especificado. Queremos saber el promedio del precio de los libros referentes a "PHP":

select avg(precio)

from libros

where titulo like '%PHP%';

Ahora podemos entender porque estas funciones se denominan "funciones de agrupamiento", porque operan sobre conjuntos de registros, no con datos individuales.

Tratamiento de los valores nulos:

Si realiza una consulta con la función "count" de un campo que contiene 18 registros, 2 de los cuales contienen valor nulo, el resultado devuelve un total de 16 filas porque no considera aquellos con valor nulo.

Todas las funciones de agregado, excepto "count(\*)", excluye los valores nulos de los campos. "count(\*)" cuenta todos los registros, incluidos los que contienen "null".

**Ejercicio:**

Trabajamos con la tabla "libros" de una librería.

Eliminamos la tabla, si existe:

if object\_id('libros') is not null

drop table libros;

Creamos la tabla:

create table libros(

codigo int identity,

titulo varchar(40) not null,

autor varchar(30) default 'Desconocido',

editorial varchar(15),

precio decimal(5,2),

cantidad tinyint,

primary key(codigo)

);

Ingresamos algunos registros:

insert into libros

values('El aleph','Borges','Planeta',15,null);

insert into libros

values('Martin Fierro','Jose Hernandez','Emece',22.20,200);

insert into libros

values('Antologia poetica','J.L. Borges','Planeta',null,150);

insert into libros

values('Aprenda PHP','Mario Molina','Emece',18.20,null);

insert into libros

values('Cervantes y el quijote','Bioy Casares- J.L. Borges','Paidos',null,100);

insert into libros

values('Manual de PHP', 'J.C. Paez', 'Siglo XXI',31.80,120);

insert into libros

values('Harry Potter y la piedra filosofal','J.K. Rowling',default,45.00,90);

insert into libros

values('Harry Potter y la camara secreta','J.K. Rowling','Emece',46.00,100);

insert into libros (titulo,autor,cantidad)

values('Alicia en el pais de las maravillas','Lewis Carroll',220);

insert into libros (titulo,autor,cantidad)

values('PHP de la A a la Z',default,0);

Para conocer la cantidad total de libros, sumamos las cantidades de cada uno:

select sum(cantidad)

from libros;

Retorna 980; verifique la suma, sumando los valores de todos los registros del campo "cantidad".

Queremos saber cuántos libros tenemos de la editorial "Emece":

select sum(cantidad)

from libros

where editorial='Emece';

Retorna 300.

Queremos saber cuál es el libro más costoso; usamos la función "max()":

select max(precio)

from libros;

Retorna 46.00.

Para conocer el precio mínimo de los libros de "Rowling" tipeamos:

select min(precio)

from libros

where autor like '%Rowling%';

Retorna 45.00.

Queremos saber el promedio del precio de los libros referentes a "PHP":

select avg(precio)

from libros

where titulo like '%PHP%';

Devuelve 25.00. Note que hay 3 libros sobre "PHP", pero uno de ellos tiene precio nulo entonces SQL Server no lo incluye para calcular el promedio.

**Código:**

if object\_id('libros') is not null

drop table libros;

create table libros(

codigo int identity,

titulo varchar(40) not null,

autor varchar(30) default 'Desconocido',

editorial varchar(15),

precio decimal(5,2),

cantidad tinyint,

primary key(codigo)

);

insert into libros

values('El aleph','Borges','Planeta',15,null);

insert into libros

values('Martin Fierro','Jose Hernandez','Emece',22.20,200);

insert into libros

values('Antologia poetica','J.L. Borges','Planeta',null,150);

insert into libros

values('Aprenda PHP','Mario Molina','Emece',18.20,null);

insert into libros

values('Cervantes y el quijote','Bioy Casares- J.L. Borges','Paidos',null,100);

insert into libros

values('Manual de PHP', 'J.C. Paez', 'Siglo XXI',31.80,120);

insert into libros

values('Harry Potter y la piedra filosofal','J.K. Rowling',default,45.00,90);

insert into libros

values('Harry Potter y la camara secreta','J.K. Rowling','Emece',46.00,100);

insert into libros (titulo,autor,cantidad)

values('Alicia en el pais de las maravillas','Lewis Carroll',220);

insert into libros (titulo,autor,cantidad)

values('PHP de la A a la Z',default,0);

select sum(cantidad)

from libros;

select sum(cantidad)

from libros

where editorial='Emece';

select max(precio)

from libros;

select min(precio)

from libros

where autor like '%Rowling%';

select avg(precio)

from libros

where titulo like '%PHP%';

**Problema:**

Una empresa almacena los datos de sus empleados en una tabla "empleados".

1- Elimine la tabla, si existe:

if object\_id('empleados') is not null

drop table empleados;

2- Cree la tabla:

create table empleados(

nombre varchar(30),

documento char(8),

domicilio varchar(30),

seccion varchar(20),

sueldo decimal(6,2),

cantidadhijos tinyint,

primary key(documento)

);

3- Ingrese algunos registros:

insert into empleados

values('Juan Perez','22333444','Colon 123','Gerencia',5000,2);

insert into empleados

values('Ana Acosta','23444555','Caseros 987','Secretaria',2000,0);

insert into empleados

values('Lucas Duarte','25666777','Sucre 235','Sistemas',4000,1);

insert into empleados

values('Pamela Gonzalez','26777888','Sarmiento 873','Secretaria',2200,3);

insert into empleados

values('Marcos Juarez','30000111','Rivadavia 801','Contaduria',3000,0);

insert into empleados

values('Yolanda Perez','35111222','Colon 180','Administracion',3200,1);

insert into empleados

values('Rodolfo Perez','35555888','Coronel Olmedo 588','Sistemas',4000,3);

insert into empleados

values('Martina Rodriguez','30141414','Sarmiento 1234','Administracion',3800,4);

insert into empleados

values('Andres Costa','28444555',default,'Secretaria',null,null);

4- Muestre la cantidad de empleados usando "count" (9 empleados)

5- Muestre la cantidad de empleados con sueldo no nulo de la sección "Secretaria" (2 empleados)

6- Muestre el sueldo más alto y el más bajo colocando un alias (5000 y 2000)

7- Muestre el valor mayor de "cantidadhijos" de los empleados "Perez" (3 hijos)

8- Muestre el promedio de sueldos de todo los empleados (3400. Note que hay un sueldo nulo y no es tenido en cuenta)

9- Muestre el promedio de sueldos de los empleados de la sección "Secretaría" (2100)

10- Muestre el promedio de hijos de todos los empleados de "Sistemas" (2)

**Solución:**

if object\_id('empleados') is not null

drop table empleados;

create table empleados(

nombre varchar(30),

documento char(8),

domicilio varchar(30),

seccion varchar(20),

sueldo decimal(6,2),

cantidadhijos tinyint,

primary key(documento)

);

insert into empleados

values('Juan Perez','22333444','Colon 123','Gerencia',5000,2);

insert into empleados

values('Ana Acosta','23444555','Caseros 987','Secretaria',2000,0);

insert into empleados

values('Lucas Duarte','25666777','Sucre 235','Sistemas',4000,1);

insert into empleados

values('Pamela Gonzalez','26777888','Sarmiento 873','Secretaria',2200,3);

insert into empleados

values('Marcos Juarez','30000111','Rivadavia 801','Contaduria',3000,0);

insert into empleados

values('Yolanda Perez','35111222','Colon 180','Administracion',3200,1);

insert into empleados

values('Rodolfo Perez','35555888','Coronel Olmedo 588','Sistemas',4000,3);

insert into empleados

values('Martina Rodriguez','30141414','Sarmiento 1234','Administracion',3800,4);

insert into empleados

values('Andres Costa','28444555',default,'Secretaria',null,null);

select count(\*)

from empleados;

select count(sueldo)

from empleados

where seccion='Secretaria';

select max(sueldo) as 'Mayor sueldo',

min(sueldo) as 'Menor sueldo'

from empleados;

select max(cantidadhijos)

from empleados

where nombre like '%Perez%';

select avg(sueldo)

from empleados;

select avg(sueldo)

from empleados

where seccion='Secretaria';

select avg(cantidadhijos)

from empleados

where seccion='Sistemas';

# Agrupar registros (group by)

Hemos aprendido que las funciones de agregado permiten realizar varios cálculos operando con conjuntos de registros.

Las funciones de agregado solas producen un valor de resumen para todos los registros de un campo. Podemos generar valores de resumen para un solo campo, combinando las funciones de agregado con la cláusula "group by", que agrupa registros para consultas detalladas.

Queremos saber la cantidad de libros de cada editorial, podemos tipear la siguiente sentencia:

select count(\*) from libros

where editorial='Planeta';

y repetirla con cada valor de "editorial":

select count(\*) from libros

where editorial='Emece';

select count(\*) from libros

where editorial='Paidos';

...

Pero hay otra manera, utilizando la cláusula "group by":

select editorial, count(\*)

from libros

group by editorial;

La instrucción anterior solicita que muestre el nombre de la editorial y cuente la cantidad agrupando los registros por el campo "editorial". Como resultado aparecen los nombres de las editoriales y la cantidad de registros para cada valor del campo.

Los valores nulos se procesan como otro grupo.

Entonces, para saber la cantidad de libros que tenemos de cada editorial, utilizamos la función "count()", agregamos "group by" (que agrupa registros) y el campo por el que deseamos que se realice el agrupamiento, también colocamos el nombre del campo a recuperar; la sintaxis básica es la siguiente:

select CAMPO, FUNCIONDEAGREGADO

from NOMBRETABLA

group by CAMPO;

También se puede agrupar por más de un campo, en tal caso, luego del "group by" se listan los campos, separados por comas. Todos los campos que se especifican en la cláusula "group by" deben estar en la lista de selección.

select CAMPO1, CAMPO2, FUNCIONDEAGREGADO

from NOMBRETABLA

group by CAMPO1,CAMPO2;

Para obtener la cantidad libros con precio no nulo, de cada editorial utilizamos la función "count()" enviándole como argumento el campo "precio", agregamos "group by" y el campo por el que deseamos que se realice el agrupamiento (editorial):

select editorial, count(precio)

from libros

group by editorial;

Como resultado aparecen los nombres de las editoriales y la cantidad de registros de cada una, sin contar los que tienen precio nulo.

Recuerde la diferencia de los valores que retorna la función "count()" cuando enviamos como argumento un asterisco o el nombre de un campo: en el primer caso cuenta todos los registros incluyendo los que tienen valor nulo, en el segundo, los registros en los cuales el campo especificado es no nulo.

Para conocer el total en dinero de los libros agrupados por editorial:

select editorial, sum(precio)

from libros

group by editorial;

Para saber el máximo y mínimo valor de los libros agrupados por editorial:

select editorial,

max(precio) as mayor,

min(precio) as menor

from libros

group by editorial;

Para calcular el promedio del valor de los libros agrupados por editorial:

select editorial, avg(precio)

from libros

group by editorial;

Es posible limitar la consulta con "where".

Si incluye una cláusula "where", sólo se agrupan los registros que cumplen las condiciones.

Vamos a contar y agrupar por editorial considerando solamente los libros cuyo precio sea menor a 30 pesos:

select editorial, count(\*)

from libros

where precio<30

group by editorial;

Note que las editoriales que no tienen libros que cumplan la condición, no aparecen en la salida. Para que aparezcan todos los valores de editorial, incluso los que devuelven cero o "null" en la columna de agregado, debemos emplear la palabra clave "all" al lado de "group by":

select editorial, count(\*)

from libros

where precio<30

group by all editorial;

Entonces, usamos "group by" para organizar registros en grupos y obtener un resumen de dichos grupos. SQL Server produce una columna de valores por cada grupo, devolviendo filas por cada grupo especificado.

**Ejercicio:**

Trabajamos con la tabla "libros" de una librería.

Eliminamos la tabla, si existe:

if object\_id('libros') is not null

drop table libros;

Creamos la tabla:

create table libros(

codigo int identity,

titulo varchar(40),

autor varchar(30),

editorial varchar(15),

precio decimal(5,2),

cantidad tinyint,

primary key(codigo)

);

Ingresamos algunos registros:

insert into libros

values('El aleph','Borges','Planeta',15,null);

insert into libros

values('Martin Fierro','Jose Hernandez','Emece',22.20,200);

insert into libros

values('Antologia poetica','J.L. Borges','Planeta',null,150);

insert into libros

values('Aprenda PHP','Mario Molina','Emece',18.20,null);

insert into libros

values('Cervantes y el quijote','Bioy Casares- J.L. Borges','Paidos',null,100);

insert into libros

values('Manual de PHP', 'J.C. Paez', 'Siglo XXI',31.80,120);

insert into libros

values('Harry Potter y la piedra filosofal','J.K. Rowling',default,45.00,90);

insert into libros

values('Harry Potter y la camara secreta','J.K. Rowling','Emece',null,100);

insert into libros

values('Alicia en el pais de las maravillas','Lewis Carroll','Paidos',22.50,200);

insert into libros

values('PHP de la A a la Z',null,null,null,0);

Queremos saber la cantidad de libros de cada editorial, utilizando la cláusula "group by":

select editorial, count(\*)

from libros

group by editorial;

El resultado muestra los nombres de las editoriales y la cantidad de registros para cada valor del campo. Note que los valores nulos se procesan como otro grupo.

Obtenemos la cantidad libros con precio no nulo de cada editorial:

select editorial, count(precio)

from libros

group by editorial;

La salida muestra los nombres de las editoriales y la cantidad de registros de cada una, sin contar los que tienen precio nulo.

Para conocer el total en dinero de los libros agrupados por editorial, tipeamos:

select editorial, sum(precio)

from libros

group by editorial;

Obtenemos el máximo y mínimo valor de los libros agrupados por editorial, en una sola sentencia:

select editorial,

max(precio) as mayor,

min(precio) as menor

from libros

group by editorial;

Calculamos el promedio del valor de los libros agrupados por editorial:

select editorial, avg(precio)

from libros

group by editorial;

Es posible limitar la consulta con "where". Vamos a contar y agrupar por editorial considerando solamente los libros cuyo precio es menor a 30 pesos:

select editorial, count(\*)

from libros

where precio<30

group by editorial;

Note que las editoriales que no tienen libros que cumplan la condición, no aparecen en la salida. Para que aparezcan todos los valores de editorial, incluso los que devuelven cero o "null" en la columna de agregado, debemos emplear la palabra clave "all" al lado de "group by":

select editorial, count(\*)

from libros

where precio<30

group by all editorial;

**Código:**

if object\_id('libros') is not null

drop table libros;

create table libros(

codigo int identity,

titulo varchar(40),

autor varchar(30),

editorial varchar(15),

precio decimal(5,2),

cantidad tinyint,

primary key(codigo)

);

insert into libros

values('El aleph','Borges','Planeta',15,null);

insert into libros

values('Martin Fierro','Jose Hernandez','Emece',22.20,200);

insert into libros

values('Antologia poetica','J.L. Borges','Planeta',null,150);

insert into libros

values('Aprenda PHP','Mario Molina','Emece',18.20,null);

insert into libros

values('Cervantes y el quijote','Bioy Casares- J.L. Borges','Paidos',null,100);

insert into libros

values('Manual de PHP', 'J.C. Paez', 'Siglo XXI',31.80,120);

insert into libros

values('Harry Potter y la piedra filosofal','J.K. Rowling',default,45.00,90);

insert into libros

values('Harry Potter y la camara secreta','J.K. Rowling','Emece',null,100);

insert into libros

values('Alicia en el pais de las maravillas','Lewis Carroll','Paidos',22.50,200);

insert into libros

values('PHP de la A a la Z',null,null,null,0);

select editorial, count(\*)

from libros

group by editorial;

select editorial, count(precio)

from libros

group by editorial;

select editorial, sum(precio)

from libros

group by editorial;

select editorial,

max(precio) as mayor,

min(precio) as menor

from libros

group by editorial;

select editorial, avg(precio)

from libros

group by editorial;

select editorial, count(\*)

from libros

where precio<30

group by editorial;

select editorial, count(\*)

from libros

where precio<30

group by all editorial;

**Problema:**

Un comercio que tiene un stand en una feria registra en una tabla llamada "visitantes" algunos datos

de las personas que visitan o compran en su stand para luego enviarle publicidad de sus productos.

1- Elimine la tabla "visitantes", si existe:

if object\_id('visitantes') is not null

drop table visitantes;

2- Cree la tabla con la siguiente estructura:

create table visitantes(

nombre varchar(30),

edad tinyint,

sexo char(1) default 'f',

domicilio varchar(30),

ciudad varchar(20) default 'Cordoba',

telefono varchar(11),

mail varchar(30) default 'no tiene',

montocompra decimal (6,2)

);

3- Ingrese algunos registros:

insert into visitantes

values ('Susana Molina',35,default,'Colon 123',default,null,null,59.80);

insert into visitantes

values ('Marcos Torres',29,'m',default,'Carlos Paz',default,'marcostorres@hotmail.com',150.50);

insert into visitantes

values ('Mariana Juarez',45,default,default,'Carlos Paz',null,default,23.90);

insert into visitantes (nombre, edad,sexo,telefono, mail)

values ('Fabian Perez',36,'m','4556677','fabianperez@xaxamail.com');

insert into visitantes (nombre, ciudad, montocompra)

values ('Alejandra Gonzalez','La Falda',280.50);

insert into visitantes (nombre, edad,sexo, ciudad, mail,montocompra)

values ('Gaston Perez',29,'m','Carlos Paz','gastonperez1@gmail.com',95.40);

insert into visitantes

values ('Liliana Torres',40,default,'Sarmiento 876',default,default,default,85);

insert into visitantes

values ('Gabriela Duarte',21,null,null,'Rio Tercero',default,'gabrielaltorres@hotmail.com',321.50);

4- Queremos saber la cantidad de visitantes de cada ciudad utilizando la cláusula "group by" (4 filas devueltas)

5- Queremos la cantidad visitantes con teléfono no nulo, de cada ciudad (4 filas devueltas)

6- Necesitamos el total del monto de las compras agrupadas por sexo (3 filas)

7- Se necesita saber el máximo y mínimo valor de compra agrupados por sexo y ciudad (6 filas)

8- Calcule el promedio del valor de compra agrupados por ciudad (4 filas)

9- Cuente y agrupe por ciudad sin tener en cuenta los visitantes que no tienen mail (3 filas):

10- Realice la misma consulta anterior, pero use la palabra clave "all" para mostrar todos los

valores de ciudad, incluyendo las que devuelven cero o "null" en la columna de agregado (4 filas)

**Solución:**

if object\_id('visitantes') is not null

drop table visitantes;

create table visitantes(

nombre varchar(30),

edad tinyint,

sexo char(1) default 'f',

domicilio varchar(30),

ciudad varchar(20) default 'Cordoba',

telefono varchar(11),

mail varchar(30) default 'no tiene',

montocompra decimal (6,2)

);

insert into visitantes

values ('Susana Molina',35,default,'Colon 123',default,null,null,59.80);

insert into visitantes

values ('Marcos Torres',29,'m',default,'Carlos Paz',default,'marcostorres@hotmail.com',150.50);

insert into visitantes

values ('Mariana Juarez',45,default,default,'Carlos Paz',null,default,23.90);

insert into visitantes (nombre, edad,sexo,telefono, mail)

values ('Fabian Perez',36,'m','4556677','fabianperez@xaxamail.com');

insert into visitantes (nombre, ciudad, montocompra)

values ('Alejandra Gonzalez','La Falda',280.50);

insert into visitantes (nombre, edad,sexo, ciudad, mail,montocompra)

values ('Gaston Perez',29,'m','Carlos Paz','gastonperez1@gmail.com',95.40);

insert into visitantes

values ('Liliana Torres',40,default,'Sarmiento 876',default,default,default,85);

insert into visitantes

values ('Gabriela Duarte',21,null,null,'Rio Tercero',default,'gabrielaltorres@hotmail.com',321.50);

select ciudad, count(\*)

from visitantes

group by ciudad;

select ciudad, count(telefono)

from visitantes

group by ciudad;

select sexo, sum(montocompra)

from visitantes

group by sexo;

select sexo,ciudad,

max(montocompra) as mayor,

min(montocompra) as menor

from visitantes

group by sexo,ciudad;

select ciudad,

avg(montocompra) as 'promedio de compras'

from visitantes

group by ciudad;

select ciudad,

count(\*) as 'cantidad con mail'

from visitantes

where mail is not null and

mail<>'no tiene'

group by ciudad;

select ciudad,

count(\*) as 'cantidad con mail'

from visitantes

where mail is not null and

mail<>'no tiene'

group by all ciudad;

# Seleccionar grupos (having)

Así como la cláusula "where" permite seleccionar (o rechazar) registros individuales; la cláusula "having" permite seleccionar (o rechazar) un grupo de registros.

Si queremos saber la cantidad de libros agrupados por editorial usamos la siguiente instrucción ya aprendida:

select editorial, count(\*)

from libros

group by editorial;

Si queremos saber la cantidad de libros agrupados por editorial pero considerando sólo algunos grupos, por ejemplo, los que devuelvan un valor mayor a 2, usamos la siguiente instrucción:

select editorial, count(\*) from libros

group by editorial

having count(\*)>2;

Se utiliza "having", seguido de la condición de búsqueda, para seleccionar ciertas filas retornadas por la cláusula "group by".

Veamos otros ejemplos. Queremos el promedio de los precios de los libros agrupados por editorial, pero solamente de aquellos grupos cuyo promedio supere los 25 pesos:

select editorial, avg(precio) from libros

group by editorial

having avg(precio)>25;

En algunos casos es posible confundir las cláusulas "where" y "having". Queremos contar los registros agrupados por editorial sin tener en cuenta a la editorial "Planeta".

Analicemos las siguientes sentencias:

select editorial, count(\*) from libros

where editorial<>'Planeta'

group by editorial;

select editorial, count(\*) from libros

group by editorial

having editorial<>'Planeta';

Ambas devuelven el mismo resultado, pero son diferentes. La primera, selecciona todos los registros rechazando los de editorial "Planeta" y luego los agrupa para contarlos. La segunda, selecciona todos los registros, los agrupa para contarlos y finalmente rechaza fila con la cuenta correspondiente a la editorial "Planeta".

No debemos confundir la cláusula "where" con la cláusula "having"; la primera establece condiciones para la selección de registros de un "select"; la segunda establece condiciones para la selección de registros de una salida "group by".

Veamos otros ejemplos combinando "where" y "having". Queremos la cantidad de libros, sin considerar los que tienen precio nulo, agrupados por editorial, sin considerar la editorial "Planeta":

select editorial, count(\*) from libros

where precio is not null

group by editorial

having editorial<>'Planeta';

Aquí, selecciona los registros rechazando los que no cumplan con la condición dada en "where", luego los agrupa por "editorial" y finalmente rechaza los grupos que no cumplan con la condición dada en el "having".

Se emplea la cláusula "having" con funciones de agrupamiento, esto no puede hacerlo la cláusula "where". Por ejemplo queremos el promedio de los precios agrupados por editorial, de aquellas editoriales que tienen más de 2 libros:

select editorial, avg(precio) from libros

group by editorial

having count(\*) > 2;

En una cláusula "having" puede haber hasta 128 condiciones. Cuando utilice varias condiciones, tiene que combinarlas con operadores lógicos (and, or, not).

Podemos encontrar el mayor valor de los libros agrupados y ordenados por editorial y seleccionar las filas que tengan un valor menor a 100 y mayor a 30:

select editorial, max(precio) as 'mayor'

from libros

group by editorial

having min(precio)<100 and

min(precio)>30

order by editorial;

Entonces, usamos la claúsula "having" para restringir las filas que devuelve una salida "group by". Va siempre después de la cláusula "group by" y antes de la cláusula "order by" si la hubiere.

**Ejercicio:**

Trabajamos con la tabla "libros" de una librería.

Eliminamos la tabla, si existe:

if object\_id('libros') is not null

drop table libros;

Creamos la tabla:

create table libros(

codigo int identity,

titulo varchar(40),

autor varchar(30),

editorial varchar(15),

precio decimal(5,2),

cantidad tinyint,

primary key(codigo)

);

Ingresamos algunos registros:

insert into libros

values('El aleph','Borges','Planeta',35,null);

insert into libros

values('Martin Fierro','Jose Hernandez','Emece',22.20,200);

insert into libros

values('Martin Fierro','Jose Hernandez','Planeta',40,200);

insert into libros

values('Antologia poetica','J.L. Borges','Planeta',null,150);

insert into libros

values('Aprenda PHP','Mario Molina','Emece',18,null);

insert into libros

values('Manual de PHP', 'J.C. Paez', 'Siglo XXI',56,120);

insert into libros

values('Cervantes y el quijote','Bioy Casares- J.L. Borges','Paidos',null,100);

insert into libros

values('Harry Potter y la piedra filosofal','J.K. Rowling',default,45.00,90);

insert into libros

values('Harry Potter y la camara secreta','J.K. Rowling','Emece',null,100);

insert into libros

values('Alicia en el pais de las maravillas','Lewis Carroll','Paidos',42,80);

insert into libros

values('PHP de la A a la Z',null,null,110,0);

insert into libros

values('Uno','Richard Bach','Planeta',25,null);

Queremos saber la cantidad de libros agrupados por editorial pero considerando sólo algunos grupos, por ejemplo, los que devuelvan un valor mayor a 2, usamos la siguiente instrucción:

select editorial, count(\*) from libros

group by editorial

having count(\*)>2;

Queremos el promedio de los precios de los libros agrupados por editorial, pero solamente de aquellos grupos cuyo promedio supere los 25 pesos:

select editorial, avg(precio) from libros

group by editorial

having avg(precio)>25;

Queremos la cantidad de libros, sin considerar los que tienen precio nulo (where), agrupados por editorial (group by), sin considerar la editorial "Planeta" (having):

select editorial, count(\*) from libros

where precio is not null

group by editorial

having editorial<>'Planeta';

Necesitamos el promedio de los precios agrupados por editorial, de aquellas editoriales que tienen más de 2 libros:

select editorial, avg(precio) from libros

group by editorial

having count(\*) > 2;

Buscamos el mayor valor de los libros agrupados y ordenados por editorial y seleccionamos las filas que tienen un valor menor a 100 y mayor a 30:

select editorial, max(precio) as 'mayor'

from libros

group by editorial

having max(precio)<100 and

max(precio)>30

order by editorial;

**Código:**

if object\_id('libros') is not null

drop table libros;

create table libros(

codigo int identity,

titulo varchar(40),

autor varchar(30),

editorial varchar(15),

precio decimal(5,2),

cantidad tinyint,

primary key(codigo)

);

insert into libros

values('El aleph','Borges','Planeta',35,null);

insert into libros

values('Martin Fierro','Jose Hernandez','Emece',22.20,200);

insert into libros

values('Martin Fierro','Jose Hernandez','Planeta',40,200);

insert into libros

values('Antologia poetica','J.L. Borges','Planeta',null,150);

insert into libros

values('Aprenda PHP','Mario Molina','Emece',18,null);

insert into libros

values('Manual de PHP', 'J.C. Paez', 'Siglo XXI',56,120);

insert into libros

values('Cervantes y el quijote','Bioy Casares- J.L. Borges','Paidos',null,100);

insert into libros

values('Harry Potter y la piedra filosofal','J.K. Rowling',default,45.00,90);

insert into libros

values('Harry Potter y la camara secreta','J.K. Rowling','Emece',null,100);

insert into libros

values('Alicia en el pais de las maravillas','Lewis Carroll','Paidos',42,80);

insert into libros

values('PHP de la A a la Z',null,null,110,0);

insert into libros

values('Uno','Richard Bach','Planeta',25,null);

select editorial, count(\*) from libros

group by editorial

having count(\*)>2;

select editorial, avg(precio) from libros

group by editorial

having avg(precio)>25;

select editorial, count(\*) from libros

where precio is not null

group by editorial

having editorial<>'Planeta';

select editorial, avg(precio) from libros

group by editorial

having count(\*) > 2;

select editorial, max(precio) as 'mayor'

from libros

group by editorial

having max(precio)<100 and

max(precio)>30

order by editorial;

**Problema:**

Una empresa tiene registrados sus clientes en una tabla llamada "clientes".

1- Elimine la tabla "clientes", si existe:

if object\_id('clientes') is not null

drop table clientes;

2- Créela con la siguiente estructura:

create table clientes (

codigo int identity,

nombre varchar(30) not null,

domicilio varchar(30),

ciudad varchar(20),

provincia varchar (20),

telefono varchar(11),

primary key(codigo)

);

3- Ingrese algunos registros:

insert into clientes

values ('Lopez Marcos','Colon 111','Cordoba','Cordoba','null');

insert into clientes

values ('Perez Ana','San Martin 222','Cruz del Eje','Cordoba','4578585');

insert into clientes

values ('Garcia Juan','Rivadavia 333','Villa del Rosario','Cordoba','4578445');

insert into clientes

values ('Perez Luis','Sarmiento 444','Rosario','Santa Fe',null);

insert into clientes

values ('Pereyra Lucas','San Martin 555','Cruz del Eje','Cordoba','4253685');

insert into clientes

values ('Gomez Ines','San Martin 666','Santa Fe','Santa Fe','0345252525');

insert into clientes

values ('Torres Fabiola','Alem 777','Villa del Rosario','Cordoba','4554455');

insert into clientes

values ('Lopez Carlos',null,'Cruz del Eje','Cordoba',null);

insert into clientes

values ('Ramos Betina','San Martin 999','Cordoba','Cordoba','4223366');

insert into clientes

values ('Lopez Lucas','San Martin 1010','Posadas','Misiones','0457858745');

4- Obtenga el total de los registros agrupados por ciudad y provincia (6 filas)

5- Obtenga el total de los registros agrupados por ciudad y provincia sin considerar los que tienen menos de 2 clientes (3 filas)

6- Obtenga el total de los clientes que viven en calle "San Martin" (where), agrupados por provincia (group by), teniendo en cuenta todos los valores (all), de aquellas ciudades que tengan menos de 2 clientes (having) y omitiendo la fila correspondiente a la ciudad de "Cordoba" (having) (4 filas devueltas)

**Solución:**

if object\_id('clientes') is not null

drop table clientes;

create table clientes (

codigo int identity,

nombre varchar(30) not null,

domicilio varchar(30),

ciudad varchar(20),

provincia varchar (20),

telefono varchar(11),

primary key(codigo)

);

insert into clientes

values ('Lopez Marcos','Colon 111','Cordoba','Cordoba','null');

insert into clientes

values ('Perez Ana','San Martin 222','Cruz del Eje','Cordoba','4578585');

insert into clientes

values ('Garcia Juan','Rivadavia 333','Villa del Rosario','Cordoba','4578445');

insert into clientes

values ('Perez Luis','Sarmiento 444','Rosario','Santa Fe',null);

insert into clientes

values ('Pereyra Lucas','San Martin 555','Cruz del Eje','Cordoba','4253685');

insert into clientes

values ('Gomez Ines','San Martin 666','Santa Fe','Santa Fe','0345252525');

insert into clientes

values ('Torres Fabiola','Alem 777','Villa del Rosario','Cordoba','4554455');

insert into clientes

values ('Lopez Carlos',null,'Cruz del Eje','Cordoba',null);

insert into clientes

values ('Ramos Betina','San Martin 999','Cordoba','Cordoba','4223366');

insert into clientes

values ('Lopez Lucas','San Martin 1010','Posadas','Misiones','0457858745');

select ciudad, provincia,

count(\*) as cantidad

from clientes

group by ciudad,provincia;

select ciudad, provincia,

count(\*) as cantidad

from clientes

group by ciudad,provincia

having count(\*)>1;

select ciudad, count(\*)

from clientes

where domicilio like '%San Martin%'

group by all ciudad

having count(\*)<2 and

ciudad <> 'Cordoba';

# Modificador del group by (with rollup)

Podemos combinar "group by" con los operadores "rollup" y "cube" para generar valores de resumen a la salida.

El operador "rollup" resume valores de grupos. representan los valores de resumen de la precedente.

Tenemos la tabla "visitantes" con los siguientes campos: nombre, edad, sexo, domicilio, ciudad, telefono, montocompra.

Si necesitamos la cantidad de visitantes por ciudad empleamos la siguiente sentencia:

select ciudad,count(\*) as cantidad

from visitantes

group by ciudad;

Esta consulta muestra el total de visitantes agrupados por ciudad; pero si queremos además la cantidad total de visitantes, debemos realizar otra consulta:

select count(\*) as total

from visitantes;

Para obtener ambos resultados en una sola consulta podemos usar "with rollup" que nos devolverá ambas salidas en una sola consulta:

select ciudad,count(\*) as cantidad

from visitantes

group by ciudad with rollup;

La consulta anterior retorna los registros agrupados por ciudad y una fila extra en la que la primera columna contiene "null" y la columna con la cantidad muestra la cantidad total.

La cláusula "group by" permite agregar el modificador "with rollup", el cual agrega registros extras al resultado de una consulta, que muestran operaciones de resumen.

Si agrupamos por 2 campos, "ciudad" y "sexo":

select ciudad,sexo,count(\*) as cantidad

from visitantes

group by ciudad,sexo

with rollup;

La salida muestra los totales por ciudad y sexo y produce tantas filas extras como valores existen del primer campo por el que se agrupa ("ciudad" en este caso), mostrando los totales para cada valor, con la columna correspondiente al segundo campo por el que se agrupa ("sexo" en este ejemplo) conteniendo "null", y 1 fila extra mostrando el total de todos los visitantes (con las columnas correspondientes a ambos campos conteniendo "null"). Es decir, por cada agrupación, aparece una fila extra con el/ los campos que no se consideran, seteados a "null".

Con "rollup" se puede agrupar hasta por 10 campos.

Es posible incluir varias funciones de agrupamiento, por ejemplo, queremos la cantidad de visitantes y la suma de sus compras agrupados por ciudad y sexo:

select ciudad,sexo,

count(\*) as cantidad,

sum(montocompra) as total

from visitantes

group by ciudad,sexo

with rollup;

Entonces, "rollup" es un modificador para "group by" que agrega filas extras mostrando resultados de resumen de los subgrupos. Si se agrupa por 2 campos SQL Server genera tantas filas extras como valores existen del primer campo (con el segundo campo seteado a "null") y una fila extra con ambos campos conteniendo "null".

Con "rollup" se puede emplear "where" y "having", pero no es compatible con "all".

**Ejercicio:**

Un comercio que tiene un stand en una feria registra en una tabla llamada "visitantes" algunos datos de las personas que visitan o compran en su stand para luego enviarle publicidad de sus productos.

Eliminamos la tabla si existe:

if object\_id('visitantes') is not null

drop table visitantes;

La creamos con la siguiente estructura:

create table visitantes(

nombre varchar(30),

edad tinyint,

sexo char(1),

domicilio varchar(30),

ciudad varchar(20),

telefono varchar(11),

montocompra decimal(6,2) not null

);

Ingresamos algunos registros:

insert into visitantes

values ('Susana Molina',28,'f',null,'Cordoba',null,45.50);

insert into visitantes

values ('Marcela Mercado',36,'f','Avellaneda 345','Cordoba','4545454',22.40);

insert into visitantes

values ('Alberto Garcia',35,'m','Gral. Paz 123','Alta Gracia','03547123456',25);

insert into visitantes

values ('Teresa Garcia',33,'f',default,'Alta Gracia','03547123456',120);

insert into visitantes

values ('Roberto Perez',45,'m','Urquiza 335','Cordoba','4123456',33.20);

insert into visitantes

values ('Marina Torres',22,'f','Colon 222','Villa Dolores','03544112233',95);

insert into visitantes

values ('Julieta Gomez',24,'f','San Martin 333','Alta Gracia',null,53.50);

insert into visitantes

values ('Roxana Lopez',20,'f','null','Alta Gracia',null,240);

insert into visitantes

values ('Liliana Garcia',50,'f','Paso 999','Cordoba','4588778',48);

insert into visitantes

values ('Juan Torres',43,'m','Sarmiento 876','Cordoba',null,15.30);

Necesitamos la cantidad de visitantes por ciudad y el total de visitantes. Para obtener ambos resultados en una sola consulta empleamos "with rollup":

select ciudad,

count(\*) as cantidad

from visitantes

group by ciudad with rollup;

La consulta anterior retorna los registros agrupados por ciudad y una fila extra (la cuarta) en la que la primera columna contiene "null" y la columna con la cantidad muestra la cantidad total.

Veamos las filas de resumen cuando agrupamos por 2 campos, "ciudad" y "sexo":

select ciudad,sexo,

count(\*) as cantidad

from visitantes

group by ciudad,sexo

with rollup;

La salida muestra los totales por ciudad y sexo (5 filas) y produce 4 filas extras, 3 muestran los totales para cada ciudad (con la columna correspondiente al sexo conteniendo "null") y 1 mostrando el total de todos los visitantes (con las columnas correspondientes a la ciudad y al sexo conteniendo "null").

Podemos incluir varias funciones de agrupamiento; para conocer la cantidad de visitantes y la suma de sus compras agrupados por ciudad y sexo, tipeamos:

select ciudad,sexo,

count(\*) as cantidad,

sum(montocompra) as total

from visitantes

group by ciudad,sexo

with rollup;

Note que por cada agrupación aparece una fila extra con valores de resumen.

**Código:**

if object\_id('visitantes') is not null

drop table visitantes;

create table visitantes(

nombre varchar(30),

edad tinyint,

sexo char(1),

domicilio varchar(30),

ciudad varchar(20),

telefono varchar(11),

montocompra decimal(6,2) not null

);

insert into visitantes

values ('Susana Molina',28,'f',null,'Cordoba',null,45.50);

insert into visitantes

values ('Marcela Mercado',36,'f','Avellaneda 345','Cordoba','4545454',22.40);

insert into visitantes

values ('Alberto Garcia',35,'m','Gral. Paz 123','Alta Gracia','03547123456',25);

insert into visitantes

values ('Teresa Garcia',33,'f',default,'Alta Gracia','03547123456',120);

insert into visitantes

values ('Roberto Perez',45,'m','Urquiza 335','Cordoba','4123456',33.20);

insert into visitantes

values ('Marina Torres',22,'f','Colon 222','Villa Dolores','03544112233',95);

insert into visitantes

values ('Julieta Gomez',24,'f','San Martin 333','Alta Gracia',null,53.50);

insert into visitantes

values ('Roxana Lopez',20,'f','null','Alta Gracia',null,240);

insert into visitantes

values ('Liliana Garcia',50,'f','Paso 999','Cordoba','4588778',48);

insert into visitantes

values ('Juan Torres',43,'m','Sarmiento 876','Cordoba',null,15.30);

select ciudad,

count(\*) as cantidad

from visitantes

group by ciudad with rollup;

select ciudad,sexo,

count(\*) as cantidad

from visitantes

group by ciudad,sexo

with rollup;

select ciudad,sexo,

count(\*) as cantidad,

sum(montocompra) as total

from visitantes

group by ciudad,sexo

with rollup;

**Problema:**

Una empresa tiene registrados sus clientes en una tabla llamada "clientes".

1- Elimine la tabla "clientes", si existe:

if object\_id('clientes') is not null

drop table clientes;

2- Créela con la siguiente estructura:

create table clientes (

codigo int identity,

nombre varchar(30) not null,

domicilio varchar(30),

ciudad varchar(20),

estado varchar (20),

pais varchar(20),

primary key(codigo)

);

3- Ingrese algunos registros:

insert into clientes

values ('Lopez Marcos','Colon 111', 'Cordoba','Cordoba','Argentina');

insert into clientes

values ('Perez Ana','San Martin 222', 'Carlos Paz','Cordoba','Argentina');

insert into clientes

values ('Garcia Juan','Rivadavia 333', 'Carlos Paz','Cordoba','Argentina');

insert into clientes

values ('Perez Luis','Sarmiento 444', 'Rosario','Santa Fe','Argentina');

insert into clientes

values ('Gomez Ines','San Martin 987', 'Santa Fe','Santa Fe','Argentina');

insert into clientes

values ('Gomez Ines','San Martin 666', 'Santa Fe','Santa Fe','Argentina');

insert into clientes

values ('Lopez Carlos','Irigoyen 888', 'Cordoba','Cordoba','Argentina');

insert into clientes

values ('Ramos Betina','San Martin 999', 'Cordoba','Cordoba','Argentina');

insert into clientes

values ('Fernando Salas','Mariano Osorio 1234', 'Santiago','Region metropolitana','Chile');

insert into clientes

values ('German Rojas','Allende 345', 'Valparaiso','Region V','Chile');

insert into clientes

values ('Ricardo Jara','Pablo Neruda 146', 'Santiago','Region metropolitana','Chile');

insert into clientes

values ('Joaquin Robles','Diego Rivera 147', 'Guadalajara','Jalisco','Mexico');

4- Necesitamos la cantidad de clientes por país y la cantidad total de clientes en una sola consulta (4 filas)

Note que la consulta retorna los registros agrupados por pais y una fila extra en la que la columna "pais" contiene "null" y la columna con la cantidad muestra la cantidad total.

5- Necesitamos la cantidad de clientes agrupados por pais y estado, incluyendo resultados paciales (9 filas)

Note que la salida muestra los totales por pais y estado y produce 4 filas extras: 3 muestran los totales para cada pais, con la columna "estado" conteniendo "null" y 1 muestra el total de todos los clientes, con las columnas "pais" y "estado" conteniendo "null".

6- Necesitamos la cantidad de clientes agrupados por pais, estado y ciudad, empleando "rollup" (16 filas)

El resultado muestra los totales por pais, estado y ciudad y genera 9 filas extras: 5 muestran los totales para cada estado, con la columna correspondiente a "ciudad" conteniendo "null", 3 muestran los totales para cada pais, con las columnas "ciudad" y "estado" conteniendo "null" y 1 muestra el total de todos los clientes, con las columnas "pais", "estado" y "ciudad" conteniendo "null".

**Solución:**

if object\_id('clientes') is not null

drop table clientes;

create table clientes (

codigo int identity,

nombre varchar(30) not null,

domicilio varchar(30),

ciudad varchar(20),

estado varchar (20),

pais varchar(20),

primary key(codigo)

);

insert into clientes

values ('Lopez Marcos','Colon 111', 'Cordoba','Cordoba','Argentina');

insert into clientes

values ('Perez Ana','San Martin 222', 'Carlos Paz','Cordoba','Argentina');

insert into clientes

values ('Garcia Juan','Rivadavia 333', 'Carlos Paz','Cordoba','Argentina');

insert into clientes

values ('Perez Luis','Sarmiento 444', 'Rosario','Santa Fe','Argentina');

insert into clientes

values ('Gomez Ines','San Martin 987', 'Santa Fe','Santa Fe','Argentina');

insert into clientes

values ('Gomez Ines','San Martin 666', 'Santa Fe','Santa Fe','Argentina');

insert into clientes

values ('Lopez Carlos','Irigoyen 888', 'Cordoba','Cordoba','Argentina');

insert into clientes

values ('Ramos Betina','San Martin 999', 'Cordoba','Cordoba','Argentina');

insert into clientes

values ('Fernando Salas','Mariano Osorio 1234', 'Santiago','Region metropolitana','Chile');

insert into clientes

values ('German Rojas','Allende 345', 'Valparaiso','Region V','Chile');

insert into clientes

values ('Ricardo Jara','Pablo Neruda 146', 'Santiago','Region metropolitana','Chile');

insert into clientes

values ('Joaquin Robles','Diego Rivera 147', 'Guadalajara','Jalisco','Mexico');

select pais,count(\*) as cantidad

from clientes

group by pais with rollup;

select pais,estado,

count(\*) as cantidad

from clientes

group by pais,estado with rollup;

select pais,estado,ciudad,

count(\*) as cantidad

from clientes

group by pais,estado,ciudad

with rollup;

# Modificador del group by (with cube)

Hemos aprendido el modificador "rollup", que agrega filas extras mostrando resultados de resumen por cada grupo y subgrupo.

Por ejemplo, tenemos una tabla llamada "empleados" que contiene, entre otros, los campos "sexo", "estadocivil" y "seccion".

Si se agrupa por esos tres campos (en ese orden) y se emplea "rollup":

select sexo,estadocivil,seccion,

count(\*) from empleados

group by sexo,estadocivil,seccion

with rollup;

SQL Server genera varias filas extras con información de resumen para los siguientes subgrupos:

- sexo y estadocivil (seccion seteado a "null"),

- sexo (estadocivil y seccion seteados a "null") y

- total (todos los campos seteados a "null").

Si se emplea "cube":

select sexo,estadocivil,seccion,

count(\*) from empleados

group by sexo,estadocivil,seccion

with cube;

retorna más filas extras además de las anteriores:

- sexo y seccion (estadocivil seteado a "null"),

- estadocivil y seccion (sexo seteado a "null"),

- seccion (sexo y estadocivil seteados a "null") y

- estadocivil (sexo y seccion seteados a "null"),

Es decir, "cube" genera filas de resumen de subgrupos para todas las combinaciones posibles de los valores de los campos por los que agrupamos.

Se pueden colocar hasta 10 campos en el "group by".

Con "cube" se puede emplear "where" y "having", pero no es compatible con "all"

**Ejercicio:**

Una empresa tiene registrados sus empleados en una tabla llamada "empleados".

Elimine la tabla si existe:

if object\_id('empleados') is not null

drop table empleados;

Créela con la siguiente estructura:

create table empleados (

documento varchar(8) not null,

nombre varchar(30),

sexo char(1),

estadocivil char(1),--c=casado, s=soltero,v=viudo

seccion varchar(20),

primary key (documento)

);

Ingrese algunos registros:

insert into empleados

values ('22222222','Alberto Lopez','m','c','Sistemas');

insert into empleados

values ('23333333','Beatriz Garcia','f','c','Administracion');

insert into empleados

values ('24444444','Carlos Fuentes','m','s','Administracion');

insert into empleados

values ('25555555','Daniel Garcia','m','s','Sistemas');

insert into empleados

values ('26666666','Ester Juarez','f','c','Sistemas');

insert into empleados

values ('27777777','Fabian Torres','m','s','Sistemas');

insert into empleados

values ('28888888','Gabriela Lopez','f','c','Sistemas');

insert into empleados

values ('29999999','Hector Garcia','m','c','Administracion');

insert into empleados

values ('30000000','Ines Torres','f','c','Administracion');

insert into empleados

values ('11111111','Juan Garcia','m','v','Administracion');

insert into empleados

values ('12222222','Luisa Perez','f','v','Administracion');

insert into empleados

values ('31111111','Marcela Garcia','f','s','Administracion');

insert into empleados

values ('32222222','Nestor Fuentes','m','c','Sistemas');

insert into empleados

values ('33333333','Oscar Garcia','m','s','Sistemas');

insert into empleados

values ('34444444','Patricia Juarez','f','c','Administracion');

insert into empleados

values ('35555555','Roberto Torres','m','c','Sistemas');

insert into empleados

values ('36666666','Susana Torres','f','c','Administracion');

Agrupamos por "sexo", "estadocivil" y "seccion" empleando "rollup":

select sexo,estadocivil,seccion,

count(\*) from empleados

group by sexo,estadocivil,seccion

with rollup;

SQL Server genera varias filas extras con información de resumen para los siguientes subgrupos:

- sexo y estadocivil (seccion seteado a "null"),

- sexo (estadocivil e hijos seteados a "null") y

- total (todos los campos seteados a "null").

Si empleamos "cube":

select sexo,estadocivil,seccion,

count(\*) from empleados

group by sexo,estadocivil,seccion

with cube;

retorna más filas extras además de las anteriores:

- sexo y seccion (estadocivil seteado a "null"),

- estadocivil y seccion (sexo seteado a "null"),

- seccion (sexo y estadocivil seteados a "null") y

- estadocivil (sexo e hijos seteados a "null").

**Código:**

if object\_id('empleados') is not null

drop table empleados;

create table empleados (

documento varchar(8) not null,

nombre varchar(30),

sexo char(1),

estadocivil char(1),--c=casado, s=soltero,v=viudo

seccion varchar(20),

primary key (documento)

);

insert into empleados

values ('22222222','Alberto Lopez','m','c','Sistemas');

insert into empleados

values ('23333333','Beatriz Garcia','f','c','Administracion');

insert into empleados

values ('24444444','Carlos Fuentes','m','s','Administracion');

insert into empleados

values ('25555555','Daniel Garcia','m','s','Sistemas');

insert into empleados

values ('26666666','Ester Juarez','f','c','Sistemas');

insert into empleados

values ('27777777','Fabian Torres','m','s','Sistemas');

insert into empleados

values ('28888888','Gabriela Lopez','f','c','Sistemas');

insert into empleados

values ('29999999','Hector Garcia','m','c','Administracion');

insert into empleados

values ('30000000','Ines Torres','f','c','Administracion');

insert into empleados

values ('11111111','Juan Garcia','m','v','Administracion');

insert into empleados

values ('12222222','Luisa Perez','f','v','Administracion');

insert into empleados

values ('31111111','Marcela Garcia','f','s','Administracion');

insert into empleados

values ('32222222','Nestor Fuentes','m','c','Sistemas');

insert into empleados

values ('33333333','Oscar Garcia','m','s','Sistemas');

insert into empleados

values ('34444444','Patricia Juarez','f','c','Administracion');

insert into empleados

values ('35555555','Roberto Torres','m','c','Sistemas');

insert into empleados

values ('36666666','Susana Torres','f','c','Administracion');

select sexo,estadocivil,seccion,

count(\*) from empleados

group by sexo,estadocivil,seccion

with rollup;

select sexo,estadocivil,seccion,

count(\*) from empleados

group by sexo,estadocivil,seccion

with cube;

**Problema:**

Un comercio guarda la información de sus ventas en una tabla llamada "ventas" en la que se registra diariamente, el número de venta, el monto total de la compra, el tipo de pago (c=contado, t=tarjeta) y el nombre del vendedor.

1- Elimine la tabla si existe:

if object\_id('ventas') is not null

drop table ventas;

2- Cree la tabla:

create table ventas(

numero int identity,

montocompra decimal(6,2),

tipopago char(1),--c=contado, t=tarjeta

vendedor varchar(30),

primary key (numero)

);

3- Ingrese algunos registros:

insert into ventas

values(100.50,'c','Marisa Perez');

insert into ventas

values(200,'c','Marisa Perez');

insert into ventas

values(50,'t','Juan Lopez');

insert into ventas

values(220,'c','Juan Lopez');

insert into ventas

values(150,'t','Marisa Perez');

insert into ventas

values(550.80,'c','Marisa Perez');

insert into ventas

values(300,'t','Juan Lopez');

insert into ventas

values(25,'c','Marisa Perez');

4- Agrupe por "tipopago" y "vendedor" y cuente la cantidad empleando "rollup".

Las agrupaciones de resumen son las siguientes:

- vendedor (tipopago seteado a "null"), 2 filas y

- total (todos los campos seteados a "null"), 1 fila

5- Agrupe por "tipopago" y "vendedor" y cuente la cantidad empleando "cube".

Las agrupaciones de resumen son las siguientes:

- vendedor (tipopago seteado a "null"), 2 filas,

- total (todos los campos seteados a "null"), 1 fila y

- tipopago (vendedor seteado a "null"), 2 filas.

**Solución:**

if object\_id('ventas') is not null

drop table ventas;

create table ventas(

numero int identity,

montocompra decimal(6,2),

tipopago char(1),--c=contado, t=tarjeta

vendedor varchar(30),

primary key (numero)

);

insert into ventas

values(100.50,'c','Marisa Perez');

insert into ventas

values(200,'c','Marisa Perez');

insert into ventas

values(50,'t','Juan Lopez');

insert into ventas

values(220,'c','Juan Lopez');

insert into ventas

values(150,'t','Marisa Perez');

insert into ventas

values(550.80,'c','Marisa Perez');

insert into ventas

values(300,'t','Juan Lopez');

insert into ventas

values(25,'c','Marisa Perez');

select vendedor,tipopago,

count(\*) as cantidad

from ventas

group by vendedor,tipopago

with rollup;

select vendedor,tipopago,

count(\*) as cantidad

from ventas

group by vendedor,tipopago

with cube;

# Función grouping

La función "grouping" se emplea con los operadores "rollup" y "cube" para distinguir los valores de detalle y de resumen en el resultado. Es decir, permite diferenciar si los valores "null" que aparecen en el resultado son valores nulos de las tablas o si son una fila generada por los operadores "rollup" o "cube".

Con esta función aparece una nueva columna en la salida, una por cada "grouping"; retorna el valor 1 para indicar que la fila representa los valores de resumen de "rollup" o "cube" y el valor 0 para representar los valores de campo.

Sólo se puede emplear la función "grouping" en los campos que aparecen en la cláusula "group by".

Si tenemos una tabla "visitantes" con los siguientes registros almacenados:

Nombre sexo ciudad

-----------------------------------------------------

Susana Molina f Cordoba

Marcela Mercado f Cordoba

Roberto Perez f null

Alberto Garcia m Cordoba

Teresa Garcia f Alta Gracia

y contamos la cantidad agrupando por ciudad (note que hay un valor nulo en dicho campo) empleando "rollup":

select ciudad,

count(\*) as cantidad

from visitantes

group by ciudad

with rollup;

aparece la siguiente salida:

ciudad cantidad

-------------------------

NULL 1

Alta Gracia 1

Cordoba 3

NULL 5

La última fila es la de resumen generada por "rollup", pero no es posible distinguirla de la primera fila, en la cual "null" es un valor del campo. Para diferenciarla empleamos "grouping":

select ciudad,

count(\*) as cantidad,

grouping(ciudad) as resumen

from visitantes

group by ciudad

with rollup;

aparece la siguiente salida:

ciudad cantidad resumen

---------------------------------------------------------

NULL 1 0

Alta Gracia 1 0

Cordoba 3 0

NULL 5 1

La última fila contiene en la columna generada por "grouping" el valor 1, indicando que es la fila de resumen generada por "rollup"; la primera fila, contiene en dicha columna el valor 0, que indica que el valor "null" es un valor del campo "ciudad".

Entonces, si emplea los operadores "rollup" y "cube" y los campos por los cuales agrupa admiten valores nulos, utilice la función "grouping" para distinguir los valores de detalle y de resumen en el resultado.

**Ejercicio:**

Un comercio que tiene un stand en una feria registra en una tabla llamada "visitantes" algunos datos de las personas que visitan o compran en su stand para luego enviarle publicidad de sus productos.

Eliminamos la tabla si existe:

if object\_id('visitantes') is not null

drop table visitantes;

La creamos con la siguiente estructura:

create table visitantes(

nombre varchar(30),

sexo char(1),

ciudad varchar(20)

);

Ingresamos algunos registros:

insert into visitantes values('Susana Molina', 'f', 'Cordoba');

insert into visitantes values('Marcela Mercado', 'f','Cordoba');

insert into visitantes values('Roberto Perez','f',null);

insert into visitantes values('Alberto Garcia','m','Cordoba');

insert into visitantes values('Teresa Garcia','f','Alta Gracia');

Contamos la cantidad de visitantes agrupando por ciudad y empleando "rollup":

select ciudad,

count(\*) as cantidad

from visitantes

group by ciudad

with rollup;

Note que la última fila es la de resumen generada por "rollup", pero no es sencillo distinguirla de la primera fila, en la cual "null" es un valor del campo. Para diferenciarla empleamos "grouping":

select ciudad,

count(\*) as cantidad,

grouping(ciudad) as resumen

from visitantes

group by ciudad

with rollup;

Note que la última fila contiene en la columna generada por "grouping" el valor 1, indicando que es la fila de resumen generada por "rollup"; la primera fila, contiene en dicha columna el valor 0, lo cual indica que el valor "null" es un valor del campo "ciudad".

**Código:**

if object\_id('visitantes') is not null

drop table visitantes;

create table visitantes(

nombre varchar(30),

sexo char(1),

ciudad varchar(20)

);

insert into visitantes values('Susana Molina', 'f', 'Cordoba');

insert into visitantes values('Marcela Mercado', 'f','Cordoba');

insert into visitantes values('Roberto Perez','f',null);

insert into visitantes values('Alberto Garcia','m','Cordoba');

insert into visitantes values('Teresa Garcia','f','Alta Gracia');

select ciudad,

count(\*) as cantidad

from visitantes

group by ciudad

with rollup;

select ciudad,

count(\*) as cantidad,

grouping(ciudad) as resumen

from visitantes

group by ciudad

with rollup;

**Problema:**

Una empresa tiene registrados sus empleados en una tabla llamada "empleados".

1- Elimine la tabla si existe:

if object\_id('empleados') is not null

drop table empleados;

2- Créela con la siguiente estructura:

create table empleados (

documento varchar(8) not null,

nombre varchar(30),

sexo char(1),

estadocivil char(1),--c=casado, s=soltero,v=viudo

seccion varchar(20),

primary key (documento)

);

3- Ingrese algunos registros:

insert into empleados

values ('22222222','Alberto Lopez','m','c','Sistemas');

insert into empleados

values ('23333333','Beatriz Garcia','f','c','Administracion');

insert into empleados

values ('24444444','Carlos Fuentes','m','s','Administracion');

insert into empleados

values ('25555555','Daniel Garcia','m','s','Sistemas');

insert into empleados

values ('26666666','Ester Juarez',null,'c','Sistemas');

insert into empleados

values ('27777777','Fabian Torres',null,'s','Sistemas');

insert into empleados

values ('28888888','Gabriela Lopez','f',null,'Sistemas');

insert into empleados

values ('29999999','Hector Garcia','m',null,'Administracion');

4- Cuente la cantidad de empleados agrupados por sexo y estado civil, empleando "rollup".

Es dificil distinguir los valores de detalle y resumen.

5- Realice la misma consulta anterior pero emplee la función "grouping" para los dos campos por los que se agrupa para distinguir los valores de resumen y de detalle.

Note que las columnas de resumen contienen 1 y las de detalle 0.

6- Realice la misma consulta anterior pero con "cube" en lugar de "rollup", distinguiendo los valores de resumen y de detalle.

Note que las columnas de resumen contienen 1 y las de detalle 0.

**Solución:**

if object\_id('empleados') is not null

drop table empleados;

create table empleados (

documento varchar(8) not null,

nombre varchar(30),

sexo char(1),

estadocivil char(1),--c=casado, s=soltero,v=viudo

seccion varchar(20),

primary key (documento)

);

insert into empleados

values ('22222222','Alberto Lopez','m','c','Sistemas');

insert into empleados

values ('23333333','Beatriz Garcia','f','c','Administracion');

insert into empleados

values ('24444444','Carlos Fuentes','m','s','Administracion');

insert into empleados

values ('25555555','Daniel Garcia','m','s','Sistemas');

insert into empleados

values ('26666666','Ester Juarez',null,'c','Sistemas');

insert into empleados

values ('27777777','Fabian Torres',null,'s','Sistemas');

insert into empleados

values ('28888888','Gabriela Lopez','f',null,'Sistemas');

insert into empleados

values ('29999999','Hector Garcia','m',null,'Administracion');

select sexo,estadocivil,

count(\*) as cantidad

from empleados

group by sexo,estadocivil

with rollup;

select sexo,estadocivil,

count(\*) as cantidad,

grouping(sexo) as 'resumen sexo',

grouping(estadocivil) as 'resumen estado civil'

from empleados

group by sexo,estadocivil

with rollup;

select sexo,estadocivil,

count(\*) as cantidad,

grouping(sexo) as 'resumen sexo',

grouping(estadocivil) as 'resumen estado civil'

from empleados

group by sexo,estadocivil

with cube;

# Cláusulas compute y compute by

Las cláusulas "compute" y "compute by" generan totales que aparecen en columnas extras al final del resultado.

Produce filas de detalle y un valor único para una columna.

Se usa con las funciones de agrupamiento: avg(), count(), max(), min(), sum().

La sintaxis básica y general es la siguiente:

select CAMPOS

from TABLA

compute FUNCION(CAMPO);

El campo que se coloque en la cláusula "compute" debe estar incluida en la lista de campos del "select".

Para ver todos los datos de los visitantes y el promedio del monto de compra de nuestra tabla "visitantes":

select \*from visitantes

compute avg(montocompra);

Produce la misma salida que las siguientes 2 sentencias:

select \*from visitantes;

select avg(montocompra) from visitantes;

En una misma instrucción se pueden colocar varias cláusulas "compute":

select edad,ciudad,montocompra

from visitantes

compute avg(edad),sum(montocompra);

"Compute by" genera cortes de control y subtotales. Se generan filas de detalle y varios valores de resumen cuando cambian los valores del campo.

Con "compute by" se DEBE usar también la cláusula "order by" y los campos que se incluyan luego de "by" deben estar en el "order by". Listando varios campos luego del "by" corta un grupo en subgrupos y aplica la función de agregado en cada nivel de agrupamiento:

select nombre,ciudad,provincia

from visitantes

order by provincia

compute count(provincia)

by provincia;

select nombre,ciudad,provincia

from visitantes

order by provincia,ciudad

compute count(provincia)

by provincia,ciudad;

Los campos que aparecen luego de la cláusula "compute by" DEBEN ser idénticos a un subconjunto de los campos que aparecen después de "order by" y estar en el mismo orden. Si la cláusula "order by" tiene los siguientes campos:

... order by a,b,c...

la cláusula "compute by" puede incluir los siguientes subconjuntos de campos:

... compute ...

by a...

o

... compute ...

by a,b...

o

... compute ...

by a,b,c...

En una misma instrucción se pueden colocar varias cláusulas "compute" combinadas con varias cláusulas "compute by":

select \*from visitantes

order by provincia,ciudad

compute avg(edad), sum(montocompra)

compute avg(montocompra),count(provincia)

by provincia,ciudad;

El resultado de la consulta anterior muestra el promedio de la compra y la cantidad al final de cada subgrupo de provincia y ciudad (compute by) y el promedio de las edades y el total del monto de compras de todos (compute).

Los tipos de datos ntext, text e image no se pueden incluir en una cláusula "compute" o "compute by".

**Ejercicio:**

Un comercio que tiene un stand en una feria registra en una tabla llamada "visitantes" algunos datos de las personas que visitan o compran en su stand para luego enviarle publicidad de sus productos.

Eliminamos la tabla si existe:

if object\_id('visitantes') is not null

drop table visitantes;

La creamos con la siguiente estructura:

create table visitantes(

nombre varchar(30),

edad tinyint,

ciudad varchar(20),

provincia varchar(20),

mail varchar(30),

montocompra decimal(6,2)

);

Ingresamos algunos registros:

insert into visitantes

values ('Susana Molina',28,'Cordoba','Cordoba','susanamolina@gmail.com',50);

insert into visitantes

values ('Marcela Mercado',36,'Carlos Paz','Cordoba',null,20);

insert into visitantes

values ('Alberto Garcia',35,'La Falda','Cordoba','albertgomgarcia@hotmail.com',30);

insert into visitantes

values ('Teresa Garcia',33,'Alta Gracia','Cordoba',null,120);

insert into visitantes

values ('Roberto Perez',45,'Alta Gracia','Cordoba','robertomarioperez@hotmail.com',30);

insert into visitantes

values ('Marina Torres',22,'Rosario','Santa Fe','marinatorres@xaxamail.com',90);

insert into visitantes

values ('Julieta Gomez',24,'Rosario','Santa Fe',null,50);

insert into visitantes

values ('Roxana Lopez',20,'Alta Gracia','Cordoba',null,200);

insert into visitantes

values ('Liliana Garcia',50,'Capital Federal','Buenos Aires','lilianagarcia@hotmail.com',40);

insert into visitantes

values ('Juan Torres',43,'Carlos Paz','Cordoba',null,10);

Veamos todos los datos de los visitantes y el promedio del monto de compra:

select \*from visitantes

compute avg(montocompra);

Empleamos dos cláusulas "compute" en una misma instrucción para averiguar el promedio de las edades y el total de los montos de las compras:

select edad,ciudad,montocompra

from visitantes

compute avg(edad),sum(montocompra);

Veamos la cantidad de visitantes por provincia y ciudad empleando "compute by". Recuerde que DEBE usarse "order by" y los campos que se incluyan luego de "by" deben estar en el "order by":

select nombre,ciudad,provincia

from visitantes

order by provincia,ciudad

compute count(provincia)

by provincia,ciudad;

Combinamos dos cláusulas "compute" con dos cláusulas "compute by" para averiguar el promedio de la compra y la cantidad por provincia y ciudad y el promedio de las edades y el total del monto de compras de todos los visitantes:

select \*from visitantes

order by provincia,ciudad

compute avg(edad), sum(montocompra)

compute avg(montocompra),count(provincia)

by provincia,ciudad;

**Código:**

if object\_id('visitantes') is not null

drop table visitantes;

create table visitantes(

nombre varchar(30),

edad tinyint,

ciudad varchar(20),

provincia varchar(20),

mail varchar(30),

montocompra decimal(6,2)

);

insert into visitantes

values ('Susana Molina',28,'Cordoba','Cordoba','susanamolina@gmail.com',50);

insert into visitantes

values ('Marcela Mercado',36,'Carlos Paz','Cordoba',null,20);

insert into visitantes

values ('Alberto Garcia',35,'La Falda','Cordoba','albertgomgarcia@hotmail.com',30);

insert into visitantes

values ('Teresa Garcia',33,'Alta Gracia','Cordoba',null,120);

insert into visitantes

values ('Roberto Perez',45,'Alta Gracia','Cordoba','robertomarioperez@hotmail.com',30);

insert into visitantes

values ('Marina Torres',22,'Rosario','Santa Fe','marinatorres@xaxamail.com',90);

insert into visitantes

values ('Julieta Gomez',24,'Rosario','Santa Fe',null,50);

insert into visitantes

values ('Roxana Lopez',20,'Alta Gracia','Cordoba',null,200);

insert into visitantes

values ('Liliana Garcia',50,'Capital Federal','Buenos Aires','lilianagarcia@hotmail.com',40);

insert into visitantes

values ('Juan Torres',43,'Carlos Paz','Cordoba',null,10);

select \*from visitantes

compute avg(montocompra);

select edad,ciudad,montocompra

from visitantes

compute avg(edad),sum(montocompra);

select nombre,ciudad,provincia

from visitantes

order by provincia,ciudad

compute count(provincia)

by provincia,ciudad;

select \*from visitantes

order by provincia,ciudad

compute avg(edad), sum(montocompra)

compute avg(montocompra),count(provincia)

by provincia,ciudad;

**Problema:**

La provincia almacena en una tabla llamada "inmuebles" los siguientes datos de los inmuebles y sus propietarios para cobrar impuestos:

1- Elimine la tabla si existe:

if object\_id('inmuebles') is not null

drop table inmuebles;

2- Créela con la siguiente estructura:

create table inmuebles (

documento varchar(8) not null,

nombre varchar(30),

domicilio varchar(20),

barrio varchar(20),

ciudad varchar(20),

tipo char(1),--b=baldio, e: edificado

superficie decimal (8,2),

monto decimal (8,2)

);

3- Ingrese algunos registros:

insert into inmuebles

values ('11111111','Alberto Acosta','Avellaneda 800','Centro','Cordoba','e',100,1200);

insert into inmuebles

values ('11111111','Alberto Acosta','Sarmiento 245','Gral. Paz','Cordoba','e',200,2500);

insert into inmuebles

values ('22222222','Beatriz Barrios','San Martin 202','Centro','Cordoba','e',250,1900);

insert into inmuebles

values ('33333333','Carlos Caseres','Paso 1234','Alberdi','Cordoba','b',200,1000);

insert into inmuebles

values ('33333333','Carlos Caseres','Guemes 876','Alberdi','Cordoba','b',300,1500);

insert into inmuebles

values ('44444444','Diana Dominguez','Calderon 456','Matienzo','Cordoba','b',200,800);

insert into inmuebles

values ('55555555','Estela Fuentes','San Martin 321','Flores','Carlos Paz','e',500,4500);

insert into inmuebles

values ('55555555','Estela Fuentes','Lopez y Planes 853','Alberdi','Cordoba','e',350,2200);

4- Muestre todos los datos y el promedio del monto empleando "compute" (1 resultado parcial)

5- Empleando cláusulas "compute" consulte el promedio de las superficies y el total de los montos (2 columnas extras)

6- Realice la misma consulta anterior pero empleando "compute by" para obtener resultados parciales por documento,barrio y ciudad.

7- Realice la misma consulta anterior pero con resultados parciales por documento y barrio (6 resultados parciales dobles)

8- Realice la misma consulta anterior pero con resultados parciales por documento (4 resultados parciales dobles)

9- Intente realizar la misma consulta anterior pero con resultados parciales por documento y ciudad.

Aparece un mensaje de error indicando que el subgrupo de campos listados luego del "by" no es correcto.

10- Combine cláusulas "compute" con "compute by" para averiguar el total de monto a pagar por propietario y el promedio de monto de todos (4 resultados parciales y 1 general)

**Solución:**

if object\_id('inmuebles') is not null

drop table inmuebles;

create table inmuebles (

documento varchar(8) not null,

nombre varchar(30),

domicilio varchar(20),

barrio varchar(20),

ciudad varchar(20),

tipo char(1),--b=baldio, e: edificado

superficie decimal (8,2),

monto decimal (8,2)

);

insert into inmuebles

values ('11111111','Alberto Acosta','Avellaneda 800','Centro','Cordoba','e',100,1200);

insert into inmuebles

values ('11111111','Alberto Acosta','Sarmiento 245','Gral. Paz','Cordoba','e',200,2500);

insert into inmuebles

values ('22222222','Beatriz Barrios','San Martin 202','Centro','Cordoba','e',250,1900);

insert into inmuebles

values ('33333333','Carlos Caseres','Paso 1234','Alberdi','Cordoba','b',200,1000);

insert into inmuebles

values ('33333333','Carlos Caseres','Guemes 876','Alberdi','Cordoba','b',300,1500);

insert into inmuebles

values ('44444444','Diana Dominguez','Calderon 456','Matienzo','Cordoba','b',200,800);

insert into inmuebles

values ('55555555','Estela Fuentes','San Martin 321','Flores','Carlos Paz','e',500,4500);

insert into inmuebles

values ('55555555','Estela Fuentes','Lopez y Planes 853','Alberdi','Cordoba','e',350,2200);

select \*from inmuebles

compute avg(monto);

select \*from inmuebles

compute avg(superficie),sum(monto);

select \*from inmuebles

order by documento,barrio,ciudad

compute avg(superficie),sum(monto)

by documento,barrio,ciudad;

select \*from inmuebles

order by documento,barrio,ciudad

compute avg(superficie),sum(monto)

by documento,barrio;

select \*from inmuebles

order by documento,barrio,ciudad

compute avg(superficie),sum(monto)

by documento;

select \*from inmuebles

order by documento,barrio,ciudad

compute avg(superficie),sum(monto)

by documento,ciudad;

select \*from inmuebles

order by documento

compute sum(monto)

compute avg(monto)

by documento;

# Registros duplicados (distinct)

Con la cláusula "distinct" se especifica que los registros con ciertos datos duplicados sean obviados en el resultado. Por ejemplo, queremos conocer todos los autores de los cuales tenemos libros, si utilizamos esta sentencia:

select autor from libros;

Aparecen repetidos. Para obtener la lista de autores sin repetición usamos:

select distinct autor from libros;

También podemos tipear:

select autor from libros

group by autor;

Note que en los tres casos anteriores aparece "null" como un valor para "autor"· Si sólo queremos la lista de autores conocidos, es decir, no queremos incluir "null" en la lista, podemos utilizar la sentencia siguiente:

select distinct autor from libros

where autor is not null;

Para contar los distintos autores, sin considerar el valor "null" usamos:

select count(distinct autor)

from libros;

Note que si contamos los autores sin "distinct", no incluirá los valores "null" pero si los repetidos:

select count(autor)

from libros;

Esta sentencia cuenta los registros que tienen autor.

Podemos combinarla con "where". Por ejemplo, queremos conocer los distintos autores de la editorial "Planeta":

select distinct autor from libros

where editorial='Planeta';

También puede utilizarse con "group by" para contar los diferentes autores por editorial:

select editorial, count(distinct autor)

from libros

group by editorial;

La cláusula "distinct" afecta a todos los campos presentados. Para mostrar los títulos y editoriales de los libros sin repetir títulos ni editoriales, usamos:

select distinct titulo,editorial

from libros

order by titulo;

Note que los registros no están duplicados, aparecen títulos iguales pero con editorial diferente, cada registro es diferente.

La palabra clave "distinct" no está permitida con las cláusulas "compute" y "compute by".

Entonces, "distinct" elimina registros duplicados.

**Ejercicio:**

Trabajamos con la tabla "libros" de una librería.

Eliminamos la tabla, si existe:

if object\_id('libros') is not null

drop table libros;

Creamos la tabla:

create table libros(

codigo int identity,

titulo varchar(40),

autor varchar(30),

editorial varchar(15),

primary key(codigo)

);

Ingresamos algunos registros:

insert into libros

values('El aleph','Borges','Planeta');

insert into libros

values('Martin Fierro','Jose Hernandez','Emece');

insert into libros

values('Martin Fierro','Jose Hernandez','Planeta');

insert into libros

values('Antologia poetica','Borges','Planeta');

insert into libros

values('Aprenda PHP','Mario Molina','Emece');

insert into libros

values('Aprenda PHP','Lopez','Emece');

insert into libros

values('Manual de PHP', 'J. Paez', null);

insert into libros

values('Cervantes y el quijote',null,'Paidos');

insert into libros

values('Harry Potter y la piedra filosofal','J.K. Rowling','Emece');

insert into libros

values('Harry Potter y la camara secreta','J.K. Rowling','Emece');

insert into libros

values('Alicia en el pais de las maravillas','Lewis Carroll','Paidos');

insert into libros

values('Alicia en el pais de las maravillas','Lewis Carroll','Planeta');

insert into libros

values('PHP de la A a la Z',null,null);

insert into libros

values('Uno','Richard Bach','Planeta');

Para obtener la lista de autores sin repetición tipeamos:

select distinct autor from libros;

Note que aparece "null" como un valor para "autor"· Para obtener la lista de autores conocidos, es decir, no incluyendo "null" en la lista:

select distinct autor from libros

where autor is not null;

Contamos los distintos autores:

select count(distinct autor)

from libros;

Queremos los nombres de las editoriales sin repetir:

select distinct editorial from libros;

Queremos saber la cantidad de editoriales distintas:

select count(distinct editorial) from libros;

La combinamos con "where" para obtener los distintos autores de la editorial "Planeta":

select distinct autor from libros

where editorial='Planeta';

Contamos los distintos autores que tiene cada editorial empleando "group by":

select editorial,count(distinct autor)

from libros

group by editorial;

Mostramos los títulos y editoriales de los libros sin repetir títulos ni editoriales:

select distinct titulo,editorial

from libros

order by titulo;

Note que los registros no están duplicados, aparecen títulos iguales pero con editorial diferente, cada registro es diferente.

**Código:**

if object\_id('libros') is not null

drop table libros;

create table libros(

codigo int identity,

titulo varchar(40),

autor varchar(30),

editorial varchar(15),

primary key(codigo)

);

insert into libros

values('El aleph','Borges','Planeta');

insert into libros

values('Martin Fierro','Jose Hernandez','Emece');

insert into libros

values('Martin Fierro','Jose Hernandez','Planeta');

insert into libros

values('Antologia poetica','Borges','Planeta');

insert into libros

values('Aprenda PHP','Mario Molina','Emece');

insert into libros

values('Aprenda PHP','Lopez','Emece');

insert into libros

values('Manual de PHP', 'J. Paez', null);

insert into libros

values('Cervantes y el quijote',null,'Paidos');

insert into libros

values('Harry Potter y la piedra filosofal','J.K. Rowling','Emece');

insert into libros

values('Harry Potter y la camara secreta','J.K. Rowling','Emece');

insert into libros

values('Alicia en el pais de las maravillas','Lewis Carroll','Paidos');

insert into libros

values('Alicia en el pais de las maravillas','Lewis Carroll','Planeta');

insert into libros

values('PHP de la A a la Z',null,null);

insert into libros

values('Uno','Richard Bach','Planeta');

select distinct autor from libros;

select distinct autor from libros

where autor is not null;

select count(distinct autor)

from libros;

select distinct editorial from libros;

select count(distinct editorial) from libros;

select distinct autor from libros

where editorial='Planeta';

select editorial,count(distinct autor)

from libros

group by editorial;

select distinct titulo,editorial

from libros

order by titulo;

**Problema:**

Una empresa tiene registrados sus clientes en una tabla llamada "clientes".

1- Elimine la tabla "clientes", si existe:

if object\_id('clientes') is not null

drop table clientes;

2- Créela con la siguiente estructura:

create table clientes (

codigo int identity,

nombre varchar(30) not null,

domicilio varchar(30),

ciudad varchar(20),

provincia varchar (20),

primary key(codigo)

);

3- Ingrese algunos registros:

insert into clientes

values ('Lopez Marcos','Colon 111','Cordoba','Cordoba');

insert into clientes

values ('Perez Ana','San Martin 222','Cruz del Eje','Cordoba');

insert into clientes

values ('Garcia Juan','Rivadavia 333','Villa del Rosario','Cordoba');

insert into clientes

values ('Perez Luis','Sarmiento 444','Rosario','Santa Fe');

insert into clientes

values ('Pereyra Lucas','San Martin 555','Cruz del Eje','Cordoba');

insert into clientes

values ('Gomez Ines','San Martin 666','Santa Fe','Santa Fe');

insert into clientes

values ('Torres Fabiola','Alem 777','Villa del Rosario','Cordoba');

insert into clientes

values ('Lopez Carlos',null,'Cruz del Eje','Cordoba');

insert into clientes

values ('Ramos Betina','San Martin 999','Cordoba','Cordoba');

insert into clientes

values ('Lopez Lucas','San Martin 1010','Posadas','Misiones');

4- Obtenga las provincias sin repetir (3 registros)

5- Cuente las distintas provincias.

6- Se necesitan los nombres de las ciudades sin repetir (6 registros)

7- Obtenga la cantidad de ciudades distintas.

8- Combine con "where" para obtener las distintas ciudades de la provincia de Cordoba (3 registros)

9- Contamos las distintas ciudades de cada provincia empleando "group by" (3 registros)

**Solución:**

if object\_id('clientes') is not null

drop table clientes;

create table clientes (

codigo int identity,

nombre varchar(30) not null,

domicilio varchar(30),

ciudad varchar(20),

provincia varchar (20),

primary key(codigo)

);

insert into clientes

values ('Lopez Marcos','Colon 111','Cordoba','Cordoba');

insert into clientes

values ('Perez Ana','San Martin 222','Cruz del Eje','Cordoba');

insert into clientes

values ('Garcia Juan','Rivadavia 333','Villa del Rosario','Cordoba');

insert into clientes

values ('Perez Luis','Sarmiento 444','Rosario','Santa Fe');

insert into clientes

values ('Pereyra Lucas','San Martin 555','Cruz del Eje','Cordoba');

insert into clientes

values ('Gomez Ines','San Martin 666','Santa Fe','Santa Fe');

insert into clientes

values ('Torres Fabiola','Alem 777','Villa del Rosario','Cordoba');

insert into clientes

values ('Lopez Carlos',null,'Cruz del Eje','Cordoba');

insert into clientes

values ('Ramos Betina','San Martin 999','Cordoba','Cordoba');

insert into clientes

values ('Lopez Lucas','San Martin 1010','Posadas','Misiones');

select distinct provincia from clientes;

select count(distinct provincia) as cantidad

from clientes;

select distinct ciudad from clientes;

select count(distinct ciudad) from clientes;

select distinct ciudad from clientes

where provincia='Cordoba';

select provincia,count(distinct ciudad)

from clientes

group by provincia;

# Cláusula top

La palabra clave "top" se emplea para obtener sólo una cantidad limitada de registros, los primeros n registros de una consulta.

Con la siguiente consulta obtenemos todos los datos de los primeros 2 libros de la tabla:

select top 2 \*from libros;

Es decir, luego del "select" se coloca "top" seguido de un número entero positivo y luego se continúa con la consulta.

Se puede combinar con "order by":

select top 3 titulo,autor

from libros

order by autor;

En la consulta anterior solicitamos los títulos y autores de los 3 primeros libros, ordenados por autor.

Cuando se combina con "order by" es posible emplear también la cláusula "with ties". Esta cláusula permite incluir en la seleccion, todos los registros que tengan el mismo valor del campo por el que se ordena, que el último registro retornado si el último registro retornado (es decir, el número n) tiene un valor repetido en el registro n+1. Es decir, si el valor del campo por el cual se ordena del último registro retornado (el número n) está repetido en los siguientes registros (es decir, el n+1 tiene el mismo valor que n, y el n+2, etc.), lo incluye en la selección.

Veamos un ejemplo:

select top 3 with ties

\*from libros

order by autor;

Esta consulta solicita el retorno de los primeros 3 registros; en caso que el registro número 4 (y los posteriores), tengan el mismo valor en "autor" que el último registro retornado (número 3), también aparecerán en la selección.

Si colocamos un valor para "top" que supera la cantidad de registros de la tabla, SQL Server muestra todos los registros.

**Ejercicio:**

Trabajamos con la tabla "libros" de una librería.

Eliminamos la tabla, si existe:

if object\_id('libros') is not null

drop table libros;

Creamos la tabla:

create table libros(

codigo int identity,

titulo varchar(40),

autor varchar(20),

editorial varchar(20)

);

Ingresamos algunos registros:

insert into libros values ('Uno','Richard Bach','Planeta');

insert into libros values ('El aleph','Borges','Emece');

insert into libros values ('Alicia en el pais...','Carroll','Planeta');

insert into libros values ('Aprenda PHP','Mario Molina','Siglo XXI');

insert into libros values ('Java en 10 minutos','Mario Molina','Siglo XXI');

insert into libros values ('Java desde cero','Mario Molina','Emece');

insert into libros values ('Ilusiones','Richard Bach','Planeta');

Obtenemos todos los datos de los primeros 2 libros de la tabla:

select top 2 \*from libros;

Mostramos los títulos y autores de los 3 primeros libros ordenados por autor:

select top 3 titulo,autor

from libros

order by autor;

Realizamos la misma consulta anterior pero empleamos la cláusula "with ties", con lo cual incluiremos en la selección, todos los registros que tengan el mismo autor que el último registro retornado, aunque pasemos de 3:

select top 3 with ties titulo,autor

from libros

order by autor;

Note que retorna los 5 primeros registros porque incluye los dos siguientes que tienen el mismo valor que el último en el campo "autor" (por el cual se ordena).

**Código:**

if object\_id('libros') is not null

drop table libros;

create table libros(

codigo int identity,

titulo varchar(40),

autor varchar(20),

editorial varchar(20)

);

insert into libros values ('Uno','Richard Bach','Planeta');

insert into libros values ('El aleph','Borges','Emece');

insert into libros values ('Alicia en el pais...','Carroll','Planeta');

insert into libros values ('Aprenda PHP','Mario Molina','Siglo XXI');

insert into libros values ('Java en 10 minutos','Mario Molina','Siglo XXI');

insert into libros values ('Java desde cero','Mario Molina','Emece');

insert into libros values ('Ilusiones','Richard Bach','Planeta');

select top 2 \*from libros;

select top 3 titulo,autor

from libros

order by autor;

select top 3 with ties titulo,autor

from libros

order by autor;

**Problema:**

Una empresa tiene registrados sus empleados en una tabla llamada "empleados".

1- Elimine la tabla si existe:

if object\_id('empleados') is not null

drop table empleados;

2- Créela con la siguiente estructura:

create table empleados (

documento varchar(8) not null,

nombre varchar(30),

estadocivil char(1),--c=casado, s=soltero,v=viudo

seccion varchar(20)

);

3- Ingrese algunos registros:

insert into empleados

values ('22222222','Alberto Lopez','c','Sistemas');

insert into empleados

values ('23333333','Beatriz Garcia','c','Administracion');

insert into empleados

values ('24444444','Carlos Fuentes','s','Administracion');

insert into empleados

values ('25555555','Daniel Garcia','s','Sistemas');

insert into empleados

values ('26666666','Ester Juarez','c','Sistemas');

insert into empleados

values ('27777777','Fabian Torres','s','Sistemas');

insert into empleados

values ('28888888','Gabriela Lopez',null,'Sistemas');

insert into empleados

values ('29999999','Hector Garcia',null,'Administracion');

4- Muestre los 5 primeros registros (5 registros)

5- Muestre nombre y seccion de los 4 primeros registros ordenados por sección (4 registros)

6- Realice la misma consulta anterior pero incluya todos los registros que tengan el mismo valor en "seccion" que el último (8 registros)

7- Muestre nombre, estado civil y seccion de los primeros 4 empleados ordenados por estado civil y sección (4 registros)

8- Realice la misma consulta anterior pero incluya todos los valores iguales al último registro retornado (5 registros)

**Solución:**

if object\_id('empleados') is not null

drop table empleados;

create table empleados (

documento varchar(8) not null,

nombre varchar(30),

estadocivil char(1),--c=casado, s=soltero,v=viudo

seccion varchar(20)

);

insert into empleados

values ('22222222','Alberto Lopez','c','Sistemas');

insert into empleados

values ('23333333','Beatriz Garcia','c','Administracion');

insert into empleados

values ('24444444','Carlos Fuentes','s','Administracion');

insert into empleados

values ('25555555','Daniel Garcia','s','Sistemas');

insert into empleados

values ('26666666','Ester Juarez','c','Sistemas');

insert into empleados

values ('27777777','Fabian Torres','s','Sistemas');

insert into empleados

values ('28888888','Gabriela Lopez',null,'Sistemas');

insert into empleados

values ('29999999','Hector Garcia',null,'Administracion');

select top 5 \*from empleados;

select top 4 nombre,seccion

from empleados

order by seccion;

select top 4 with ties nombre,seccion

from empleados

order by seccion;

select top 4 nombre,estadocivil,seccion

from empleados

order by estadocivil,seccion;

select top 4 with ties nombre,estadocivil,seccion

from empleados

order by estadocivil,seccion;