



6. Agrupamiento de datos

Temario

- 6.1 Funciones de renglón múltiple
- 6.2. Agrupando datos (clausula group by)
- 6.3 Condicionando grupos de datos (clausula having).

6.1 Funciones de renglón múltiple

- Las funciones para grupos operan con un conjunto de renglones para devolver un solo resultado.
- A diferencia de las funciones de renglón simples, las funciones para grupos, operan con un conjunto de renglones para obtener un resultado por grupo.
- El conjunto de renglones debe ser toda la tabla o la tabla dividida en grupos

6.1 Funciones de renglón múltiple

- Cada función acepta un argumento

| Función | Descripción |
|---------------------------------------|---|
| AVG([DISTINCT ALL] <i>n</i>) | Valor promedio de <i>n</i> , ignora valores null |
| COUNT({*[DISTINCT ALL] <i>expr</i> }) | Número de renglones, donde <i>expr</i> evalúe otro valor diferente de null. Cuenta todos los renglones seleccionados utilizando *, incluyendo renglones duplicados con valores nulos. |
| MAX([DISTINCT ALL] <i>expr</i>) | Valor máximo de <i>expr</i> , ignorando valores null |
| MIN([DISTINCT ALL] <i>expr</i>) | Valor mínimo de <i>expr</i> , ignorando valores null |
| SUM([DISTINCT ALL] <i>n</i>) | Suma los valores de <i>expr</i> , ignorando valores null |

6.1 Funciones de renglón múltiple

- DISTINCT realiza la función de considerar solo aquellos renglones no duplicados
- ALL considera los renglones duplicados. Por default es ALL y no necesita ser especificado.
- Todas las funciones de grupo excepto COUNT(*) ignoran los valores null.
- Para sustituir los valores null, se utiliza la función ISNULL.
- Se presenta en formato de una consulta que usa función de grupo:

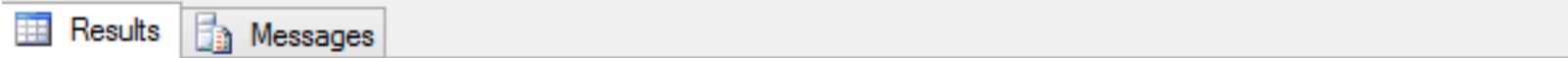
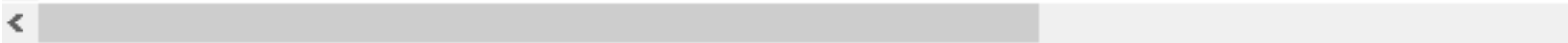
```
SELECT columna, funcion_de_grupo(columna)
FROM   tabla
[WHERE condicion]
[ORDER BY column];
```

6.1.1 Funciones de agregación (AVG, MAX, MIN, SUM)

- Se puede utilizar las funciones AVG, MAX, MIN y SUM con columnas que almacenan datos numéricos
- Las funciones AVG y SUM se pueden usar sólo con números
- Las funciones MIN Y MAX se pueden usar con fechas
- Ejemplo: muestra el promedio, el máximo, el mínimo y la suma de los salarios de los trabajadores

6.1.1 Funciones de agregación (AVG, MAX, MIN, SUM)

```
select AVG(e.sal) as promedio, MAX(e.sal) as maximo, MIN(e.sal) as minimo,  
SUM(e.sal) as suma from EMP e ;
```



| | promedio | maximo | minimo | suma |
|---|-------------|---------|--------|----------|
| 1 | 2068.333333 | 5000.00 | 800.00 | 31025.00 |

6.1.2 Funciones de agregación (COUNT)

- La función COUNT tiene dos formatos:
 - COUNT (*): regresa el número de renglones en una tabla, incluyendo renglones que contengan valores null
 - COUNT (exp.): a diferencia de COUNT (expr), esta regresa el número de renglones no nulos en la columna identificada por expr.

6.1.2 Funciones de agregación (COUNT)

```
select COUNT(*) from EMP where EMP.DEPTNO=30;  
select COUNT(COMM) from EMP where EMP.DEPTNO=30;  
select COUNT(*) from EMP;  
select COUNT(COMM) from EMP;
```

6.1.2 Funciones de agregación (COUNT)

```
SELECT COUNT (DISTINCT (DEPTNO) ) FROM EMP;
```

| Results | |
|------------------|---|
| Messages | |
| (No column name) | |
| 1 | 3 |

6.2 Group By

- Hasta ahora, todas las funciones de grupo tratan a la tabla como un solo grupo de información.
- En ocasiones, se necesitará dividir la tabla en pequeños grupos de información. Esto se puede realizar utilizando la cláusula GROUP BY.
- GROUP BY reorganiza en el sentido lógico la tabla representada por la cláusula FROM formando particiones o grupos de manera que dentro de un grupo dado todas las filas tengan el mismo valor en el campo GROUP BY.

6.2 Group By

EMP

| DEPTNO | SAL |
|--------|------|
| 10 | 2450 |
| 10 | 5000 |
| 10 | 1300 |
| 20 | 800 |
| 20 | 1100 |
| 20 | 3000 |
| 20 | 3000 |
| 20 | 2975 |
| 30 | 1600 |
| 30 | 2850 |
| 30 | 1250 |
| 30 | 950 |
| 30 | 1500 |
| 30 | 1250 |

2916.667

2175

1566.667

"promedio salarial en la tabla EMP
por cada departamento"

| deptno | avg(sal) |
|--------|----------|
| 10 | 2916.667 |
| 20 | 2175.000 |
| 30 | 1566.667 |

6.2 Group By

- Se puede utilizar la cláusula GROUP BY para dividir en pequeños grupos de información una tabla. Entonces se puede utilizar las funciones de grupo para resumir la información de estos grupos.

```
SELECT columna,funcion_de_agrupacion (columna)
FROM   tabla
[WHERE condicion]
[GROUP BY expresion group_by]
[ORDER BY column];
```

6.2 Group By

- Group By especifica las columnas por las que se efectuará el agrupamiento de los renglones de la tabla.
- Utilizando WHERE se puede pre-excluir renglones antes de ser divididos en grupos
- Por defecto, los renglones son ordenados ascendentemente por las columnas especificadas en GROUP BY. Se puede alterar este orden utilizando ORDEN BY

6.2 Group By

- Cuando se utilice la cláusula GROUP BY se debe asegurar de que las columnas en la lista de SELECT que no estén en funciones de grupo se encuentren en la cláusula GROUP BY.
- El ejemplo muestra el número de departamento y el promedio salarial por cada departamento.

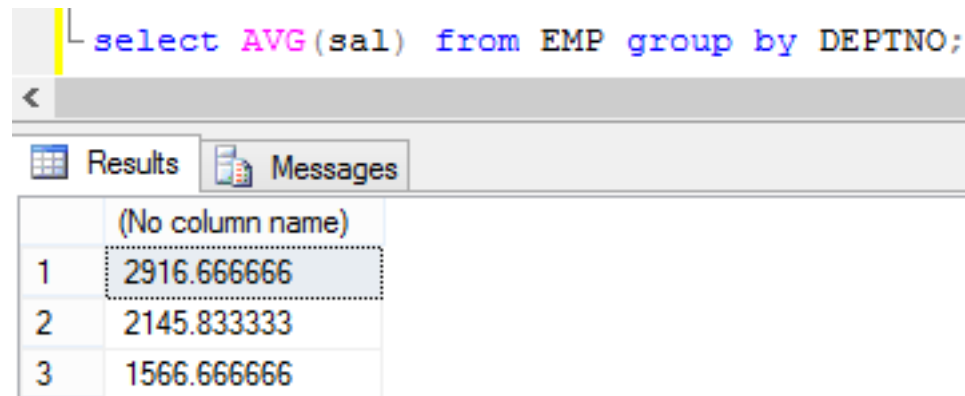
6.2 Group By

```
SQL> select EMP.DEPTNO, AVG(sal) from EMP group by DEPTNO;
```

| < | | |
|------------------|--------|------------------|
| Results Messages | | |
| | DEPTNO | (No column name) |
| 1 | 10 | 2916.666666 |
| 2 | 20 | 2145.833333 |
| 3 | 30 | 1566.666666 |

6.2 Group By

- Las columnas que aparecen en GROUP BY no necesariamente deben aparecer en la lista de SELECT.



The screenshot shows a SQL query execution window. The query is: `select AVG(sal) from EMP group by DEPTNO;`. Below the query, there are two tabs: "Results" and "Messages". The "Results" tab is active, displaying a table with three rows. The first row has a header "(No column name)". The subsequent rows show the average salary for each department: 1 (2916.666666), 2 (2145.833333), and 3 (1566.666666).

| | (No column name) |
|---|------------------|
| 1 | 2916.666666 |
| 2 | 2145.833333 |
| 3 | 1566.666666 |

6.2 Group By

- En ciertas ocasiones se necesitará ver los resultados de grupos más pequeños dentro de los mismos grupos. El ejemplo muestra un reporte que despliega el salario total para cada puesto, dentro de cada departamento.

EMP

| DEPTNO | JOB | SAL |
|--------|-----------|------|
| 10 | MANAGER | 2450 |
| 10 | PRESIDENT | 5000 |
| 10 | CLERK | 1300 |
| 20 | CLERK | 800 |
| 20 | CLERK | 1100 |
| 20 | ANALYST | 3000 |
| 20 | ANALYST | 3000 |
| 20 | MANAGER | 2975 |
| 30 | SALESMAN | 1600 |
| 30 | SALESMAN | 1250 |
| 30 | SALESMAN | 1500 |
| 30 | SALESMAN | 1250 |
| 30 | MANAGER | 2850 |
| 30 | CLERK | 850 |

"sumar los salarios de la
tabla EMP por puesto,
agrupados por departamento"

| DEPTNO | JOB | SUM(SAL) |
|--------|-----------|----------|
| 10 | CLERK | 1300 |
| 10 | MANAGER | 2450 |
| 10 | PRESIDENT | 5000 |
| 20 | ANALYST | 6000 |
| 20 | CLERK | 1900 |
| 20 | MANAGER | 2975 |
| 30 | CLERK | 950 |
| 30 | MANAGER | 2950 |
| 30 | SALESMAN | 5600 |

6.3 Group By

```
select e.DEPTNO,e.JOB,SUM(e.SAL) as reporte  
from EMP e group by e.DEPTNO,e.JOB  
order by e.DEPTNO;
```

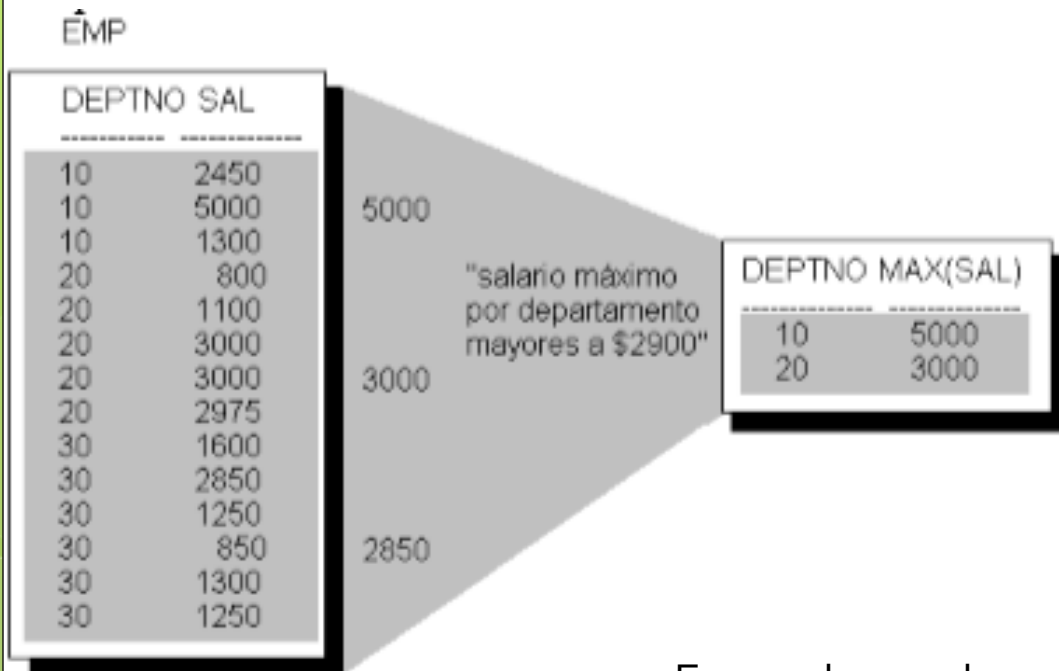
| | DEPTNO | JOB | reporte |
|----|--------|-----------|---------|
| 1 | 10 | CLERK | 1300.00 |
| 2 | 10 | MANAGER | 2450.00 |
| 3 | 10 | PRESIDENT | 5000.00 |
| 4 | 20 | ANALYST | 6000.00 |
| 5 | 20 | CLERK | 1900.00 |
| 6 | 20 | instruc | 2000.00 |
| 7 | 20 | MANAGER | 2975.00 |
| 8 | 30 | CLERK | 950.00 |
| 9 | 30 | MANAGER | 2850.00 |
| 10 | 30 | SALESMAN | 5600.00 |

6.3 Clausula HAVING

- De la misma forma que WHERE elimina renglones en un SELECT, se utiliza HAVING para condicionar resultados por grupo. Si se especifica HAVING deberá haberse especificado también GROUP BY.

6.3 Clausula HAVING

- Dada la siguiente representación:



Para encontrar el salario máximo de cada departamento y que muestre solo aquellos departamentos cuyo salario máximo sea mayor a \$2900, se necesita realizar lo siguiente:

- Encontrar el salario máximo por cada departamento agrupando por número de departamento.
- Restringir cada resultado de grupo, para que muestre solo aquellos que el salario máximo sea mayor a \$2900.

6.3 Clausula HAVING

- Cuando se utiliza la cláusula HAVING para restringir grupos, se realiza lo siguiente:
- Se agrupan los renglones Se aplican las funciones de grupo a cada grupo Se muestran los grupos que cumplen la condición HAVING

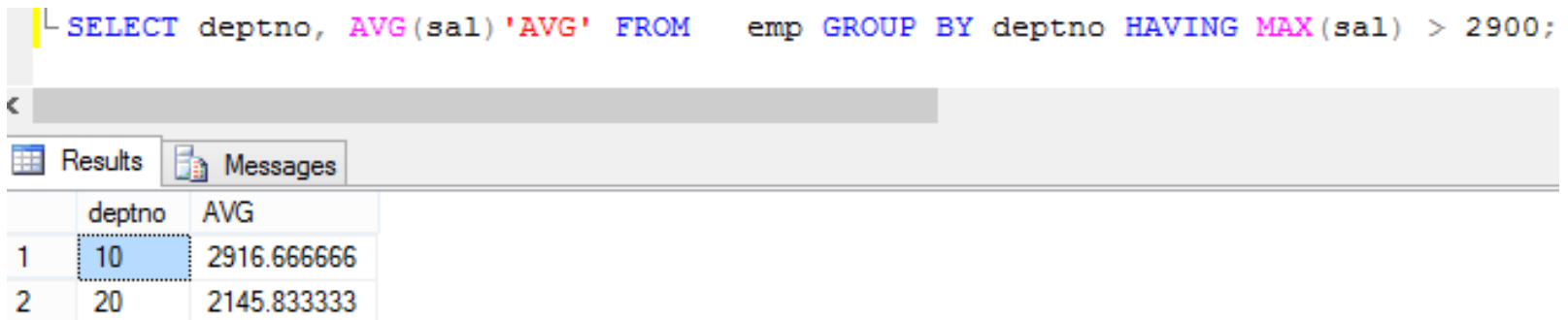
```
SELECT deptno, MAX(sal) FROM emp GROUP BY deptno HAVING MAX(sal) > 2900;
```

| Results | | | Messages | | |
|---------|--------|------------------|----------|--|--|
| | deptno | (No column name) | | | |
| 1 | 10 | 5000.00 | | | |
| 2 | 20 | 3000.00 | | | |

6.3 Clausula HAVING

- El ejemplo siguiente muestra los números de departamento y el promedio salarial para aquellos departamentos cuyo salario máximo sea mayor a \$2900.

```
SELECT deptno, AVG(sal) 'AVG' FROM emp GROUP BY deptno HAVING MAX(sal) > 2900;
```



The screenshot shows a database query result in a window with two tabs: 'Results' and 'Messages'. The 'Results' tab is active, displaying a table with two columns: 'deptno' and 'AVG'. The table contains two rows of data. The first row shows department 10 with an average salary of 2916.666666. The second row shows department 20 with an average salary of 2145.833333. The cell containing '10' in the first row is highlighted with a blue border.

| | deptno | AVG |
|---|--------|-------------|
| 1 | 10 | 2916.666666 |
| 2 | 20 | 2145.833333 |

6.3 Clausula HAVING

- El siguiente ejemplo muestra el puesto y la suma total salarial por puesto, para aquellos puestos con una nómina total mayor a \$5000. En el ejemplo se elimina a los puestos SALESMAN, se ordena por suma total de cada puesto.

```
SELECT JOB, SUM(sal) PAYROLL
FROM emp
WHERE job not like 'SALES%'
GROUP BY job
HAVING SUM(sal) > 5000
ORDER BY SUM(sal);
```



Results



Messages

| | JOB | PAYROLL |
|---|---------|---------|
| 1 | ANALYST | 6000.00 |
| 2 | MANAGER | 8275.00 |