People matter, results count.

For internal use only

Modulo SQL

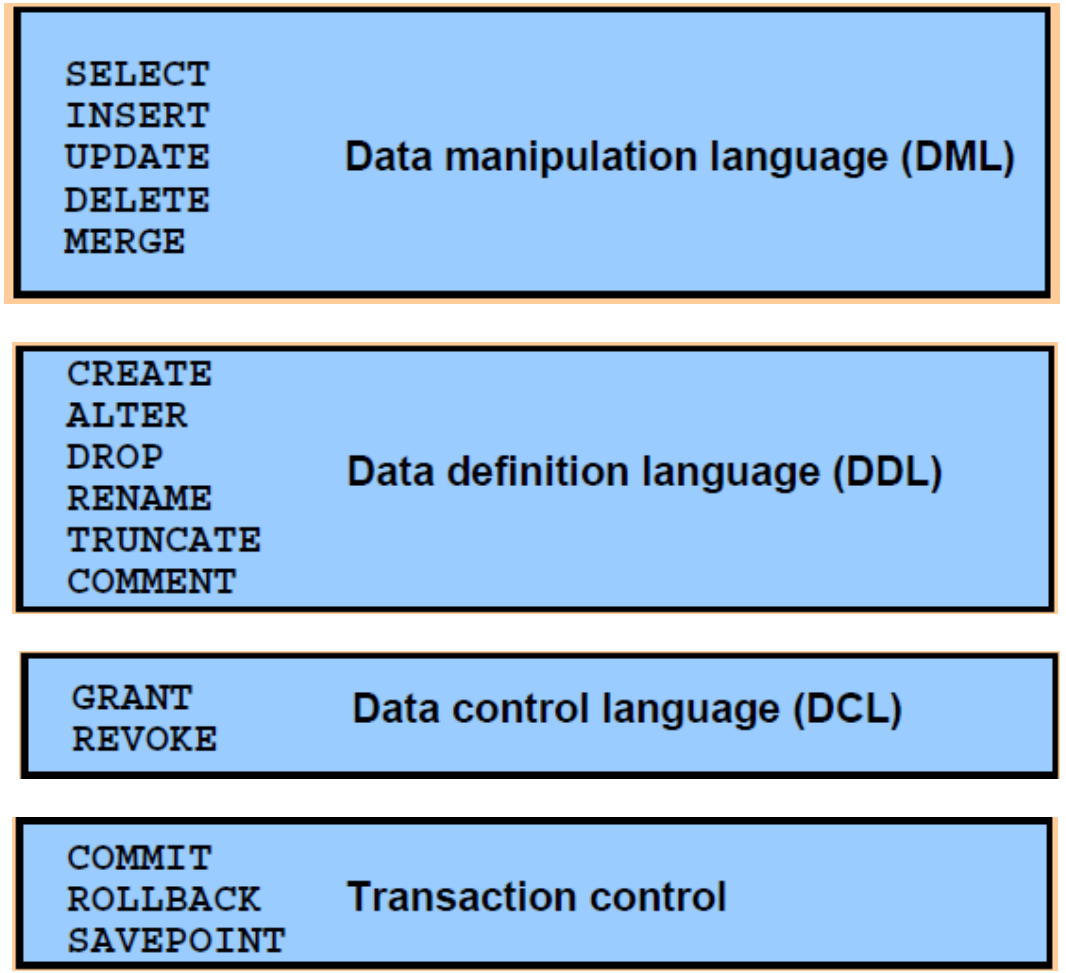
Consultas Generales de tabla unica

El lenguaje SQL.

¿Qué es SQL (Structured Query Language)?

- Lenguaje que permite el acceso a las bases de datos relacionales.
- Aprovecha al máximo el poder y la flexibilidad de los Sistemas Relacionales.
- Tips de escritura para sentencias SQL:
  - No diferencia mayúsculas de minúsculas.
  - Las sentencias pueden estar escritas en una o más líneas.
  - Las palabras claves no pueden estar abreviadas o divididas en más de una línea.
  - Las cláusulas se ubican generalmente en diferentes líneas para tener mayor claridad en la comprensión de la sentencia.

Las Sentencias SQL: tipos y para qué se usan ?



Manipulación de Datos.

Permiten crear-modificar-borrar objetos de la BD.

Permiten configurar los privilegios de usuarios de BD.

Permiten manejar los cambios realizados por sentencias del tipo DML.

Sentencia SELECT.

SELECT: ejemplo.

- **Sobre la tabla EMPLOYEES se plantea:**
  - Qué empleados (nombre y apellido) tienen un sueldo mayor a $14000 ?
- **Qué preguntas nos haríamos para resolverlo?**
  - Qué datos nos están pidiendo?
  - Dónde están esos datos ?
  - Qué requisistos deben cumplir los registros ?
- **Sintaxis SQL**
  - En SQL la forma de operar es parecida y se obtiene mediante la siguiente consulta :

SELECT: Sintaxis SQL

SELECT ***first_name, last_name, salary***

Qué datos nos están pidiendo ?

FROM ***employees***

Dónde están esos datos ?

WHERE ***salary > 14000***

Qué requisistos deben cumplir los registros ?

SELECT

- **Sintaxis general**

SELECT *|{*[DISTINCT] column*|*expression [alias],...}*
[**FROM** *tablas]*
[**WHERE** *condiciones de búsqueda o filtro]*
[**GROUP BY** *expresión de agrupación]*
[**HAVING** *condición de filtro o búsqueda]*
[**ORDER BY** *condición de filtro o búsqueda]*

Usando expresiones aritméticas.

```
SELECT last_name, salary, salary + 300
FROM   employees;
```

LAST_NAME	SALARY	SALARY+300
King	24000	24300
Kochhar	17000	17300
De Haan	17000	17300
Hunold	9000	9300
Ernst	6000	6300

...

20 rows selected.

Null Value.

- **Es un valor que no está disponible, que es desconocido, inaplicable.**
  - No es lo mismo que un espacio en blanco ó cero.
  - Las expresiones aritméticas que contienen un valor nulo evalúan como null.

```
SELECT last_name, 12*salary*commission_pct
FROM   employees;
```

LAST_NAME	12*SALARY*COMMISSION_PCT
King	
Kochhar	
...	
Zlotkey	25200
Abel	39600
Taylor	20640
...	
Gietz	

20 rows selected.

Uso del Distinct : Filas Duplicadas.

- **El comportamiento por defecto de la sentencia SELECT es mostrar todas las filas incluso las repetidas ó duplicadas.**

```
SELECT department_id
FROM   employees;
```

Todos los departamentos

DEPARTMENT_ID
90
90
90

...

20 rows selected

```
SELECT DISTINCT department_id
FROM   employees;
```

Los distintos departamentos

DEPARTMENT_ID
10
20
50

Condiciones de Comparación.

Operator	Meaning
=	Equal to
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to
<>	Not equal to
BETWEEN ...AND...	Between two values (inclusive)
IN(set)	Match any of a list of values
LIKE	Match a character pattern
IS NULL	Is a null value

Usando Condiciones de Comparación: ejemplo

- **BETWEEN**
  - Utilizamos la condición para mostrar las filas en donde la condición especificada en el where se encuentran entre un rango de valores.

```
SELECT last_name, salary
FROM   employees
WHERE  salary BETWEEN 2500 AND 3500 ;
```

Lower limit Upper limit

LAST_NAME	SALARY
Rajs	3500
Davies	3100
Matos	2600
Vargas	2500

Usando Condiciones de Comparación: ejemplo

- **IN**
  - Utilizamos la condición de pertenencia para testear condiciones de una lista.

```
SELECT employee_id, last_name, salary, manager_id
FROM   employees
WHERE  manager_id IN (100, 101, 201) ;
```

EMPLOYEE_ID	LAST_NAME	SALARY	MANAGER_ID
202	Fay	6000	201
200	Whalen	4400	101
205	Higgins	12000	101
101	Kochhar	17000	100
102	De Haan	17000	100
124	Mourgos	5800	100
149	Zlotkey	10500	100
201	Hartstein	13000	100

8 rows selected.

Usando Condiciones de Comparación: ejemplo

- **LIKE**
  - Este operador se aplica a datos de tipo String y es capaz de hallar coincidencias dentro de una cadena bajo un patrón dado.
  - Las condiciones de búsqueda pueden contener caracteres literales ó números.

```
SELECT   first_name
FROM     employees
WHERE    first_name LIKE 'S%' ;
```

```
SELECT last_name
FROM   employees
WHERE  last_name LIKE '_o%' ;
```

LAST_NAME
Kochhar
Lorentz
Mourgos

Usando Condiciones de Comparación: ejemplo

- **IS NULL**
  - Para testear valores nulos.

```
SELECT last_name, manager_id
FROM   employees
WHERE  manager_id IS NULL ;
```

LAST_NAME	MANAGER_ID
King	

Condiciones Lógicas.

Operator	Meaning
AND	Returns TRUE if *both* component conditions are true
OR	Returns TRUE if *either* component condition is true
NOT	Returns TRUE if the following condition is false

Usando Condiciones Lógicas: ejemplo

- **AND**
  - El operador requiere que ambas condiciones sean verdaderas.

```
SELECT employee_id, last_name, job_id, salary
FROM   employees
WHERE  salary >=10000
AND    job_id LIKE '%MAN%' ;
```

EMPLOYEE_ID	LAST_NAME	JOB_ID	SALARY
149	Zlotkey	SA_MAN	10500
201	Hartstein	MK_MAN	13000

Usando Condiciones Lógicas: ejemplo

- **OR**
  - El operador requiere que alguna de las condiciones sean verdaderas.

```
SELECT employee_id, last_name, job_id, salary
FROM   employees
WHERE  salary >= 10000
OR     job_id LIKE '%MAN%' ;
```

EMPLOYEE_ID	LAST_NAME	JOB_ID	SALARY
100	King	AD_PRES	24000
101	Kochhar	AD_VP	17000
102	De Haan	AD_VP	17000
124	Mourgos	ST_MAN	5800
149	Zlotkey	SA_MAN	10500
174	Abel	SA_REP	11000
201	Hartstein	MK_MAN	13000
205	Higgins	AC_MGR	12000

8 rows selected.

Usando Condiciones Lógicas: ejemplo

- **NOT**

```
SELECT last_name, job_id
FROM   employees
WHERE  job_id
       NOT IN ('IT_PROG', 'ST_CLERK', 'SA_REP') ;
```

LAST_NAME	JOB_ID
King	AD_PRES
Kochhar	AD_VP
De Haan	AD_VP
Mourgos	ST_MAN
Zlotkey	SA_MAN
Whalen	AD_ASST
Hartstein	MK_MAN
Fay	MK_REP
Higgins	AC_MGR
Gietz	AC_ACCOUNT

10 rows selected.

Ordenación: cláusula ORDER BY.

- **Permite ordenar las filas recuperadas.**
  - ASC: orden ascendente, default.
  - DESC: orden descendente.
- **Es la última claúsula en la sentencia SELECT.**

```
SELECT   last_name, job_id, department_id, hire_date
FROM     employees
ORDER BY hire_date ;
```

LAST_NAME	JOB_ID	DEPARTMENT_ID	HIRE_DATE
King	AD_PRES	90	17-JUN-87
Whalen	AD_ASST	10	17-SEP-87
Kochhar	AD_VP	90	21-SEP-89
Hunold	IT_PROG	60	03-JAN-90
Ernst	IT_PROG	60	21-MAY-91

...

- **Se puede ordenar por varias columnas.**

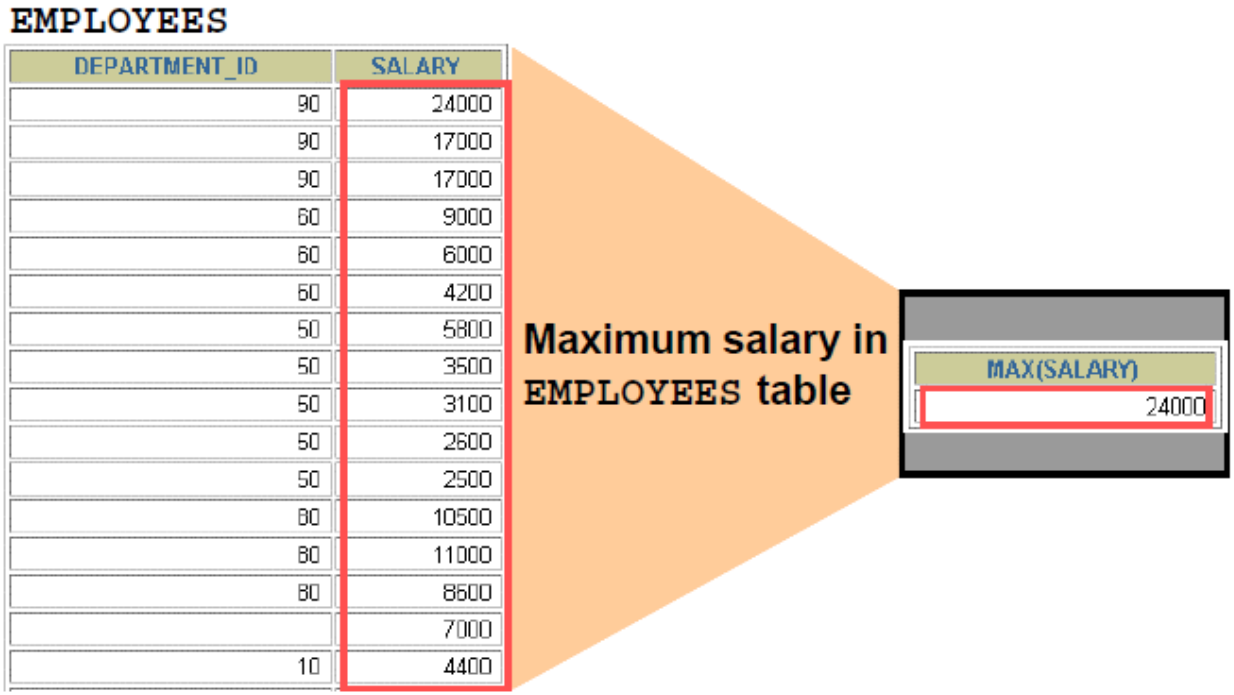
```
SELECT last_name, department_id, salary
FROM   employees
ORDER BY department_id, salary DESC;
```

3

Funciones de Grupo.

Qué son las funciones de grupo ?

- **Las funciones de grupo operan sobre un conjunto de filas y devuelven un resultado por grupo.**



Funciones de Grupo.

- **Tipos:**
  - COUNT
  - MAX/MIN
  - SUM
  - AVG (average = promedio)

- **Sintaxis**

```
SELECT      [column,] group_function(column), ...
FROM        table
[WHERE      condition]
[GROUP BY   column]
[ORDER BY   column];
```

Ejemplos.

```
SELECT AVG(salary), MAX(salary),
       MIN(salary), SUM(salary)
FROM   employees
WHERE  job_id LIKE '%REP%';
```

```
SELECT MIN(hire_date), MAX(hire_date)
FROM       employees;
```

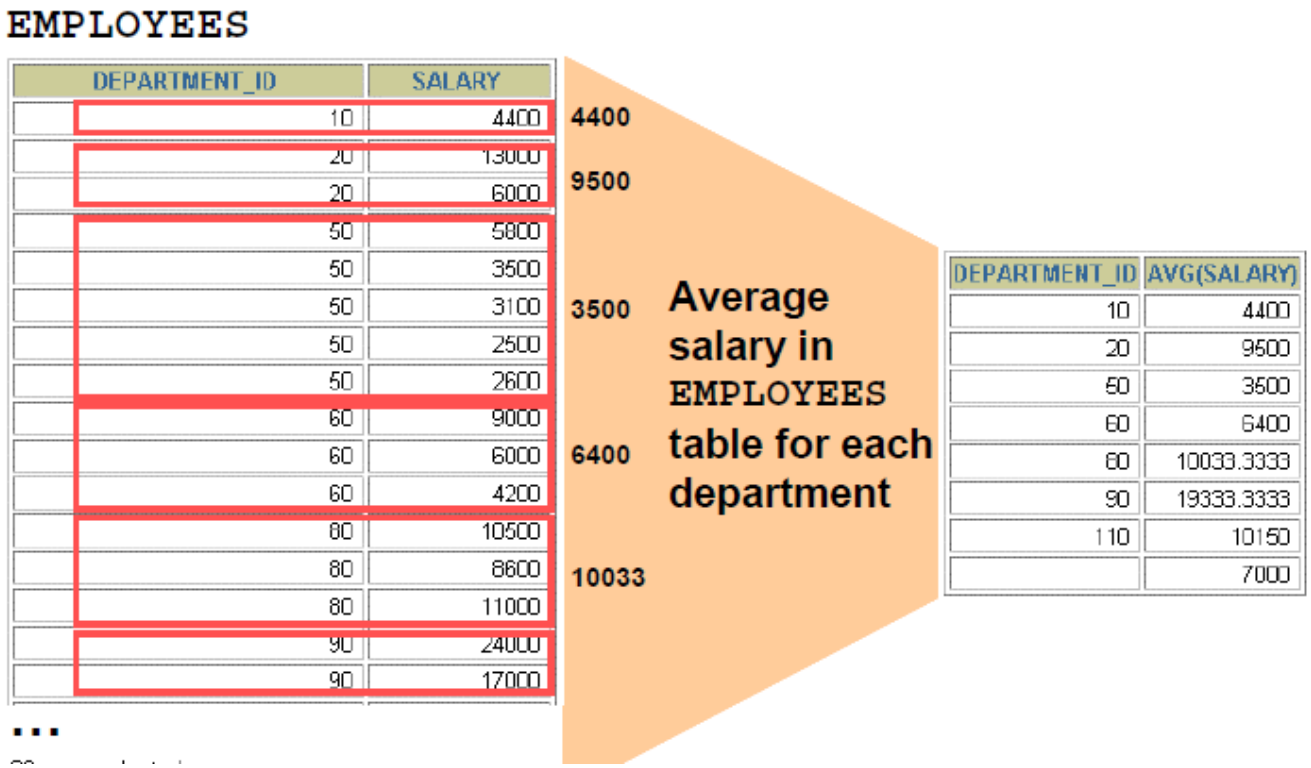
- **Uso de la función COUNT.**
  - Count(*) retorna el número de filas de una tabla
  - Count (expr) retorna el número de filas no incluyendo los que tengan valor nulo en el campo por el cual se hace el count.

```
SELECT COUNT(*)
FROM   employees
WHERE  department_id = 50;
```

```
SELECT COUNT(commission_pct)
FROM   employees
WHERE  department_id = 80;
```

Creando grupos de datos

- **Agrupamos resultados por departamento.**
  - Claúsula GROUP BY



GROUP BY

- **Sintaxis**
  - Podemos dividir las filas de una tabla en grupos más pequeños usando la cláusula GROUP BY.

```
SELECT     column, group_function(column)
FROM       table
[WHERE     condition]
[GROUP BY group_by_expression]
[ORDER BY column];
```

- Si un campo aparece en la cláusula SELECT junto con las funciones de totalización, entonces debemos forzosamente agrupar por ese campo, o lo que es lo mismo, debe formar parte de la cláusula GROUP BY.
- Un campo por el que agrupamos puede omitirse en la cláusula SELECT

```
SELECT   department_id, AVG(salary)
FROM     employees
GROUP BY department_id ;
```

Errores comunes cláusula GROUP BY

```
SELECT department_id, COUNT(last_name)
FROM   employees;
```

```
SELECT department_id, COUNT(last_name)
       *
ERROR at line 1:
ORA-00937: not a single-group group function
```

Column missing in the `GROUP BY` clause

Agrupando por más de una columna

- **Podemos usar más de un criterio para agrupar**

EMPLOYEES

DEPARTMENT_ID	JOB_ID	SALARY
90	AD_PRES	24000
90	AD_VP	17000
90	AD_VP	17000
60	IT_PROG	9000
60	IT_PROG	6000
60	IT_PROG	4200
50	ST_MAN	5800
50	ST_CLERK	3500
50	ST_CLERK	3100
50	ST_CLERK	2600
50	ST_CLERK	2500
80	SA_MAN	10500
80	SA_REP	11000
80	SA_REP	8600
...		
20	MK_REP	6000
110	AC_MGR	12000
110	AC_ACCOUNT	8300

Add the salaries in the EMPLOYEES table for each job, grouped by department

DEPARTMENT_ID	JOB_ID	SUM(SALARY)
10	AD_ASST	4400
20	MK_MAN	13000
20	MK_REP	6000
50	ST_CLERK	11700
50	ST_MAN	5800
60	IT_PROG	19200
80	SA_MAN	10500
80	SA_REP	19600
90	AD_PRES	24000
90	AD_VP	34000
110	AC_ACCOUNT	8300
110	AC_MGR	12000
	SA_REP	7000

13 rows selected.

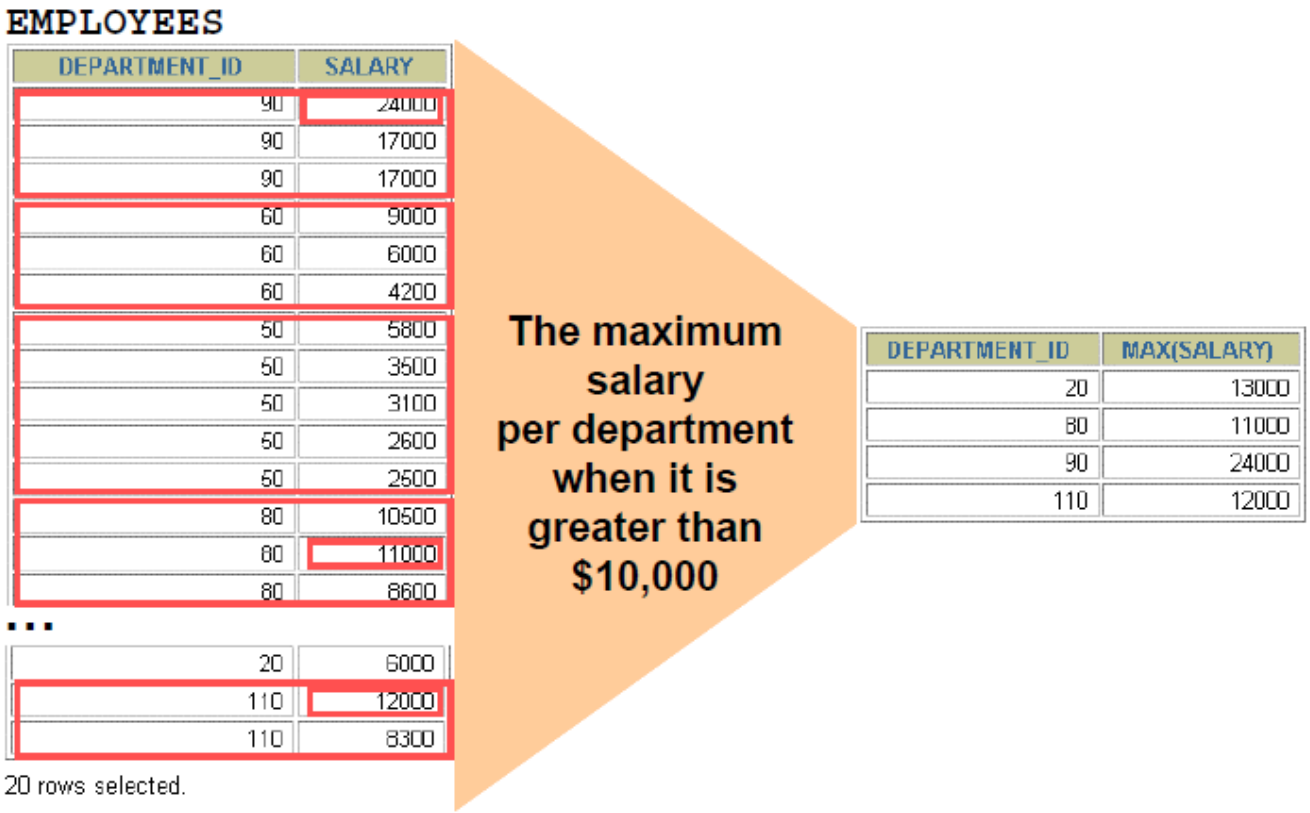
Usando GROUP BY sobre múltiples columnas.

- **Podemos sumar los salarios de los empleados agrupados por trabajo (job_id) y por el departamento al cual pertenecen.**

```
SELECT   department_id dept_id, job_id, SUM(salary)
FROM     employees
GROUP BY department id, job id ;
```

Filtrar cálculos de totalización: HAVING.

- **Permite restringir el grupo de resultados.**
  - Permite de todas las filas resultantes ocultar las que no nos interesan y mostrar el resto. Sería como un filtro en segunda instancia una vez resuelta la consulta.



HAVING

- **Sintaxis**

```
SELECT     column, group_function
FROM       table
[WHERE     condition]
[GROUP BY  group_by_expression]
[HAVING    group_condition]
[ORDER BY  column];
```

```
SELECT   department_id, MAX(salary)
FROM     employees
GROUP BY department_id
HAVING   MAX(salary)>10000 ;
```

- **Se usa mucho para**
  - Obtener registros duplicados en una tabla.
  - Cómo puedo saber aquellos nombres que se repiten de la tabla empleados ?