



LENGUAJE SQL



Índice

1. Consultas Sencillas
 1. Utilización de Alias
2. Condiciones de Selección
3. Funciones predefinidas en Expresiones y Condiciones
 1. Funciones Numéricas o Aritméticas.
 2. Funciones de Caracteres
 3. Funciones de Fechas
 4. Funciones de Comparación
 5. Otras Funciones
4. Ordenación
5. Selección con Limitación de Filas



1. Consultas Sencillas

- Realizar una consulta en SQL consiste en recuperar u obtener aquellos datos que, almacenados en filas y columnas de una o varias tablas de una base de datos, cumplen unas determinadas especificaciones. Para realizar cualquier consulta se utiliza la sentencia SELECT.
- La consulta más sencilla es la instrucción de selección. Las cláusulas que aparecen en ellas, SELECT y FROM, son obligatorias.
- La consulta más sencilla consiste en recuperar una o varias columnas o expresiones relacionadas de una tabla
- El formato inicial de la sentencia SELECT es:
 - **SELECT [ALL/DISTINCT] *ExpresionColumna* [, *ExpresionColumna*....]**
 - **FROM *NombreTabla*;**



1. Consultas Sencillas

- Donde:
 - ***ExpresionColumna:*** es un conjunto de nombres de columnas, literales o constantes, operadores, funciones y paréntesis.
 - ***NombreTabla:*** es el nombre de la tabla de la que queremos seleccionar filas.
 - **ALL:** obtiene los valores de todos los elementos seleccionados en todas las filas, aunque sean repetidos. Es la opción por defecto y no suele escribirse
 - **DISTINCT:** obtiene los valores no repetidos de todos los elementos.
- En caso de que queramos mostrar varias expresiones, estas irán separadas por comas. Este es el formato mínimo con las cláusulas SELECT y FROM que son siempre obligatorias. A este formato mínimo le iremos añadiendo otras cláusulas opcionales en los siguientes apartados y siguientes temas.



1. Consultas Sencillas

- En lugar de las expresiones puede parecer el carácter * que indica que deben seleccionarse todas las columnas de la tabla. No se ha escrito para no complicar el formato inicial y hacerlo más fácil de comprender. El formato sería:
 - **SELECT { * | [ALL/DISTINCT] *ExpresionColumna* [,*ExpresionColumna*....] }**
 - **FROM *NombreTabla***
- Veamos algunos ejemplos



1. Consultas Sencillas

- Veamos algunos ejemplos
 - Obtener todos los empleados de la tabla empleados con todos sus datos.

```
mysql> SELECT *  
-> FROM empleados;
```

EMP_NO	APELLIDO	OFICIO	DIRECTOR	FECHA_ALTA	SALARIO	COMISION	DEP_NO
7499	ALONSO	VENDEDOR	7698	1981-02-23	1400.00	400.00	30
7521	LOPEZ	EMPLEADO	7782	1981-05-08	1350.50	NULL	10
7654	MARTIN	VENDEDOR	7698	1981-09-28	1500.00	1600.00	30
7698	GARRIDO	DIRECTOR	7839	1981-05-01	3850.12	NULL	30
7782	MARTINEZ	DIRECTOR	7839	1981-06-09	2450.00	NULL	10
7839	REY	PRESIDENTE	NULL	1981-11-17	6000.00	NULL	10
7844	CALVO	VENDEDOR	7698	1981-09-08	1800.00	0.00	30
7876	GIL	ANALISTA	7782	1982-05-06	3350.00	NULL	20
7900	JIMENEZ	EMPLEADO	7782	1983-03-24	1400.00	NULL	20



1. Consultas Sencillas

- Obtener los números de empleados, los apellidos y el número de departamento de todos los empleados de la tabla empleados.

```
mysql> SELECT emp_no, apellido, dep_no  
-> FROM empleados;
```

emp_no	apellido	dep_no
7499	ALONSO	30
7521	LOPEZ	10
7654	MARTIN	30
7698	GARRIDO	30
7782	MARTINEZ	10
7839	REY	10
7844	CALVO	30
7876	GIL	20
7900	JIMENEZ	20



1. Consultas Sencillas

- Obtener los departamentos diferentes que hay en la tabla empleados

```
mysql> SELECT DISTINCT dep_no  
-> FROM empleados;
```

```
+-----+  
| dep_no |  
+-----+  
|      10 |  
|      20 |  
|      30 |  
+-----+
```

- Obtener los diferentes oficios que hay en cada departamento de la tabla empleados

```
mysql> SELECT DISTINCT oficio, dep_no  
-> FROM empleados;
```

```
+-----+-----+  
| oficio      | dep_no |  
+-----+-----+  
| VENDEDOR    |      30 |  
| EMPLEADO    |      10 |  
| DIRECTOR    |      30 |  
| DIRECTOR    |      10 |  
| PRESIDENTE  |      10 |  
| ANALISTA    |      20 |  
| EMPLEADO    |      20 |  
+-----+-----+
```




1.1- Utilización de Alias

- `SELECT [ALL/DISTINCT] ExpresionColumna [AS] AliasColumna[,
ExpresionColumna [AS] AliasColumna]....]`
`FROM NombreTabla [AliasTabla];`
- *Los alias de columna y de tabla aparecen entre corchetes porque son opcionales.*
- Donde:
 - ***AliasTabla:*** SQL permite asignar otro nombre a la misma tabla, dentro de la misma consulta.
 - ***AliasColumna:*** se escribe detrás de la expresión de columna, separado de ella al menos por un espacio. Puede ir entre comillas dobles o sin comillas (si no lleva espacios en blanco y si solo es una palabra)
 - La cláusula AS que asigna el alias puede omitirse



1.1- Utilización de Alias

- Usar alias para una tabla es opcional cuando su finalidad consiste en simplificar su nombre original, y obligatorio en consultas cuya sintaxis lo requiera (más adelante lo utilizaremos).
- Usar alias para las columnas puede ser necesario porque los títulos o cabeceras que muestra la salida de una consulta para las columnas seleccionadas, se corresponden con los nombres de las columnas de las tablas o las expresiones y esto no siempre es muy visual. Para mejorar su legibilidad y estética se utilizan los alias de columna. También puede ser utilizado, en algunos casos, para renombrar la columna y utilizar este nombre posteriormente.



1.1- Utilización de Alias

- Veamos algunos ejemplos:
 - Mostrar el apellido y la fecha de alta de todos los empleados.
 - `mysql> SELECT apellido, fecha_alt`
`-> FROM empleados emp;`

```
+-----+-----+
| apellido | fecha_alt |
+-----+-----+
| ALONSO   | 1981-02-23 |
| LOPEZ    | 1981-05-08 |
| MARTIN   | 1981-09-28 |
| GARRIDO  | 1981-05-01 |
| MARTINEZ | 1981-06-09 |
| REY      | 1981-11-17 |
| CALVO    | 1981-09-08 |
| GIL      | 1982-05-06 |
| JIMENEZ  | 1983-03-24 |
+-----+-----+
```

- De momento no podemos ver la utilidad del alias de tabla, pero se verá más adelante.



1.1- Utilización de Alias

- Obtener el salario total anual (14 pagas) de los empleados de la empresa mostrando el mensaje Salario total
- Vemos, en este ejemplo, las diferentes posibilidades para escribir el alias.
 - Con la cláusula AS
 - `mysql> SELECT salario*14 AS Salario_total`
`FROM empleados;`

```
+-----+  
| Salario_total |  
+-----+  
|      18907.00 |  
|      19600.00 |  
|      19600.00 |  
|      21000.00 |  
|      25200.00 |  
|      34300.00 |  
|      46900.00 |  
|      53901.68 |  
|      84000.00 |  
+-----+
```



1.1- Utilización de Alias

- Con comillas dobles (admite el carácter blanco en el alias)
 - mysql> SELECT salario*14 "Salario total"
 - > FROM empleados;

```
+-----+
| Salario total |
+-----+
|      19600.00 |
|      18907.00 |
|      21000.00 |
|      53901.68 |
|      34300.00 |
|      84000.00 |
|      25200.00 |
|      46900.00 |
|      19600.00 |
+-----+
```



1.1- Utilización de Alias

- Con comillas simples (admite el carácter blanco en el alias)
 - mysql> SELECT salario*14 'Salario total'
- > FROM empleados;

```
+-----+
| Salario total |
+-----+
|      19600.00 |
|      18907.00 |
|      21000.00 |
|      53901.68 |
|      34300.00 |
|      84000.00 |
|      25200.00 |
|      46900.00 |
|      19600.00 |
+-----+
```



1.1- Utilización de Alias

- Mostrar el número de empleado, el apellido y el departamento de los empleados de la empresa.
 - `mysql> SELECT emp_no "Nº Empleado", apellido Apellido, dept_no Departamento`
`-> FROM empleados;`

Nº_Empleado	Apellido	Departamento
7499	ALONSO	30
7521	LOPEZ	10
7654	MARTIN	30
7698	GARRIDO	30
7782	MARTINEZ	10
7839	REY	10
7844	CALVO	30
7876	GIL	20
7900	JIMENEZ	20



2. Condiciones de Selección

- Para seleccionar las filas de la tabla sobre las que realizar una consulta, la cláusula WHERE permite incorporar una condición de selección a la sentencia SELECT.
- Muestra todas aquellas filas para las que el resultado de evaluar la condición de selección es VERDADERO
 - `SELECT [ALL/DISTINCT] ExpresionColumna [,ExpresionColumna...]`
`FROM NombreTabla`
[WHERE *CondicionSeleccion*];
- donde ***CondicionSeleccion*** es una expresión (conjunto de nombres de columnas, literales, operadores, funciones y paréntesis) cuyo resultado es *VERDADERO/FALSO/NULO*
- El funcionamiento es el siguiente: para cada fila se evalúa la condición de selección y si el resultado es VERDADERO se visualizan las expresiones indicadas.



2. Condiciones de Selección

- Veamos algunos ejemplo:
 - Seleccionar aquellos empleados cuyo apellido empiece por 'M' y tengan un salario entre 1000 y 2000 euros.
 - `mysql> SELECT emp_no "Nº Empleado", apellido "Apellido", dept_no "Departamento"`
 - > `FROM empleados`
 - > `WHERE apellido LIKE 'M%' AND salario BETWEEN 1000 AND 2000;`

Nº Empleado	Apellido	Departamento
7654	MARTIN	30

- El operador LIKE usado con '%' indica que puede sustituirse por cualquier grupo de caracteres



2. Condiciones de Selección

- Seleccionar aquellos empleados cuyo apellido incluya una 'A' en el segundo carácter.
 - `mysql> SELECT emp_no "Nº Empleado", apellido "Apellido", dept_no "Departamento"`
 - > `FROM empleados`
 - > `WHERE (apellido LIKE `_A%');`

+-----+-----+-----+		
N° Empleado	Apellido	Departamento
+-----+-----+-----+		
7654	MARTIN	30
7698	GARRIDO	30
7782	MARTINEZ	10
7844	CALVO	30
+-----+-----+-----+		

- El operador LIKE usado con `_` indica que ocupa la posición de un carácter.



2. Condiciones de Selección

- Seleccionar los empleados existentes en los departamentos 10 y 30.
 - `mysql> SELECT emp_no "Nº Empleado", apellido "Apellido", dept_no "Departamento"`
 - > `FROM empleados`
 - > `WHERE dept_no = 10 OR dept_no = 30;`

-----+-----+-----+		
Nº Empleado	Apellido	Departamento
-----+-----+-----+		
7499	ALONSO	30
7521	LOPEZ	10
7654	MARTIN	30
7698	GARRIDO	30
7782	MARTINEZ	10
7839	REY	10
7844	CALVO	30
-----+-----+-----+		



2. Condiciones de Selección

- También puede hacerse utilizando el operador IN: El operador IN comprueba si una determinada expresión toma alguno de los valores indicados entre paréntesis..
 - `mysql> SELECT emp_no "Nº Empleado", apellido "Apellido", dept_no "Departamento"`
 - > `FROM empleados`
 - > `WHERE dept_no IN(10,30);`



2. Condiciones de Selección

- Seleccionar los empleados que tienen de oficio ANALISTA

- mysql> SELECT emp_no , apellido , oficio

- > FROM empleados

- > WHERE oficio = 'ANALISTA';

```
+-----+-----+-----+
| emp_no | apellido | oficio  |
+-----+-----+-----+
|    7876 | GIL      | ANALISTA |
+-----+-----+-----+
```

- Aunque el campo oficio está grabado en mayúsculas obtenemos el mismo resultado si lo escribimos en minúsculas (MySQL no diferencia entre ambas)



3. Funciones predefinidas en Expresiones y Condiciones

- Hemos dicho que una expresión es un conjunto de nombres de columnas, literales o constantes, operadores, funciones y paréntesis, y una condición es una expresión cuyo resultado es *VERDADERO/FALSO/NULO*. Estas expresiones y condiciones nos aparecen en diferentes cláusulas de la sentencia select
- Dentro de las expresiones y de las condiciones podemos utilizar las funciones predefinidas enumeradas en el primer tema
- Estas funciones pueden ser utilizadas en todas las expresiones y condiciones con la única restricción determinada por los tipos de datos con los que operan y que devuelven.
- Vamos a ver algún ejemplo de cada uno de estos tipos de funciones dentro de las expresiones y las condiciones.



3.1- Funciones Numéricas o Aritméticas

- Operan con datos numéricos y el resultado es un número.
 - Visualizar los salarios de los empleados redondeados sin decimales
 - `mysql> SELECT apellido, ROUND(salario,0) "SALARIO SIN DECIMALES"`
`-> FROM empleados;`

apellido	SALARIO SIN DECIMALES
ALONSO	1400
LOPEZ	1350
MARTIN	1500
GARRIDO	3850
MARTINEZ	2450
REY	6000
CALVO	1800
GIL	3350
JIMENEZ	1400



3.1- Funciones Numéricas o Aritméticas

- Mostrar los datos de los empleados en los que su comisión sea múltiplo de 100, y no sea cero.
 - mysql> SELECT *
 - > FROM empleados
 - > WHERE MOD(comision,100)=0 AND comision!=0;

EMP_NO	APELLIDO	OFICIO	DIRECTOR	FECHA_ALTA	SALARIO	COMISION	DEP_NO
7499	ALONSO	VENDEDOR	7698	1981-02-23	1400.00	400.00	30
7654	MARTIN	VENDEDOR	7698	1981-09-28	1500.00	1600.00	30



3.2- Funciones de Caracteres

- Operan con datos alfanuméricos y el resultado puede ser un dato alfanumérico o un valor numérico.
 - Visualizar los tres primeros caracteres de los apellidos de los empleados seguidos de un punto
 - `mysql> SELECT CONCAT(SUBSTR(apellido,1,3),'.') "INICIALES"`
`-> FROM empleados;`

INICIALES
ALO.
LOP.
MAR.
GAR.
MAR.
REY.
CAL.
GIL.
JIM.



3.2- Funciones de Caracteres

- Visualizar los nombres de los departamentos cuyo nombre tenga más de 6 caracteres reemplazando las letras 'A' por '*'
 - mysql> SELECT REPLACE(dnombre,'A','*')
 - > FROM departamentos
 - > WHERE LENGTH(dnombre)>6;

```
+-----+
| REPLACE(dnombre, 'A', '*') |
+-----+
| CONTABILIDAD                |
| INVESTIGACION               |
| PRODUCCION                  |
+-----+
```



3.3- Funciones de Fechas

- Son funciones que operan con datos de tipo fecha y pueden devolver diferentes tipos de datos. Son las más complejas y que más diversidad presentan. Su manejo es muy importante dentro de los procesos de gestión que se pueden realizar con las bases de datos.
 - Visualizar la fecha que será dentro de una semana
 - `mysql> SELECT CURDATE() "HOY", ADDDATE(CURDATE(),7) "DENTRO DE UNA SEMANA";`

HOY	DENTRO DE UNA SEMANA
2005-04-02	2005-04-09



3.3- Funciones de Fechas

- Visualizar fecha de alta de los empleados con el formato <día de la semana> - <dia> de <mes> de <año>.
- `mysql> SELECT apellido,CONCAT(DAYNAME(fecha_alt),' - ',DAYOFMONTH(fecha_alt),' de ', MONTH(fecha_alt),' de ', YEAR(fecha_alt)) "fecha alta"`
`-> FROM empleados;`

apellido	fecha alta
ALONSO	Monday - 23 de 2 de 1981
LOPEZ	Friday - 8 de 5 de 1981
MARTIN	Monday - 28 de 9 de 1981
GARRIDO	Friday - 1 de 5 de 1981
MARTINEZ	Tuesday - 9 de 6 de 1981
REY	Tuesday - 17 de 11 de 1981
CALVO	Tuesday - 8 de 9 de 1981
GIL	Thursday - 6 de 5 de 1982
JIMENEZ	Thursday - 24 de 3 de 1982



3.3- Funciones de Fechas

- Mostrar los datos de los empleados que entraron en la empresa en lunes
 - mysql> SELECT *
 - > FROM empleados
 - > WHERE DAYOFWEEK(fecha_alt)=2;

EMP_NO	APELLIDO	OFICIO	DIRECTOR	FECHA_ALTA	SALARIO	COMISION	DEP_NO
7499	ALONSO	VENDEDOR	7698	1981-02-23	1400.00	400.00	30
7654	MARTIN	VENDEDOR	7698	1981-09-28	1500.00	1600.00	30



3.3- Funciones de Fechas

- Mostrar para cada empleado su apellido junto con el número de trienios que tiene (se tiene un trienio por cada tres años en la empresa)
 - `mysql> SELECT APELLIDO,
TRUNCATE(((DATEDIFF(CURDATE(),fecha_alt)/365)/3),0)
"TRIENIOS"
-> FROM empleados;`

APELLIDO	TRIENIOS
ALONSO	8
LOPEZ	7
MARTIN	7
GARRIDO	7
MARTINEZ	7
REY	7
CALVO	7
GIL	7
JIMENEZ	7



3.3- Funciones de Fechas

- Mostrar los empleados que llevan más de 23 años en la empresa.
 - mysql> SELECT *
 - > FROM empleados
 - > WHERE fecha_alt<DATE_SUB(CURDATE(),INTERVAL 23 YEAR);

EMP_NO	APELLIDO	OFICIO	DIRECTOR	FECHA_ALTA	SALARIO	COMISION	DEP
7499	ALONSO	VENDEDOR	7698	1981-02-23	1400.00	400.00	30
7521	LOPEZ	EMPLEADO	7782	1981-05-08	1350.50	NULL	10
7654	MARTIN	VENDEDOR	7698	1981-09-28	1500.00	1600.00	30
7698	GARRIDO	DIRECTOR	7839	1981-05-01	3850.12	NULL	30
7782	MARTINEZ	DIRECTOR	7839	1981-06-09	2450.00	NULL	10
7839	REY	PRESIDENTE	NULL	1981-11-17	6000.00	NULL	10
7844	CALVO	VENDEDOR	7698	1981-09-08	1800.00	0.00	30



3.3- Funciones de Fechas

- Visualizar la fecha de 4/10/1997 con el formato <día de la semana>, <número de día> de <nombre del mes> de <año>
 - mysql> SELECT DATE_FORMAT('1997-10-04','%W, %e de %M de %Y') ;

```
DATE_FORMAT('1997-10-04','%W, %e de %M de %Y')
Saturday, 4 de October de 1997
```

- Visualizar la fecha de 2009/05/18 con el formato <número de día> / <numero de mes> / <año>
 - mysql> SELECT DATE_FORMAT('2009-05-18','%e / %c / %Y') "Fecha";

```
+-----+
| Fecha  |
+-----+
| 18/5/2009 |
+-----+
```




3.4- Funciones de Comparación

- Comparan un valor con otro dando y el resultado obtenido dependerá de la función concreta.
 - Visualizar para cada empleado el valor que sea mayor entre su salario y su comisión
 - `mysql> SELECT apellido, salario, comision, GREATEST(salario,comision)`
-> `FROM empleados;`

apellido	salario	comision	GREATEST(salario,comision)
ALONSO	1400.00	400.00	1400.00
LOPEZ	1350.50	NULL	1350.50
MARTIN	1500.00	1600.00	1600.00
GARRIDO	3850.12	NULL	3850.12
MARTINEZ	2450.00	NULL	2450.00
REY	6000.00	NULL	6000.00
CALVO	1800.00	0.00	1800.00
GIL	3350.00	NULL	3350.00
JIMENEZ	1400.00	NULL	1400.00



3.4- Funciones de Comparación

- Mostar los empleados en los que la suma de su salario más su comisión es menor de 2.000 euros
 - mysql> SELECT apellido, salario, comision
-> FROM empleados
-> WHERE salario+IFNULL(comision,0)<2000;

apellido	salario	comision
ALONSO	1400.00	400.00
LOPEZ	1350.50	NULL
CALVO	1800.00	0.00
JIMENEZ	1400.00	NULL



3.5- Otras Funciones

- Son funciones que nos dan información sobre la base de datos o realizan algunas operaciones con valores o listas de valores de las filas de la tabla.
 - Indicar la versión de MySQL que estamos utilizando

- `mysql> SELECT VERSION();`

VERSION()
5.1.0-alpha-nt-max

- Indicar el usuario con el que estamos conectados

- `mysql> SELECT USER();`

USER()
ODBC@localhost



4. Ordenación

- Para obtener la salida de una consulta clasificada por algún criterio o especificación, la sentencia SELECT dispone de la cláusula ORDER BY para ordenar.
 - `SELECT [ALL/DISTINCT] ExpresionColumna [, ExpresionColumna....
FROM NombreTabla
[WHERE CondicionSeleccion]
[ORDER BY {ExpresionColumna / Posicion} [ASC | DESC]
[, {ExpresionColumna / Posicion} [ASC | DESC].....]] ;`
- Donde:
 - **ASC|DESC:** ASC (ascendente) o DESC (descendente) indica la forma de ordenación para esa expresión. Por omisión es ASC.
 - **ExpresionColumna:** conjunto de nombres de columna con literales, operadores y/o funciones. También admite alias.



4. Ordenación

- **Posicion:** si queremos ordenar por expresiones que se muestran en el select la ExpresionColumna puede ser sustituida por el número, **Posicion** corresponde al número de orden que ocupa en la lista de expresiones visualizadas en la select.
- Si existe más de una expresión por la que ordenar estas aparecen separadas por comas y el orden en que se realizan las clasificaciones es de izquierda a derecha, es decir, a igualdad de la expresión más a la izquierda, ordena por la siguiente expresión y así sucesivamente.



4. Ordenación

- **Veamos algunos ejemplos**

- Obtener relación alfabética de todos los empleados con todos sus datos.

- `mysql> SELECT dept_no, apellido, salario`

- > `FROM empleados`

- > `ORDER BY apellido;`

dep_no	apellido	salario
30	ALONSO	1400.00
30	CALVO	1800.00
30	GARRIDO	3850.12
20	GIL	3350.00
20	JIMENEZ	1400.00
10	LOPEZ	1350.50
30	MARTIN	1500.00
10	MARTINEZ	2450.00
10	REY	6000.00



4. Ordenación

- Obtener clasificación alfabética de empleados por departamentos.

- `mysql> SELECT dept_no, apellido, salario`
-> `FROM empleados`
-> `ORDER BY dept_no, apellido;`

dep_no	apellido	salario
10	LOPEZ	1350.50
10	MARTINEZ	2450.00
10	REY	6000.00
20	GIL	3350.00
20	JIMENEZ	1400.00
30	ALONSO	1400.00
30	CALVO	1800.00
30	GARRIDO	3850.12
30	MARTIN	1500.00

- O también podemos escribir:

- `mysql> SELECT dept_no, apellido, salario`
-> `FROM empleados`
-> `ORDER BY 1, 2;`



4. Ordenación

- Obtener los datos de los empleados clasificados por oficios y en orden descendente de salarios..
 - mysql> SELECT emp_no, apellido, oficio, salario
 - > FROM empleados
 - > ORDER BY oficio, salario DESC;

emp_no	apellido	oficio	salario
7876	GIL	ANALISTA	3850.00
7698	GARRIDO	DIRECTOR	3850.12
7782	MARTINEZ	DIRECTOR	2450.00
7900	JIMENEZ	EMPLEADO	1400.00
7521	LOPEZ	EMPLEADO	1350.50
7839	REY	PRESIDENTE	6000.00
7844	CALVO	VENDEDOR	1800.00
7654	MARTIN	VENDEDOR	1500.00
7499	ALONSO	VENDEDOR	1400.00



4. Ordenación

- Obtener los apellidos de los empleados junto con su salario anual (salario + comision en 14 pagas) ordenado de mayor a menor por este salario total.
 - `mysql> SELECT apellido, (salario+IFNULL(comision,0))*14 "Salario anual"`
 - > `FROM empleados`
 - > `ORDER BY (salario+IFNULL(comision,0))*14 DESC;`

apellido	Salario anual
REY	84000.00
GARRIDO	53901.68
GIL	46900.00
MARTIN	43400.00
MARTINEZ	34300.00
ALONSO	25200.00
CALVO	25200.00
JIMENEZ	19600.00
LOPEZ	18907.00



4. Ordenación

- Si no queremos volver a escribir la expresión podemos utilizar el alias o la posición de la expresión por la que queremos ordenar
 - `mysql> SELECT apellido, (salario+IFNULL(comision,0))*14 "Salario anual"`
 - > `FROM empleados`
 - > `ORDER BY "Salario anual" DESC;`
 - NOTA:** no funciona con el alias entre comillas, poned `Salario_anual`
 - `mysql> SELECT apellido, (salario+IFNULL(comision,0))*14 "Salario anual"`
 - > `FROM empleados`
 - > `ORDER BY 2 DESC;`



5. Selección con Limitación de Filas

- Nos va a permitir limitar el número de filas que se visualicen como resultado de una sentencia select.
 - `SELECT [ALL/DISTINCT] ExpresionColumna [,ExpresionColumna....]
FROM NombreTabla
[WHERE CondicionSeleccion]
[ORDER BY {ExpresionColumna/Posicion} [ASC|DESC]
[, {ExpresionColumna/Posicion} [ASC|DESC].....]]
[LIMIT [m,] n] ;`
- Donde:
 - **M**: es el número de fila por el que se comienza la visualización. Las filas se empiezan a numerar por 0. Es opcional y en caso de omitirse se supone el valor 0 (1ª fila)
 - **N**: indica el número de filas que se quieren visualizar.



5. Selección con Limitación de Filas

- **Ejemplos:**

- Obtener los datos de los 5 empleados con menos salario.

- `mysql> SELECT emp_no, apellido, salario, dept_no`

- > `FROM empleados`

- > `ORDER BY salario`

- > `LIMIT 5;`

emp_no	apellido	salario	dept_no
7521	LOPEZ	1350.50	10
7499	ALONSO	1400.00	30
7900	JIMENEZ	1400.00	20
7654	MARTIN	1500.00	30
7844	CALVO	1800.00	30



5. Selección con Limitación de Filas

■ Ejemplos:

- Obtener clasificación alfabética de empleados según su apellido y mostrar desde el 5º hasta el 7º de la lista
 - `mysql> SELECT emp_no, apellido, salario, dept_no`
 - > `FROM empleados`
 - > `ORDER BY apellido`
 - > `LIMIT 4,3;`

emp_no	apellido	salario	dept_no
7900	JIMENEZ	1400.00	20
7521	LOPEZ	1350.50	10
7654	MARTIN	1500.00	30



5. Selección con Limitación de Filas

- **Ejemplos:**

- Si observamos la salida producida al ordenar por apellido comprobamos que se han visualizado 3 filas desde la 5ª (Fila 4 empezando por 0)

- ```
mysql> SELECT emp_no, apellido, salario, dept_no
```

  

```
-> FROM empleados
```

  

```
-> ORDER BY apellido
```

| emp_no | apellido | salario | dept_no |        |
|--------|----------|---------|---------|--------|
| 7499   | ALONSO   | 1400.00 | 30      | Fila 0 |
| 7844   | CALVO    | 1800.00 | 30      | Fila 1 |
| 7698   | GARRIDO  | 3850.12 | 30      | Fila 2 |
| 7876   | GIL      | 3350.00 | 20      | Fila 3 |
| 7900   | JIMENEZ  | 1400.00 | 20      | Fila 4 |
| 7521   | LOPEZ    | 1350.50 | 10      | Fila 5 |
| 7654   | MARTIN   | 1500.00 | 30      | Fila 6 |
| 7782   | MARTINEZ | 2450.00 | 10      | Fila 7 |
| 7839   | REY      | 6000.00 | 10      | Fila 8 |