Contenido

[MYSQL 4](#_Toc480978425)

[Ordenes iniciales. 4](#_Toc480978426)

[De entorno. 4](#_Toc480978427)

[Prompt (significado del prompt): 5](#_Toc480978428)

[Variables del Entorno: 5](#_Toc480978429)

[Instrucciones de uso básico de BBDD: 5](#_Toc480978430)

[De carga de fichero de comandos: 5](#_Toc480978431)

[SQL (Structured Query Language): 5](#_Toc480978432)

[DDL (Data definition language – Lenguaje de definición de datos): 5](#_Toc480978433)

[DML (Data Manipulation Language - Lenguaje de manipulación de datos). 6](#_Toc480978434)

[DCL (Data Control Language – Lenguaje de control de datos). 6](#_Toc480978435)

[TCL (Transaction Control Language – Lenguaje de control de transacciones). 6](#_Toc480978436)

[Objetos en MySQL 7](#_Toc480978437)

[DDL. 7](#_Toc480978438)

[Bases de datos. 7](#_Toc480978439)

[Creación de una base de datos. 7](#_Toc480978440)

[Borrar una base de datos. 7](#_Toc480978441)

[Usuarios. 7](#_Toc480978442)

[Creación de un usuario. 7](#_Toc480978443)

[Borrar un usuario. 7](#_Toc480978444)

[Tablas. 7](#_Toc480978445)

[Creación de una tabla. 7](#_Toc480978446)

[Clausulas. 7](#_Toc480978447)

[Reglas de integridad. 8](#_Toc480978448)

[Integridad Referencial. 8](#_Toc480978449)

[Borrado de tablas: 8](#_Toc480978450)

[Modificación de tablas: 8](#_Toc480978451)

[Cambiar de nombre a tablas: 9](#_Toc480978452)

[Borrar todos los registros de una tabla: 9](#_Toc480978453)

[Vistas. 9](#_Toc480978454)

[Consultar los registros de una tabla o vista (Select). 9](#_Toc480978455)

[Selección simple de una sola tabla. 9](#_Toc480978456)

[Selección simple de más de una tabla. 9](#_Toc480978457)

[Uso de alias en las tablas para hacer más cortas las sentencias. 9](#_Toc480978458)

[Select con condiciones. 10](#_Toc480978459)

[Unión entre tablas con clave ajenas. 10](#_Toc480978460)

[Operadores. 10](#_Toc480978461)

[Selección con ordenación (Order By) 11](#_Toc480978462)

[Notas sobre el “Order by”. 11](#_Toc480978463)

[Clausula “Limit X”. 12](#_Toc480978464)

[Selección con cabecera y uso de alias de campos. 12](#_Toc480978465)

[Operadores de conjunto. 13](#_Toc480978466)

[Funciones. 13](#_Toc480978467)

[Variables de usuario. 15](#_Toc480978468)

[GROUP BY 15](#_Toc480978469)

[HAVING 16](#_Toc480978470)

[DISTINCT 16](#_Toc480978471)

[ALL 17](#_Toc480978472)

[Combinaciones de tablas 17](#_Toc480978473)

[Sentencia anidadas 17](#_Toc480978474)

[EXIST 18](#_Toc480978475)

# MYSQL

Para entrar en mysql lo haremos a través de la consola de mysql o en su directorio de instalación con el siguiente comando

- mysql –help   
 - mysql –u root –p –h localhost (o ip del equipo) [base de datos]

-u: sirve para especificar el usuario con el que nos conectamos al SGBD.   
-p: sirve para que pida el password del usuario.   
-h: sirve para especificar el host donde vamos a realizar conexión.   
[base de datos]: sirve para especificar qué base de datos vamos a utilizar.

## Ordenes iniciales.

En MYSQL no se distingue entre mayúsculas y minúsculas. A continuación vamos a exponer algunas órdenes que nos permite manejar básicamente el entorno y las bases de datos. Inicialmente las clasificaremos en las siguientes categorías.

### De entorno.

“;”: El punto y coma es la señal que indica que un comando ha finalizado, mientras no se escriba estaremos dentro del mismo comando. Se puede escribir comandos de varias líneas que no estarán finalizados has que no pongamos “;”.

“help [comando];” o “? [comando];”: Sirve para mostrar la información/ayuda de los comandos.

“status;”: Muestra las opciones básicas con las que está configurado nuestro motor de bases de datos.

“show status;”: Muestran los valores que tienen las variables del entorno de MYSQL.

“show warnings;”: Muestra una descripción ampliada del ultimo error registrado.

“prompt;” “prompt [cadena];”: Permite cambiar el prompt de la consola de MYSQL, si no se especifica ninguna cadena, se vuelve al prompt por defecto.

“tee fichero;”: Permite guardar en un fichero de texto la salida de la consola de MYSQL, si existe el fichero se añade a lo que tuviera.

“notee;”: Permite cerrar el flujo del comando tee.

“#Comentario;”: Desde la almohadilla hasta el “;” será un comentario. La consola no hará nada.

“/\* Comentario… \*/;”: Comentario de varias líneas, desde el “/\*” hasta el “\*/” será un comentario y la consola no hará nada.

Las dos formas de comentario no tiene ninguna utiliza en el modo comando, pero si se utilizan mucho para documentar los scripts que podamos crear, sobre todo si son muy extensos.

### Prompt (significado del prompt):

Mysql>: Prompt preparado para un nuevo comando.   
->: Esperando la siguiente línea de comando.   
‘>: Se encuentra abierta una cadena que empieza con el apostrofo tilde.   
“>: Se encuentra abierta una cadena que empieza con el apostrofo comillas dobles.   
/\*>: Se encuentra abierta un comentario.

Es frecuente que se origine una sentencia de múltiples líneas que no estaba cerrada hasta que no se termine con el “;”.

### Variables del Entorno:

“select version();”: Nos muestra la versión del motor de BBDD que tenemos funcionando.  
“select current\_Date();”: Nos devuelve la fecha del sistema.   
“select current\_Time();”: Nos devuelve la hora del sistema.   
“select user();”: Nos devuelve el usuario con el que estas conectado.   
“select current\_user();”: Nos devuelve el usuario con el que estas conectado.   
“select now();”: Nos devuelve la fecha y la hora del sistema.   
“show variables;”: Nos devuelve el valor de las variables del entorno.

#### Instrucciones de cálculo:

Permite realizar operaciones matemáticas si se acompañan con el comanto “select”, ejemplo: “select 2+2;” ó “select pi();”.

### Instrucciones de uso básico de BBDD:

“create database [nombre de la base de datos];”: Permite crear una base de datos.   
“show databases;”: Permite listar las BBDD existentes.   
“use [nombre de la base de datos];”: Permite conectarse a una BBDD para trabajar con ella.   
“show tables;”: Permite listar las tablas de la BBDD conectada.   
“select database();”: Nos devuelve la BBDD conectada actualmente.   
“drop database [nombre de la tabla];”: Permite borrar la BBDD.   
“desc [nombre de la tabla];”: Te devuelve información sobre la tabla.

### De carga de fichero de comandos:

“source [nombreDelFichero.txt];”: Permite cargar instrucciones desde un archivo de texto.

# SQL (Structured Query Language):

SQL se basa en el modelo relacional, estará formado por tablas y estas a su vez por columnas o atributos (también llamados campos).

Es un lenguaje estándar de definición, consulta, manipulación y control de BBDD. Se subdivide a su vez en tres (o cuatro según el autor) componentes distintos.

## DDL (Data definition language – Lenguaje de definición de datos):

Este componente está formado por las instrucciones usadas para crear, borrar, modificar y renombrar los componentes de la estructura de la BBDD, es decir, la parte estática. Permiten crear tablas, vista, relaciones, dominios,… además de modificarlas, borrarlas y renombrarlas. Este bloque lo forman las siguientes instrucciones:

- “create tipoDeObjeto nombreDelObjeto [definición];”: Permite crear objetos como por ejemplo vistas.   
 - “alter tipoDeObjeto nombreDelObjeto [definición];”: Permite modificar objetos.  
 - “drop tipoDeObjeto nombreDelObjeto modificiacion;”: Permite borrar objetos.  
 - “rename tipoDeObjeto nombreDelObjeto to nombreNuevo;”: Permite renombrar objetos.

#### Tipos de objetos:

* Database
* View
* Table
* Index
* Sequence
* Domain

## DML (Data Manipulation Language - Lenguaje de manipulación de datos).

Este componente lo forman los comandos utilizados para realizar consultas, modificaciones, borrado e inserciones sobre los datos contenidos en las tablas. Gestiona la parte dinámica de las BBDD, las instrucciones de este componente son las siguientes:

* “select campo/s|\* from tabla/s [opciones];”:
* “insert into tabla [(campo/s…)] values (‘valor’, … | subconsulta);”:
* “delete from tabla [where condición/es];”:
* “update tabla set campo = valor | expresión [where condición/es];”:

## DCL (Data Control Language – Lenguaje de control de datos).

Está formado por los comandos utilizados por el administrador para dar y quitar privilegios, realizar copias de seguridad, estadísticas, etc. En general permite crear y borrar usuarios y la administración de las BBDD. Alguna de las instrucciones de administrador serían las siguientes:

* “grant privilegios on ámbito to usuario/s@host;”:
* “revoke privilegios on ámbito from usuario/s@host;”:
* “show grants;”:

## TCL (Transaction Control Language – Lenguaje de control de transacciones).

Algunos autores en vez de dividir el lenguaje SQL en tres bloques lo dividen en cuatro, incorporando este último. Este bloque solo incorporaría las instrucciones que permiten controlar las transacciones. Las instrucciones que englobaría son las siguientes:

* “savepoint puntoDeRetorno;”:
* “rollback [to puntoDeRetorno];”:
* “commit;”:
* “begin;”:
* “start transaction;”:

## Objetos en MySQL

Los objetos en MySQL serán bases de datos, las tablas, las vistas, índices, campos, dominios, etc. Los nombres que le podremos asignar serán cualquier cadena de caracteres de hasta 30 de longitud. No es sensibles a mayúsculas y minúsculas admitiendo números y caracteres y también el subrayado, pero debiéndose en el primero un carácter.

* “[esquema|propietario] NombreObjeto [.parteObjeto]”

# DDL.

Este bloque de instrucciones solo lo forman cuatro comandos, que son: “create”, “drop”, “alter” y “rename”.

## Bases de datos.

### Creación de una base de datos.

* “create database|eschema NombreBaseDatos”.

### Borrar una base de datos.

* “drop database NombreBaseDatos”

Obviamente al borrar una base de datos desaparecen todas sus tablas con su contenido.

## Usuarios.

### Creación de un usuario.

* “create user NombreUsuario”.

### Borrar un usuario.

* “drop user NombreUsuario”.

## Tablas.

### Creación de una tabla.

* “create table NombreTabla (campo1 tipo, campo2 tipo);”.
* MÁS DETALLE EN LA FOTOCOPIA.

Nota 1: Tras la opción “REFERENCES tabla” en MySQL hay que poner obligatoriamente “(campo/s)”, en ORACLE no es necesario ya que toma por defecto el campo clave de la tabla.

Nota 2: En MySQL se ignoran todos los “CHECK”, pero los soporta por compatibilidad.

### Clausulas.

* TEMPORARY: crea una tabla temporal que al finalizar la sesión dejara de existir.
* ENGINE | TYPE: indica el motor de BBDD que se va a emplear para crear la tabla. Tenemos tres posibilidades;   
   - Memory: Crea una tabla temporal que desaparecerán al acabar la sesión.   
   - MyIsam e InnoDB: Consultar en el anexo.   
  Se pueden poner Engine o Type, por defecto la tabla se crean de tipo MyIsam.
* MIN\_ROWS: Número mínimo de registros que se prevea tener. No es un límite.
* MAX\_ROWS: Número máximo de registros que se prevea tener. No es un límite.
* COMMENT: Es un comentario que se guardara con la definición de la tabla.
* AUTO\_INCREMENT: Sigue para indicar el valor inicial que se empleara en un atributo que se defina de esta manera. Consultar anexo.

### Reglas de integridad.

Las reglas de integridad, llamadas en algunos sitios “reglas de negocio”, son reglas del tipo “este campo es obligatorio” o “para este campo se pueden repetir valores”. Estas reglas de integridad son las reglas referentes a las columnas de las tablas. Y son las siguientes:

* PRIMARY KEY: Se trata de la clave primaria. Lleva implícito el “NOT NULL” y “UNIQUE”.
* [NOT] NULL: El campo sobre el que se aplique no puede tomar valor nulo o sí. La opción por defecto es “NULL”.
* UNIQUE [KEY]: El valor de este campo no puede estar repetido en toda la tabla, pero puede ser nulo.
* CHECK (condición): Condición de validación, se puede utilizar paréntesis más internos y operadores “AND”, “OR”, “NOT”, “=”, “<”, “>”, “<=”, “>=”,…
* DEFAULT: Pone con un valor por defecto el campo en caso de no rellenarlo expresamente.
* ON UPDATE: Con esta propiedad podemos indicar un valor con el que se va a rellenar un campo si se actualizar o modifica ese registro.

### Integridad Referencial.

Para las que las restricciones de integridad se cumplan y funcionen (la inserciones de valores con referencias a otros y los borrados/modificaciones de valores referenciados), ambas tablas, las que referencian y las referenciadas deben estar declaradas de tipo “InnoDB”.

Estar reglas determina las relaciones entre tablas. Se aplican a las claves ajenas. La opción que se determina por defecto en MySQL para borrado y modificación es “NO ACTION” y en ORACLE no existe “ON UPDATE”. La opción “SET DEFAULT” solo puede ser utilizada con tablas “MySam”, la “InnoDB” no lo permite.

El otro formato el que aplica directamente la clausura “REFERENCES” sin poner “FOREIGN KEY” está permitido en MySQL pero no hace nada, solo permite ponerlo por compatibilidad.

### Borrado de tablas:

Drop [Temporaly] table [if exist] nombretabla [, nombretabla2,…];

### Modificación de tablas:

Una vez creada a tabla, podrá ser modificada su estructura sin tener que borrarla y volver a crearla, y sobretodo sin tener que volver a introducir los registros que ya hubiera.

Se podrá: añadir y borrar columnas, modificar el tipo de datos o la precisión de una columna, añadir y borrar restricciones, añadir/activar y desactivar/borrar restricciones de integridad y también cambiarles el nombre.

alter table nombretabla add [(] columna/s tipe/s [after columna] [restricciones] [first] [)];   
alter table nombretabla add Primary Key (Columna/s);   
alter table nombretabla add Unique (Columna/s);   
alter table nombretabla add Foreign Key (Columna/s) references tabla (Columna/s) [ON DELETE…] [ON UPDATE…];   
alter table nombretabla drop [(]Columna/s[)];   
alter table nombretabla drop Primary Key;   
alter table nombretabla drop index [(]Columna/s[)]; /\*Para borrar Unique Key\*/   
alter table nombretabla drop Foreign Key (Columna/s);   
alter table nombretabla modify [(]Columna/s [name] tipo [restricción][)];   
alter table nombretabla change NombreColumnaVieja to NombreColumnaNueva tipo [restricción];  
alter table nombretabla rename to NombreColumnaNueva;

### Cambiar de nombre a tablas:

Rename table nombreviejo to nombrenuevo;

### Borrar todos los registros de una tabla:

Truncate table nombretabla;

## Vistas.

Como objeto que es una vista debería estar aquí en el DDL, pero como es necesario conocer en profundidad el formato de las sentencias “select” y las subconsultas para poder crearlas, se estudiara al final del DML.

## Consultar los registros de una tabla o vista (Select).

Se pueden consultar tablas enteras, columnas sueltas, combinaciones de estas, etc.

Existen muchísimas combinaciones de la orden “select”, así que las veremos poco a poco hasta completar todas sus posibilidades.

En todos los puntos donde se hagan referencia a una tabla igualmente se podría hacer referencia a una lista.

### Selección simple de una sola tabla.

* Select campo1,campo2,campo3,… from tabla;
* Select \* from tabla;

Selecciona y muestra o todos los campos, o solo los indicados de la tabla especificada.

### Selección simple de más de una tabla.

Se han de especificar los campos que se desean que aparezcan en el dictado, debiendo además indicar la tabla a la que pertenecen. Todas las tablas que se utilicen en el “select” deben aparecer tras el “from” y también se deben visualizar cualquier campo de las tablas especificadas.

* Select [tabla1.]campo1, [tabla1.]campo2,… [tablaN.]campo/s from tabla1, tabla2,… tablaN;
* Select [tabla1.]\*, [tabla2.]campo1,… from tabla1, tabla2,… tablaN;

NOTA IMPORTANTE: Al hacer un “select” con más de una tabla, el resultado es la combinación de todos los registros de cada una de las tablas, es decir, si hacemos un “select” sin condiciones de dos tablas con tres y cinco registros respectivamente, nos saldrá las quince combinaciones posibles, o lo que es lo mismo, el producto cartesiano, y si combinamos las tablas de 3, 4 y 5 registros nos saldrán 60 combinaciones y así sucesivamente.

### Uso de alias en las tablas para hacer más cortas las sentencias.

Para especificar una tabla en el “from” se le puede añadir un alias que se puede usar en el resto de las sentencias. Hay que tener en cuenta que los alias solo existen en la sentencia donde se especifican, una vez acabada la sentencia el alias deja de existir. Más adelante también veremos cómo crear alias de campos, no solo de tablas.

Se pueden utilizar alias también en sentencias “update” y “delete” para hacer referencias en una subconsulta a un campo de la consulta padre. Si se utilizan se deberá tener en cuenta que en las subconsultas no puede aparecer la tabla de la que se pretende borrar o actualizar registros.

1. Cambia la importancia a “Alta” a toda la colección de Manuel Cosano.  
    - update coleccion set importancia = 'alta' where dnipropietario=(select dni from persona where nombre = 'Manuel R. Cosano');
2. Cambia la importancia a “Alta” a toda la colección de Manuel Cosano si su valor es mayor al estimado.

### Select con condiciones.

Para poner una condición a la sentencia select se hace añadiendo, tras las tablas del from, la clausula where. Puede utilizarse AND, OR, NOT, (), etc.

Select [tabla1.]campo1, [tabla1.]campo2,… [tablaN.]campoN from tabla1, [tabla2] where condición1[AND|OR condicion2[…]];

1. Mostar los datos de las colecciones cuyo valor estimado supere 1000€.  
   select \* from coleccion where valor\_estimado>1000;

### Unión entre tablas con clave ajenas.

Todas las tablas relacionadas deben estar unidas lógicamente por las claves ajenas y esto se hará con condiciones de igualdad. En la que se igualara la clave ajena normalmente a la clave de la otra.

select \* from persona, coleccion where persona.dni = coleccion.dnipropietario;

### Operadores.

Según una primer clasificación tenemos dos tipos de operadores; unarios y binarios, que serán aquellos operadores que tienen uno o dos operandos respectivamente. Existe otra clasificación por tipos que es la que vamos a ver a continuación.

* Operadores de comparación y lógicos: “=”, “!=” ó “<>”, “<”, “>”, “<=”, “>=”, NOT, AND, OR.
* Aritméticos: Se pueden combinar como se deseen y tendrán la misma precedencia que las expresiones en programación. Para variar esta precedencia se pueden utilizar los paréntesis “(“, “)”; “+”, “-“, “\*”, “/”, “%”, “(“, “)”.
* Operadores de caracteres: NombreCampo [NOT] like [Binary] ‘Patrón’  
   - Like: Sirve para seleccionar cadenas de caracteres parecidas a un patrón. Tiene los comodines “%” y “\_” que representa lo mismo que “\*” y “?” respectivamente en el MSDOS, el “%” equivale a 0 o varios caracteres faltantes y el “\_” equivale a un solo carácter faltante.   
  Para forzar la comparación “case sensitive” habría que añadir la causula “binary”.
* Operadores de grupo:   
   - NombreCampo [NOT] IN (listavalores1|subconsulta);   
   - NombreCampo [NOT] BETWEEN Inicio AND fin;   
   - NombreCampo IS [NOT] NULL;   
   - IN: Comprueba si el valor del campo es o no igual a alguno de los valores de la lista o de la subconsulta.   
   - BETWEEN: Comprueba si el valor del campo esta o no entre los valores indicados, incluido los extremos. Tambien funciona con cadena de caracteres.  
   - IS NULL: Comprueba si el valor del campo es o no el valor nulo.

El valor NULL puede resultar desconcertante hasta que se comience a utilizar. Conceptualmente NULL significa valor inexistente o desconocido y es tratado de manera diferente a otros valores. Para verificar si un valor es NULL no se puede emplear los operadores de comparación como el “>”,”<” o “=”.

Claramente no se obtiene valores significativos con las comparaciones aritméticas, si usamos en su lugar los operadores IS NULL ó IS NOT NULL obtenemos otro resultado.  
Los valores NULL son considerado iguales por las clausulas GROUP BY y ORDER BY. Si realizamos una ordenación con ORDER BY los valores NULL se presentan al principio en la ordenación ascendente y al final en la descendente. Un error muy común que se presenta cuando trabajamos con valores NULL es asumir que no se puede insertar un valor 0 o una cadena vacía en una columna definida como NOT NULL, pero no es asi. Tanto el valor 0 como la cadena vacía son valores concretos, mientras que NULL es la inexistencia de valor, por lo que se puede insertar cadenas vacías o “0” en columnas NOT NULL.

## Selección con ordenación (Order By)

* [order by [binary] campo|numero [asc|desc] [,…]];

Todos los resultados obtenidos hasta ahora con la orden “Select” se pueden ordenar por los valores de cualquiera de los campos de las tablas usadas en él, incluso aunque no aparezca en el resultado. Para hacer la ordenación se deben añadir al final del todo la cláusula “order by” y el campo o campos de ordenación permitiendo incluso que sea una expresión o función ya sea alfanumérica o aritmética.

### Notas sobre el “Order by”.

1. La ordenación puede ser incluso con un campo que no apareza en el “Select” pero que si debe de estar en algunas de las tablas que aparecen en el “from”.
2. También podemos indicar el número del campo o la expresión por el que deseamos ordenar, siendo el número del campo la posición que ocupa en la lista “Select” o la expresión que debe aparecer en la lista del “Select”.
3. Se puede ordenar por más de una columna tanto de forma ascendente “ASC” que es la acción por defecto como de forma descendente “DESC” o incluso combinadas. Para ordenar por más de un campo se separaran por comas junto con la opción correspondiente (ASC|DESC).En este caso ordenara por el primer campo, en caso de igualdad de valor por el segundo y así sucesivamente.
4. La cláusula “binary” hará una ordenación distinguiendo mayúsculas y minúsculas.
5. Si indicamos una ordenación para un campo enumerado (enum) su contenido será ordenado según el orden en que han sido definidos sus valores.

### Clausula “Limit X”.

Esta clausura opcional se puede poner al final del todo de la sentencia “select” y limitara la salida del select al número indicado con la variable “X”. También se puede indicar un “desplazamiento” que indicaría al “select” a partir de que resultado (del siguiente) debe mostrar limitando el resultado del número anterior. Si no se pone desplazamiento saldrán los primeros registros indicados.

* Select campo/s from tabla/s …. limit [desplazamiento,] NúmeroDeFilas;

### Selección con cabecera y uso de alias de campos.

Se pueden hacer “select” cambiando la cabecera del listado, haciendo que en vez de aparecer arriba el nombre del campo aparezca el mensaje que nosotros deseemos.

Si algunos de los campos no se le pone cabecera aparecerá con su nombre de campo.

También se pueden poner alias de columnas y no solo de tablas, pero estos alias solo funcionaran en la clausulas “order by” y “having” (que se verá más adelante) y en las subconsultas interiores desde una consulta padre pero también solo en “order by” y “having”. También la cadena que se ponga como cabecera será o podrá ser utilizada como alias siempre y cuando esa cadena no contenga espacios o caracteres reservados como “,” o “.”.

* Select nombre [as] “Empleado, SBase+Complementos [as] “Sueldo”, Direccion from empleados ….….…;

SELECT Nombre [AS] “Empleado”, SBase + Complementos [AS] “Sueldo”, Dirección FROM empleado …;  
SELECT dnipropietario “DNI”, descripción, importancia “valor”, valor\_estimado “Precio” FROM colección;  
SELECT dnipropietario “DNI”, descripción, importancia “valor”, valor\_estimado “Precio” FROM colección ORDER BY Precio;   
SELECT nombre, salario, comisión, salario+comisión, (salario+comisión)\*12 FROM empleados;  
SELECT nombre “Empleado”, salario+comisión “Total Mes”, (Salario+comisión)\*12 “Total Año” FROM empleados;   
SELECT nombre “Empleado”, salario+comisión “Total Mes”, (Salario+comisión)\*12 “Total Año” FROM empleados ORDER BY salario+comisión DESC;   
SELECT nombre “Empleado”, salario+comisión “Mensual”, (Salario+comisión)\*12 “Anual” FROM empleados ORDER BY Mensual DESC;   
SELECT concat(nombre, “ (“, Trabajo, “) “) “Empleado”, salario+comisión “Mensual” FROM empleados ORDER BY Mensual DESC;   
SELECT concat(nombre, “ (“, Trabajo, “) “) “Empleado”, salario+comisión “Mensual” FROM empleados ORDER BY Empleado DESC;

1 - De los ejemplares mostrar, estado, sexo, nombre zona donde han sido capturados, y nombre de la persona que lo capturó si es la misma persona que lo posee.

2 - Mostrar el nombre de todos aquellos cazadores junto con el código de ejemplar, sexo, nombre de la zona y superficie de la zona donde ha sido capturado.

3 - Mostrar código, nombre de la especie y descripción de la especie de aquellos ejemplares capturados en zonas de clima seco.

4 - Mostrar código, nombre de la especie y descripción de la especie de aquellos ejemplares capturados en zonas que no tenga zona padre.

### Operadores de conjunto.

* “UNION”: devuelve la combinación de todos los resultados de dos o más “select” distintos, dando la apariencia de un solo resultado pero sin incluir registros duplicados.  
  NOTA IMPORTANTE: El número de columnas indicadas para mostrar en los “select” que aparezcan deben de ser el mismo, sino la sentencia da un error.  
    
  “SELECT …. UNION SELECT ……;”
* “UNION [ALL]”: devuelve lo mismo que “UNION” pero incluye los duplicados.

### Funciones.

Serán operadores que se aplican sobre uno o varios valores y/o atributos devolviendo un solo valor. Todos ellos tendrán su nombre e irán seguidos por unos paréntesis en donde se especifican sus parámetros.

Ejemplos:  
 - “select sysdate();”: devuelve la fecha del sistema.   
 - “abs (-3.4);”: valor absoluto.   
 - “sqrt(valor);”: devuelve raíz cuadrada.   
 - “round(valor);”: redondea el valor.

#### Funciones numéricas y matemáticas.

Son funciones que devuelven un valor numérico.

* ABS(n): Valor absoluto de n.
* CEIL(n): Da el entero inmediato y superior del valor n.
* FLOOR(n): Da el entero inmediato e inferior del valor n.
* MOD(m, n): Da el resto de la división de m entre n.
* POWER(m, n): Potencia.
* SQRT(n): Raíz cuadrada de n.
* ROUND(n[, m]): Redondeo de n a m número decimales, si no se especifica m se redondea a entero.
* TRUNCATE(n, m): Recorta al número m de decimales especificados.
* COS(n): Coseno debiendo estar n en radianes.
* SIN(n): Seno debiendo estar n en radianes.

#### Funciones de caracteres

Serán funciones que devuelven cadenas de caracteres.

En todas ellas se pueden utilizar indistintamente campos de tipo carácter o texto entre comillas.

* CONCAT(Cadena1, Cadena2[, Cadena3, …]): Devuelve las cadenas concatenadas.
* GROUP\_CONCAT([istinct] campos): Concatena los valores de los campos separados por comas.
* LOWER(cadena): Convierte la cadena en minúsculas.
* UPPER(cadena): Convierte la cadena en mayúsculas.
* REPLACE(cadena, CBus, CRem): Reemplaza en la cadena, la cadena buscada por la cadena de reemplazo.
* SUBSTR(cadena,m[n]): Devuelve de la cadena la subcadena que empieza en la posición m y de longitud n, si no se especifica n seria hasta el final. Se pueden utilizar números negativos para el valor de n, lo que indicaría la posición del final.
* LENTGH(cadena): Devuelve la longitud de la cadena, es la única función de cadenas y caracteres que devuelve un numero.
* LTRIM(Cadena): Borra los caracteres en blanco a la izquierda de la cadena hasta encontrar el primer carácter.
* RTRIM(cadena): Borra los caracteres en banco a la derecha de la cadena hasta encontrar el primer carácter.
* TRIM(Cadena): Borra los caracteres en blanco a la izquierda y derecha de la cadena hasta encontrar los primeros caracteres.
* RIGHT(Cadena, n): Devuelve los n caracteres desde la derecha.
* LEFT(Cadena, n): Devuelve los n caracteres desde la izquierda.

#### Funciones de fecha y hora.

Operan y devuelve valores de tipo fecha, estos valores deben tener el siguiente formato:

* “AA-MM-DD”
* “AA/MM/DD”
* “AAMMDD”
* AAMMDD
* “AAAA-MM-DD”
* “AAAA/MM/DD”
* “AAAAMMDD”
* AAAAMMDD
* “HH:MM:SS”
* “HHMMSS”
* HHMMSS

Nota: Se puede usar indistintamente comillas simples o dobles.

* YEAR(fecha|sysdate()): Devuelve el año de la fecha indicada.
* MONTH(fecha|sysdate()): Devuelve el mes de la fecha indicada.
* DAY(fecha|sysdate()): Devuelve el día de la fecha indicada.
* HOUR(hora|sysdate()): Devuelve la hora de la fecha indicada.
* MINUTE(hora|sysdate()): Devuleve el minuto de la fecha indicada.
* SECOND(hora|sysdate()): Devuelve el segundo de la fecha indicada.
* LAST\_DAY(fecha|sysdate()): Devuelve la fecha del último mes de la fecha indicada.
* SYSDATE(): Devuelve la fecha y hora del sistema.
* CURDATE(): Devuelve solo la fecha del sistema.
* CURTIME(): Devuelve solo la hora del sistema.

Nota: A las variables de tipo fecha se le puede sumar y restar números, incluso sumarse y restarse entre sí pero teniendo en cuenta lo siguiente:   
Primero: Si se restan dos fechas, el resultado será igual a un número que indica el número de días entre esas dos fechas.   
Segunda: Si una fecha se le resta un número “n” el resultado es esa fecha incrementada o disminuida en “n” días.

#### Otras funciones.

* COALESCE(valor1, valor2[, valor3…]): Esta función devuelve el primer valor no nulo de la lista.
* GEATEST(valor1, valor2[, valor3…]): Esta función devuelve el mayor valor de la lista. Los valores pueden ser de cualquier tipo.
* LEAST(valor1, valor2[, valor3…]): Esta función devuelve el menor valor de la lista. Los valores pueden ser de cualquier tipo.

### Variables de usuario.

Se pueden emplear variables de usuario en MySQL para guardar resultados sin tener necesidad de apuntarlos. Para esto utilizaremos la “@” seguida del nombre de la variable y la asignación de su valor con “:=”(dos puntos más el símbolo igual).

1. Mostrar código, estado, sexo y nombre del propietario de aquellos ejemplares de estado bueno o excelente y que hayan sido capturados antes del 2016.  
    - select ej.codigo, ej.estado, ej.sexo, p.nombre from ejemplar as ej, persona as p where p.dni=ej.dnipropietario and ej.estado in ("bueno","excelente") and year(fcaptura)<2016;
2. Mostrar los datos de los ejemplares macho capturados en alguna zona de superficie superior a la media.   
    - select ej.\* from ejemplar as ej, zona as zo where zo.nombre=nomzona and sexo="macho" and zo.superficie>(select Avg(superficie) from zona);

### GROUP BY

Se puede añadir la cláusula “group by” después de las condiciones del “where”. Esta cláusula permite agrupar el resultado que debería salir con el “Select” en grupos con valores comunes en el campo o campos especificados con “Group by”.   
Los campos que aparezcan en la cláusula “group by” no tienen que aparecer obligatoriamente en el resto de la sentencia.   
Podemos hacer el “group by” con números, indicando en este caso con que campo del “select” debemos agrupar. Esta opción solo es posible si el criterio de agrupación está en la lista del select.

Las funciones que se suelen utilizar con esta cláusula devuelven un resultado en el grupo de registros con alguna característica común. Por ejemplo: se pueden contar cuantos ejemplares de cada especie hay.

Se puede agrupar por más de un campo. Los campos que aparezcan en el select no pueden contener datos diferentes para cada agrupamiento, es decir, que por ejemplo no se pueden poner en el select el nombre de la especie si agrupo por el código. Los campos que aparezcan en el select sin función deben tener los mismos valores para todos los registros del grupo, si no se respetan estas indicaciones no da error pero los resultados serán sin sentido cogiendo un valor cualquiera de los que cambian dentro del grupo.

#### Otras funciones:

Estas funciones se llaman también funciones de grupo o funciones colectivas:  
 **- AVG([DISTINCT|ALL]n**): Devuelve la media de “n”.  
 **- COUNT([DISTINCT|ALL]n):** Cuenta el número de filas de la consulta.  
 **- MAX(~~[DISTINCT|ALL]~~n):** Devuelve el valor máximo de “n”.  
 **- MIN(~~[DISTINCT|ALL]~~n):** Devuelve el valor mínimo de “n”.  
 - **SUM([DISTINCT|ALL]n):** Devuelve la suma de los valores de “n”.

Estas funciones también se puede usar fuera del “group by” y todas ignoran los nulos, incluido “COUNT”.

Las clausulas ALL y DISTINCT permiten especificar a la función respectivamente que considere todos los valores de la serie o solo los que son distintos. Por defecto la función considerada es ALL.

No tiene sentido utilizar las clausulas ALL y DISTINCT con el máximo y el mínimo, están bien, no dan error, pero no tiene ningún sentido ya que el resultado siempre será el mismo.

DISTINCT: Hace que la función considere solo los valores que sean distintos de la serie, no utilizando los duplicados quedándose con solo uno de ellos. También pueden usarse fuera de los agrupamientos.

ALL: Considerara todos los valores de la serie incluyendo los duplicados. Es la opción por defecto y también se puede usar fuera de los agrupamientos.

# HAVING

Es una cláusula que complementa a “GROUP BY” y se puede usar para poner condiciones a los grupos que se visualizan con el “GROUP BY”, no a los registros. El orden de procesamiento cuando utilizamos la cláusula “HAVING” es el siguiente:  
 - Primero: Se filtra con “where” los registros que cumplan las condiciones especificadas.  
 - Segundo: Se agrupan con “group by”.  
 - Tercero: Después de hacer los grupos se pueden poner condiciones a estos para eliminar algunos grupos.

Nota importante: cuando el “having” aparece en una condición con uno o varios campos, estos deben ser campos con los que se han agrupado o campos en los que para la agrupación sus valores no difieren en todo el grupo. Si no se hace esto no dará error pero dará resultados sin sentido.

# DISTINCT

Como hemos comentado se puede poner esta cláusula delante de los campos listados “select”, si se pone delante de un campo mostrara solo aquellos registros cuyo valor para ese campo sea distinto. Muestra solo uno de aquellos registros para cuyos campos seleccionados coincidan totalmente.

# ALL

También se puede poner delante de cada uno de los campos listados en el “select”, pero es la opción por defecto y no es necesaria ponerla. Si se pone expresamente o no se pone nada mostrara todos los registros incluidos los duplicados.

# Combinaciones de tablas

Por defecto se hace una combinación de todos los registros de las tablas implicadas en un “select” (Producto cartesiano) si no hay una condición que limite eso. Esto ya se explicó anteriormente.

En el caso de que aparezca más de una tabla en el “select” será necesario limitar las combinaciones de registros que son varias, y serán válidas solo aquellas en las que la clave ajena de una coincide con la clave de la otra con la que se relaciona. El formato general será el siguiente:  
 select campo/s from tablas where condicionEnlace, …., …;

**Nota importante:** Cuando combinamos más de una tabla solo saldrán los registros que tienen su correspondiente en todas ellas según la condición de enlace, no apareciendo la que no lo tenga.

# Sentencia anidadas

Es una sentencia “select” que se encuentra dentro de otra sentencia “select” llamada sentencia padre. La sentencia padre utiliza las filas devueltas con la subconsulta.

En las sentencia “insert” “update” “delete” no puede aparecer en la subconsulta la misma tabla en la que se pretende insertar, modificar o borrar.

Las subconsultas puede aparecer en cualquier orden, siempre que la construcción sea correcta, pudiendo incluso aparecer en la lista de campos del “select”.

Las subconsultas pueden devolver uno o varios valores en una o varias filas y hay que tener cuidado con esto, hay que tener en cuenta lo siguiente:  
 - Las órdenes que incluyan una subconsulta con el operador “in” esa subconsulta puede devolver un solo valor pero con varias filas, que serán las utilizadas en la sentencia padre como si fueran las listas de valores del “in”.  
 - En otros casos de sentencias con operadores del tipo <,>,<=,>=,… la subconsulta solo deberá devolver un valor y una fila.  
 - En una subconsulta a continuación con la clausula “values” de la sentencia “insert” puede llevar varios campos y varios registros, considerando que cada una de las filas de los resultados será un registro que se insertara con la sentencia “insert”. Es decir, se insertaran tantos registros como sentencias devuelva la subconsulta y cada uno de ellos tendrá tantos campos como los que haya en cada fila.

La subconsulta se ejecutara y dará un resultado que será utilizado por la sentencia padre.

Una subconsulta puede tener a su vez otra subconsulta y no hay límite en las anidaciones.

# EXIST

Si la subconsulta devuelve algún registro se evalúa como verdadera y la condición “where” se cumple. Da igual el número de campos y registros que devuelva la subconsulta, solo interesa saber si devuelven algo o no.

# Vistas

Como objeto que es una vista debería estar incluida en el DDL, pero como es necesario conocer en profundidad la sentencia “select” y las subconsultas para poder crearlas la hemos ubicado aquí.

## Creación de vistas.

Una vista es una tabla lógica basada en una o varias tablas. La tabla no se crea físicamente, sino lo que se almacena es la sentencia “select” que define la vista.

A la hora de utilizar una vista se usa igual que una tabla cualquiera pudiendo usar todas las cláusulas que permite la sentencia “select”. Si empleamos una cláusula sobre la vista esta se ejecutara como si se añadiera sobre las que ya estuvieran definida la vista.

También es posible definir vistas sobre otras vistas, tablas o combinaciones de tablas.

Las vistas son utilizadas sobre todo para:

* Prohibir el acceso a datos confidenciales. Ya que se crearan vistas que no incluyan columnas o tablas de las que no se quieran revelar información y dando al usuario solo acceso a las vistas y no a las tablas.
* Reducir la formulación de consultas complejas o utilizadas con mucha frecuencia. De esta forma reducimos considerablemente consultas muy largas o muy utilizadas.
* Independizar los programas de los datos al utilizarlas. Si por ejemplo utilizamos vistas en los programas y modificamos las tablas en las que se basas, por ejemplo cambiando un nombre a un campo, no habría que reescribir el programa que la usen, bastaría con modificar la vista.

Formato general para crear una vista:

Create [or replace] view NombreVista [Columna1][columna2][columna3][..] as Subconsulta [with check option];

* Or replace: Vuelve a crear la vista si ya existe para no tener que borrarla y tener que crear nuevamente.
* With check option: Si la vista permite inserciones, comprobara que los datos que se introduzca a través de la vista cumplas las condiciones del “where” que puede haber en las subconsulta. No solo no comprueba las condiciones al mostrar los datos sino también al introducirlos.

El poner nuevos nombres a las columnas es opcional, pero se hace obligatorio cuando las tablas seleccionadas en las subconsultas existan nombres de campos repetidos.

## Vista de actualización.

En principio las vistas no permiten actualización, es decir, no se permite que se ejecutan instrucciones “insert into”, “update” y “delete” sobre ellas. Si se hicieran, las modificaciones se deben reflejar en las tablas originales sobre las que están definidas, cosa que no siempre es posible. Como norma general y por el momento una vista permitirá actualizaciones si cumple todo lo siguiente:

* Está definida sobre una única tabla.
* No puede aparecer ningún “distinct” en los campos del “select”.
* No puede tener un agrupamiento.
* No contiene funciones de grupo ni colectivas.
* No puede aparecer en una subconsulta una tabla que contenga la consulta padre.

Si una vista es actualizable y contiene expresiones, esas columnas no serán actualizables.

Si realizamos inserciones a través de una vista actualizable estas se realizaran sobre la tabla real correspondiente y las restricciones que tienen.

Todas las listas de actualización son a la vez de consulta.

**Borrar vistas:** Si borramos una vista sobre la que están definidas otras vistas, estas segundas se borraran en cascada.

**Ver las vistas existentes:**