

INDICE

1. Sentencia SELECT	1
1.1. Sintaxis completa	1
1.2. Consultas Básicas	2
1.3. Filtros.....	4
1.3.1. Expresiones para filtros.....	4
1.3.1.1. Construcción de filtros	6
1.3.1.2.Filtros con operador de pertenencia a conjuntos (IN)	6
1.3.1.3.Filtros con operador de rango	7
1.3.1.4.Filtros con test de valor nulo.....	7
1.3.1.5.Filtros con test de patrón	7
1.3.1.6.Filtros con límite de número de registros	8
1.4. Ordenación	8
1.5. Consultas de resumen	8
1.5.1. Filtros de Grupos.....	9
1.6. Subconsultas.....	10
1.6.1. Test de comparación.....	10
1.6.2. Test de pertenencia a conjunto	10
1.6.3. Test de existencia	10
1.6.4. Test cuantificados ALL y ANY	10
1.6.5. Subconsultas anidadas.....	11
1.7. Consultas Multitabla	11
1.7.1. Consultas Multitabla I	12
1.7.2. Consultas Multitabla II	12
1.8. Consultas Reflexivas.....	13

1.Sentencia SELECT

Tutorial: <https://dev.mysql.com/doc/refman/8.0/en/select.html>

DQL es la abreviatura del **Data Query Language** (Lenguaje de Consulta de Datos) de SQL. El único comando que pertenece a este lenguaje es el versátil comando **SELECT**. Este comando permite:

- Obtener datos de ciertas columnas de una tabla (**proyección**).
- Obtener registros (filas) de una tabla de acuerdo con ciertos criterios (**selección**).
- Obtener datos de tablas diferentes (**asociación, join**).
- Realizar cálculos sobre los datos (**funciones de agregado**).
- Agrupar datos (**group by**).

Para ver el comando completo visita el siguiente enlace: <https://dev.mysql.com/doc/refman/8.0/en/select.html>

1.1. Sintaxis completa

```
SELECT [DISTINCT] select_expr [,select_expr]
[FROM tablas]
[WHERE condiciones]
[GROUP BY atributos]
[HAVING predicado de grupo]
{[UNION\UNION ALL \INTERSECT\MINUS] seleccion}
[ORDER BY {nombre_columna | expression | posicion} [ASC | DESC],...];
```

1.2. Consultas Básicas

SELECT [DISTINCT] select_expr, [select_expr]

FROM tablas

- **DISTINCT**, parámetro opcional que se utiliza para devolver sólo valores distintos, o, dicho de otro modo, que suprima las repeticiones.
- **select_expr**, columnas que se muestran.
 - nombre_columna [AS alias]
 - *,
 - Expresión

EJEMPLOS: Ejecutar el script 'concesionario.sql'.

CONSULTA 1: Consulta que muestre todos los campos (matricula, marca, modelo) de la tabla vehículos.

3 • `SELECT * FROM VEHICULOS;`
4

Result Grid

	matricula	modelo	marca
▶	1129FGT	Ibiza GT	Seat
	1132GHT	Leon TDI 105 CV	Seat
	3447BYD	A3 TDI 130 CV	Audi
	7423FZY	Coupe	Hyundai
	M6836YX	Corolla g6	Toyota

5 • `SELECT matricula, marca, modelo`
6 `FROM VEHICULOS;`
7

Result Grid

	matricula	marca	modelo
▶	1129FGT	Seat	Ibiza GT
	1132GHT	Seat	Leon TDI 105 CV
	3447BYD	Audi	A3 TDI 130 CV
	7423FZY	Hyundai	Coupe
	M6836YX	Toyota	Corolla g6

CONSULTA 2: Consulta que muestre la matricula y modelo-marca de la tabla vehículos.

7 • `SELECT matricula, concat (marca, ' ', modelo) as coche`
8 `FROM VEHICULOS;`

Result Grid

	matricula	coche
▶	1129FGT	Seat Ibiza GT
	1132GHT	Seat Leon TDI 105 CV
	3447BYD	Audi A3 TDI 130 CV
	7423FZY	Hyundai Coupe
	M6836YX	Toyota Corolla g6

CONSULTA 3: Consulta que muestre la matricula, modelo, resultado de la expresión (1+5) y fecha actual en formato (dd/mm/aaaa) de la tabla vehículos.

```

11 • SELECT matricula, modelo, 1+5, date_format(curdate(), '%d/%m/%Y') as fecha
12 FROM VEHICULOS;
13

```

matricula	modelo	1+5	fecha
1129FGT	Ibiza GT	6	02/02/2016
1132GHT	Leon TDI 105 CV	6	02/02/2016
3447BYD	A3 TDI 130 CV	6	02/02/2016
7423FZY	Coupe	6	02/02/2016
M6836YX	Corolla g6	6	02/02/2016

CONSULTA 4: Consulta que muestre la marca de cada coche de la tabla vehículos.

```

16 • SELECT marca
17 FROM VEHICULOS;

```

marca
Seat
Seat
Audi
Hyundai
Toyota

CONSULTA 5: Consulta que muestre las distintas marcas de la tabla vehículos.


```

19 • SELECT DISTINCT marca
20 FROM VEHICULOS;
21
22

```

marca
Seat
Audi
Hyundai
Toyota

CONSULTA 6: Consulta que muestre cuantas marcas distintas hay en la tabla vehículos.

16	•	SELECT count(marca) as numMarcas
17		FROM VEHICULOS;
18		
		
Result Grid		
	numMarcas	
	5	

19	•	SELECT count(DISTINCT marca) as numMarcas
20		FROM VEHICULOS;
Result Grid		
	numMarcas	
	4	

1.3. Filtros

Los **filtros** son condiciones que cualquier SGBD interpreta para seleccionar registros y mostrarlos como resultado de la consulta. Es una expresión que indica la/s condición/es que deben satisfacer los registros.

SELECT [DISTINCT] select_expr [,select_expr]
[FROM tablas]
[WHERE filtro] 

CONSULTA 7: Consulta que muestre todos los campos de los vehículos de la marca Seat.

```
38 • SELECT * FROM VEHICULOS WHERE MARCA = 'SEAT';
39 • SELECT * FROM VEHICULOS WHERE MARCA LIKE 'SEAT';
40
```

Result Grid

Filter Rows:

Edit:

Exp

matricula	modelo	marca
1129FGT	Ibiza GT	Seat
1132GHT	Leon TDI 105 CV	Seat

1.3.1. Expresiones para filtros

Los filtros se construyen mediante expresiones. Una **expresión** es una combinación de operadores, operandos y funciones que producen un resultado.

43	•	SELECT (2+3) > (6*2);
44		
Result Grid		
	(2+3) > (6*2)	
	0	

CONSULTA 8: Mostrar la fecha de hoy -31 años

47	•	SELECT DATE_SUB(CURDATE(), INTERVAL 31 YEAR) AS fecha;
48		
Result Grid		
	fecha	
	1985-01-04	

Elementos que pueden formar parte de las expresiones:

- **Operandos:** Los operandos pueden ser **constantes**, 3, 2.3, 'España', '2012,01,03' o **variables** como edad.
- **Operadores Aritméticos:** +, -, *, /, %;
- **Operadores Relacionales,** sirven para comparar dos o más operandos.

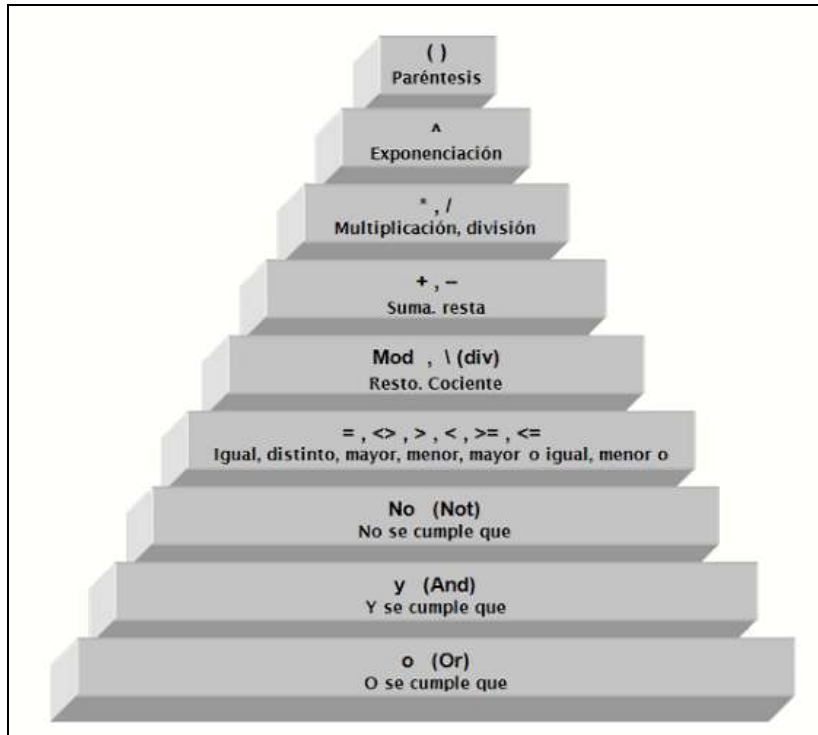
Operador	Significado
=	Igual
>	Mayor
<	Menor
>=	Mayor o igual
<=	Menor o igual
!=	Distinto
<>	Distinto
BETWEEN	Comparación con rango

- **Operadores lógicos.** Los operadores lógicos toman como operandos valores lógicos, es decir, cierto o falso, en SQL 1 o 0.

Operador	Significado
AND	Y
OR	Ó
NOT	NO

x	y	x AND y	x OR y	NOT x
TRUE	TRUE	TRUE	TRUE	FALSE
TRUE	FALSE	FALSE	TRUE	
TRUE	NULL	NULL	TRUE	
FALSE	TRUE	FALSE	TRUE	TRUE
FALSE	FALSE	FALSE	FALSE	
FALSE	NULL	FALSE	NULL	
NULL	TRUE	NULL	TRUE	NULL
NULL	FALSE	FALSE	NULL	
NULL	NULL	NULL	NULL	

- **Paréntesis: ()**. Sirven para alterar la prioridad de los operadores.



- **Funciones**. Funciones como **date_add**, **concat**, **left**, **right**... que incorpora cada SGBD para gestionar fechas, cadenas, números...

1.3.1.1. Construcción de filtros

Ejecutar el script 'nba.sql'

1. **Mostrar el nombre de los jugadores de los Lakers.**
2. **Mostrar el código, nombre y altura de los jugadores españoles que juegan en los Lakers.**
3. **Mostrar el nombre, altura y procedencia de los jugadores españoles y eslovenos que juegan en los Lakers.**

1.3.1.2. Filtros con operador de pertenencia a conjuntos (IN)

Además de los operadores aritméticos, lógicos, etc... se puede hacer uso del **operador de pertenencia IN**. Este operador permite comprobar si una columna tiene un valor igual que cualquiera de los que están incluidos dentro del paréntesis.

Sintaxis:

nombre_columna IN (valor1, valor2....)

4. **Mostrar el nombre, altura y procedencia los jugadores españoles, eslovenos o serbios que juegan en los Lakers.**

1.3.1.3. Filtros con operador de rango

El operador **BETWEEN**, permite seleccionar los registros que estén incluidos en un rango.

Sintaxis:

nombre_columna BETWEEN valor1 AND valor2

5. **Mostrar nombre de los jugadores, nombre del equipo y peso los jugadores de la nba cuyo peso este entre 270 y 300 libras.**

6. **Mostrar el nombre del jugador, nombre del equipo y el peso en kilogramos de los jugadores de la NBA que pesen entre 120 y 150 kilos. Una libra equivales a 0.4535 kilos.**

1.3.1.4. Filtros con test de valor nulo

Los operadores **IS NULL** e **IS NOT NULL** permiten verificar si un campo es o no nulo respectivamente.

7. **Mostrar el nombre de los jugadores y nombre del equipo de los jugadores de los que se desconoce su procedencia.**

8. **Mostrar el nombre de los jugadores y nombre del equipo de los jugadores de los que se conoce su procedencia.**

1.3.1.5. Filtros con test de patrón

- **Selecciona los registros que cumplan una serie de características.**
- **Compara una expresión con una cadena que contiene uno o más comodines**, que según el estándar son `'%'` ó `'_'`.
- Se usa sobre todo con textos, permite obtener registros cuyo valor en un campo cumpla una condición textual.
- LIKE utiliza una cadena que puede contener estos símbolos:

El carácter comodín `'%'` busca coincidencias de cualquier número de caracteres, incluso cero caracteres.

El carácter comodín `'_'` busca coincidencias de exactamente un carácter.

9. **Mostrar el nombre del equipo y procedencia de los equipos que empiecen por 'r'.**

10. **Mostrar el nombre del equipo y procedencia de los equipos que terminen en 'ts'.**

11. **Mostrar el nombre del equipo y procedencia de los equipos que contengan el carácter 'p'.**

12. **Mostrar el nombre del equipo y procedencia de los equipos que contengan como segunda letra la 'o'.**

13. **Mostrar el nombre del equipo y procedencia de los equipos que empiecen por 'r' que terminen por 's' y que tengan 7 caracteres.**

1.3.1.6. Filtros con límite de número de registros

Este tipo de filtros no es estándar, depende del SGBD. Consiste en limitar el número de registros devueltos por una consulta.

Sintaxis: LIMIT [desplazamiento,] nfilas

LIMIT [desplazamiento,] nfilas

nfilas, especifica el número de filas a devolver y **desplazamiento** especifica a partir de qué fila se empieza a contar.

14. Mostrar el nombre del equipo de los 5 primeros equipos.

15. Mostrar el nombre del equipo de los 3 primeros equipos a partir del 5º equipo.

1.4. Ordenación

Para mostrar ordenados un conjunto de registros se utiliza la cláusula **ORDER BY** de la sentencia SELECT. Permite ordenar de forma ascendente (**ASC**) valor por defecto o de forma descendente (**DESC**).

SELECT [DISTINCT] select_expr [,select_expr]
[FROM tablas]
[WHERE condiciones]
[ORDER BY {nombre_columna | expression | posicion} [ASC | DESC],...];



16. Mostrar nombre del equipo y división los equipos de la conferencia Oeste ordenados ascendentemente por división.

17. Mostrar nombre del equipo y división los equipos de la conferencia Oeste ordenados de forma ascendente por división y de forma descendente por nombre del equipo.

1.5. Consultas de resumen

Son consultas más complejas, cuyos resultados se obtienen extrayendo la información calculada de varios conjuntos de registros. El resultado **SIEMPRE** es un único valor.

FUNCIÓN DE AGREGADO	SIGNIFICADO
count(nbColumna)	Cuenta el número de valores de una columna <u>excepto</u> los nulos
count(*)	Cuenta el número de valores de una columna <u>incluyendo</u> los nulos
max(expresion)	Calcula el máximo
min(expresion)	Calcula el mínimo
sum(expresion)	Calcula la suma de los valores indicados en el argumento
avg(expresion)	Calcula la media de los valores indicados en el argumento

18. ¿Cuánto pesa el jugador más pesado de la nba?

19. ¿Cuánto mide el jugador más bajito de la nba?. Mostrar altura en pies y metros.


20. ¿Cuántos jugadores juegan en los Lakers?

21. ¿Cuántos jugadores juegan en los Warriors? Utiliza la columna procedencia como argumento para la función de resumen?

22. ¿Cuánto pesan de media los jugadores de los Bulls?

Con las consultas resumen se pueden realizar **agrupaciones de registros**. Se denomina **agrupación de registros** a un conjunto de registros que cumplen que tienen una o varias columnas con el mismo valor. Para hacer grupos de utiliza la cláusula **GROUP BY**.

```
SELECT [DISTINCT] select_expr [,select_expr]
[FROM tablas]
[WHERE condiciones]
[GROUP BY atributos]
[ORDER BY {nombre_columna | expression | posicion} [ASC | DESC],...];
```



23. Mostrar para cada equipo, el nombre del equipo, nombre y peso del jugador más pesado.

24. Mostrar por conferencia, cuántos equipos hay.


25. ¿Cuánto pesan de media los jugadores de España, Italia y Francia?

IMPORTANTE: Se observa que para cada agrupación, se ha seleccionado también el nombre de la columna por la cual se agrupa. Esto no es posible si no se incluye el GROUP BY.

1.5.1. Filtros de Grupos

Los **filtros de grupos** deben realizarse mediante el uso de la cláusula **HAVING** puesto que WHERE actúa antes de agrupar los registros. Es decir, si se desea filtrar resultados calculados mediante agrupaciones de debe usar la siguiente sintaxis:

```
SELECT [DISTINCT] select_expr [,select_expr]
[FROM tablas]
[WHERE condiciones]
[GROUP BY atributos]
[HAVING predicado de grupo]
[ORDER BY {nombre_columna | expression | posicion} [ASC | DESC],...];
```



26. Mostrar el nombre de los equipos de la nba cuyos jugadores pesen de media más de 228 libras, ordenarlos por la media del peso.

27. Seleccionar qué equipos de la nba tienen 1 o más jugadores españoles, ordenarlos por nombre del equipo.

1.6. Subconsultas

Las **subconsultas** se utilizan para realizar filtrados con los datos de otra consulta. Estos filtros pueden ser aplicados tanto en las cláusulas WHERE para filtrar registros como en la cláusula HAVING para filtrar grupos.

28. Mostrar el nombre de los jugadores que juegan en equipos de la división SouthWest.

1.6.1. Test de comparación

Consiste en usar los operadores de comparación para comparar el valor producido con un valor único generado por una subconsultas.

29. Mostrar nombre y altura del jugador más alto de la nba.

1.6.2. Test de pertenencia a conjunto


Consiste en usar el operador **IN** para filtrar los registros cuya expresión coincida con algún valor producido por la subconsultas.

30. Mostrar el nombre de la división de los equipos donde juegan jugadores españoles.

1.6.3. Test de existencia

El **test de existencia** permite filtrar los resultados de una consulta si existen filas en la subconsultas asociada, esto es, si la consulta genera un número de filas distinto de 0. Para usar el test de existencia se utiliza el operador **EXISTS**. Puede estar precedido de la negación **NOT**.

```
SELECT [DISTINCT] select_expr [,select_expr]
[FROM tablas]
[WHERE EXISTS (subconsulta)]
```



31. Mostrar el nombre de todos los equipos, si hay jugadores españoles jugando en el equipo 'Raptors'.

32. Mostrar el nombre de los equipos que tienen jugadores españoles.

33. Mostrar el nombre de los equipos que no tienen jugadores españoles.

1.6.4. Test cuantificados ALL y ANY

Los **test cuantificados** sirven para calcular la relación entre una expresión y todos (ALL) los registros de la subconsultas o algunos (ANY).

34. Nombre y peso de todos los jugadores de la nba que pesan más que todos los jugadores españoles.

35. Mostrar el nombre y peso de los jugadores base (posición = 'G') que pesan más que cualquier pivot (posición = 'C') de la nba.

1.6.5. Subconsultas anidadas

Se puede usar una subconsultas para filtrar los resultados de otra subconsulta. De esta manera se anidan subconsultas.


36. Mostrar todos los datos del equipo donde juega el jugador más alto de la nba.

1.7. Consultas Multitabla

Tutorial: <https://dev.mysql.com/doc/refman/8.0/en/join.html>

Una **consulta multitabla** es aquella en la que se puede consultar información de más de una tabla. Se aprovechan los campos relacionados de las tablas para unirlos (join).

```
SELECT [DISTINCT] select_expr [,select_expr]
[FROM referencias_tablas]
[WHERE condiciones]
[GROUP BY atributos]
[HAVING predicado de grupo]
[ORDER BY {nombre_columna | expression | posicion} [ASC | DESC],...];
```



referencias_tablas:

referencia_tabla [,referencia_tabla]

| referencia_tabla [INNER | CROSS] JOIN referencia_tabla [ON condicion]

| referencia_tabla LEFT [OUTER] JOIN referencia_tabla ON condición

| referencia_tabla RIGHT [OUTER] JOIN referencia_tabla ON condición

referencia_tabla

nombre_tabla [[AS] alias]

1.7.1. Consultas Multitabla I

El producto cartesiano de 2 tablas son las combinaciones de las filas de una tabla unidas a las filas de otra tabla.

```
SELECT * FROM PROPIETARIOS;
```

DNI	NOMBRE
2883477X	Matías Fernández
37276317Z	Francisco martinez
51993482Y	José Pérez
89651245K	Francisco martinez

```
SELECT * FROM animales;
```

CODIGO	NOMBRE	TIPO	PESO	PROPIETARIO
1	Cloncho	gato	3	51993482Y
2	Yoda	gato	2	51993482Y
3	Sprocket	gato	5	37276317Z
4	Arco	perro	8	NULL
5	Lobezno	perro	21	37276317Z

37. Calcular el producto cartesiano de las tablas propietarios y animales.

38. Mostrar todos los datos de cada mascota con todos los datos de su propietario.

1.7.2. Consultas Multitabla II

SQL introduce otra sintaxis para los siguientes tipos de consultas multitablas: las **join** (o composiciones) internas, externas y productos cartesianos (también llamadas composiciones cruzadas).

1. Join Interna
 - a. De equivalencia (INNER)
2. Join Externa:
 - a. De tabla derecha (RIGHT INNER JOIN)
 - b. De tabla izquierda (LEFT INNER JOIN)
 - c. Completa (FULL OUTER JOIN)

➤ **Combinación o Join Interna (INNER JOIN)**

Con esta operación es calculado el producto cruzado de todos los registros; así cada registro en la tabla A es combinado con cada registro de la tabla B; pero sólo permanecen aquellos registros en la tabla combinada que satisfacen las condiciones que se especifican. Es el JOIN más utilizado. Hay que tener especial cuidado con los valores NULL estos no se combinarán con otros valores ni siquiera un NULL, excepto que se le agreguen predicados tales como IS NULL ó IS NOT NULL.

39. Mostrar el nombre de cada mascota con el nombre de su propietario.

➤ **Combinación externa (OUTER JOIN)**

OUTER JOIN, mediante esta operación NO se requiere que cada registro en las tablas a tratar tenga un registro equivalente en la otra tabla. El registro es mantenido en la tabla combinada si no existe otro registro que le corresponda. Este tipo de operación se subdivide dependiendo de la tabla a la cual se le admitirán los registros que no tienen correspondencia, ya sean de tabla izquierda, de tabla derecha o combinación completa.

- De tabla izquierda (**LEFT OUTER JOIN**), el resultado de esta operación siempre contiene todos los registros de la tabla de la izquierda con los valores de la tabla de la derecha correspondientes, o retorna un valor nulo NULL en caso de no correspondencia.
- De tabla derecha (**RIGHT OUTER JOIN**), el resultado de esta operación siempre contiene todos los registros de la tabla de la derecha con los valores de la tabla de la izquierda correspondientes, o retorna un valor nulo NULL en caso de no correspondencia.
- Combinación completa (**FULL OUTER JOIN**), esta operación presenta los resultados de tabla izquierda y tabla derecha aunque no tengan correspondencia en la otra tabla. La tabla combinada contendrá, entonces, todos los registros de ambas tablas y presentará valores nulos NULLs para registros sin pareja.

40. Mostrar el código y nombre de cada mascota con el nombre de su propietario, incluidas las mascotas que no tienen propietario.

41. Mostrar el código y nombre de cada mascota con el nombre de su propietario, incluidas los propietarios que no tienen mascotas.

42. Mostrar el código y nombre de cada mascota con el nombre de su propietario, incluidas los propietarios que no tienen mascotas y las mascotas que no tienen propietario.

1.8. Consultas Reflexivas

A veces es necesario obtener información de relaciones reflexivas. Para ello, es necesario hacer una JOIN entre registros de la misma tabla.

43. Mostrar para cada mascota su nombre y el nombre de su progenitor incluidas las mascotas que no tienen progenitor.