

- DML -

Consultas básicas

Consulta de registros

La sintaxis básica para consultar registros es la siguiente:

SELECT columna1, columna2, ... **FROM** tabla;

Los nombres de columna han de ser iguales al nombre de la columna de la tabla.

El comodín * selecciona todas las columnas: **SELECT** * **FROM** tabla;

Ejemplo: La siguiente sentencia retornaría los siguientes registros:

SELECT par_nombre, par_dorsal, par_edad
FROM participantes;

par_nombre	par_dorsal	par_edad
Joan García	8272	14
Xisca Llabrés	5776	25
Eva Pons	3321	28

Nombre de las columnas

Por defecto, el resultado de una consulta tiene como nombre de la columna el mismo que la definida.

Es posible renombrarla utilizando **AS**:

```
SELECT columna1 AS nombre_nuevo FROM tabla;
```

Ejemplo: **SELECT** par_nombre **AS** titular,
par_dorsal **AS** número,
par_edat
FROM participantes;

titular	número	par_edad
Joan García	8272	14
Xisca Llabrés	5776	25
Eva Pons	3321	28

ORDER BY

Los datos se pueden ordenar de manera ascendente (**ASC**) o descendente (**DESC**). Además, se pueden ordenar diversas columnas diferentes.

Ejemplo: **SELECT** * **FROM** personas **ORDER BY** per_nombre **ASC**, per_edad **DESC**

per_nombre	per_edad	per_dni
Juan	18	12344555A
Juan	23	72626364B
Alvaro	45	33737636Z
Jorge	34	28237346N
Alvaro	56	88272363Z



per_nombre	per_edad	per_dni
Alvaro	56	88272363Z
Alvaro	45	33737636Z
Jorge	34	28237346N
Juan	23	72626364B
Juan	18	12344555A

LIMIT

Se puede limitar el número de resultados en la búsqueda utilizando **LIMIT**. Además, se puede establecer un desplazamiento utilizando **OFFSET**.

Ej.: **SELECT * FROM** personas **ORDER BY** per_nombre **ASC**, per_edad **DESC LIMIT** 2 **OFFSET** 2

nombre	edad	dni
Juan	18	12344555A
Juan	23	72626364B
Alvaro	45	33737636Z
Jorge	34	28237346N
Alvaro	56	88272363Z

se obtendría



nombre	edad	dni
Alvaro	56	88272363Z
Alvaro	45	33737636Z
Jorge	34	28237346N
Juan	23	72626364B
Juan	18	12344555A

GROUP BY

Se pueden agrupar los resultados utilizando **GROUP BY**.

Ej.: **SELECT** per_nombre, count(*) **AS** contador **FROM** personas **GROUP BY** per_nombre;

nombre	edad	dni
Juan	18	12344555A
Juan	23	72626364B
Alvaro	45	33737636Z
Jorge	34	28237346N
Alvaro	56	88272363Z

se obtendría



nombre	contador
Alvaro	2
Jorge	1
Juan	2

HAVING

Una cláusula **HAVING** permite filtrar resultados como la cláusula **WHERE**. Las condiciones de **HAVING** únicamente se aplican a los grupos en su totalidad, mientras que las condiciones de la cláusula **WHERE** se aplican a filas individuales.

Ej.: **SELECT** per_nombre, count(*) **AS** contador **FROM** personas **GROUP BY** per_nombre **HAVING** count(*) > 1;

nombre	edad	dni
Juan	18	12344555A
Juan	23	72626364B
Alvaro	45	33737636Z
Jorge	34	28237346N
Alvaro	56	88272363Z

se obtendría




nombre	contador
Alvaro	2
Juan	2

DISTINCT

Hay una función para obtener los diferentes valores de una columna determinada.

Ej.: `SELECT DISTINCT per_nombre AS Nombre FROM personas ORDER BY Nombre;`

nombre	edad	dni
Juan	18	12344555A
Juan	23	72626364B
Alvaro	45	33737636Z
Jorge	34	28237346N
Alvaro	56	88272363Z

se obtendría


Nombre
Alvaro
Jorge
Juan

Operaciones numéricas

Además de poder consultar datos de columnas, se pueden obtener resultados calculados.

Con los datos numéricos podemos realizar las siguientes operaciones:

- Operaciones matemáticas.
 - Ej.: `SELECT edad+1 AS nueva_edad, precio * 1.25 AS nuevo_preco FROM tabla;`
- Cálculos utilizando funciones simples.
 - Ej.: `SELECT round(salario) AS salario FROM tabla;`
- Cálculos utilizando funciones de agregación.
 - Ej.: `SELECT max(per_edad) AS edad_maxima FROM personas;`

Cálculos utilizando funciones simples

Las funciones se pueden utilizar con datos numéricos:

- **ABS**(n): Obtiene el valor absoluto de n .
- **CEIL**(n): Obtiene el valor entero igual o inmediatamente superior a n .
- **FLOOR**(n): Obtiene el valor entero igual o inmediatamente inferior a n .
- **MOD**(m, n): Obtiene el resto de la división de m entre n .
- **POWER**(m, exp): Calcula la potencia de m elevada a exp .
- **ROUND**($numero[, m]$): Redondea $numero$ a m decimales.
- **SQRT**(n): Retorna la raíz cuadrada de n .
- **TRUNCATE**($numero[, m]$): Trunca $numero$ para que tenga m decimales.

Ej.: **SELECT** **ABS**(-3.14*66), **ROUND**(pro_precio), **FLOOR**(pro_altura) **FROM** productos;

Cálculos utilizando funciones de agregación

Las funciones de agregación en SQL permiten realizar operaciones sobre un conjunto de resultados.

- **MIN**: Retorna el mínimo valor de la columna especificada.
- **MAX**: Retorna el máximo valor de la columna especificada.
- **SUM**: Obtiene la suma de los valores de la columna especificada.
- **AVG**: Obtiene la media de los valores de la columna especificada.
- **COUNT**: Retorna el número total de filas seleccionadas en la consulta.

Ej.: **SELECT** **MIN**(pro_precio), **SUM**(pro_precio) **AS** precio_total **FROM** productos;

Funciones para cadenas de caracteres

Existen muchas disponibles para trabajar con cadenas de caracteres.

Algunas de las más utilizadas son las siguientes:

- **CONCAT**
- **LEFT**
- **LENGTH**
- **LOWER**
- **LPAD**
- **LTRIM**
- **INSTR**
- **REPLACE**
- **REVERSE**
- **RIGHT**
- **RPAD**
- **RTRIM**
- **STRCMP**
- **SUBSTRING**
- **TRIM**
- **UPPER**

LOWER / UPPER

Estas dos funciones se utilizan para cambiar los caracteres a minúsculas (**LOWER**) o mayúsculas (**UPPER**). Reciben como parámetro una cadena de caracteres.

Ejemplos:

```
SELECT LOWER(per_nombre) AS nombre, UPPER (per_nombre) AS NOMBRE FROM personas;
```

se obtendría



nombre	NOMBRE
juan	JUAN
roberto	ROBERTO
alvaro	ALVARO

LTRIM / RTRIM / TRIM

Estas funciones se utilizan para eliminar espacios en blanco.

- **LTRIM**: Elimina espacios en blanco de la izquierda.
- **RTRIM**: Elimina espacios en blanco de la derecha.
- **TRIM**: Elimina espacios en blanco de ambos extremos.

Ejemplos:

```
SELECT LTRIM(' cadena ') AS l, RTRIM (' cadena ') AS r, TRIM (' cadena ') AS t FROM tabla;
```

se obtendría



l	r	t
'cadena '	' cadena'	'cadena'

LEFT / RIGHT

Estas dos funciones se utilizan para obtener los primeros n caracteres (**LEFT**) o los n últimos (**RIGHT**). Reciben como parámetro una cadena de caracteres u un número entero para establecer la longitud (n).

Ejemplos:

```
SELECT LEFT('Gimnasio',5) AS izquierda, RIGHT('Gimnasio',5) AS derecha FROM tabla;
```

se obtendría



izquierda	derecha
Gimna	nasio

LPAD / RPAD

Estas dos funciones se utilizan para obtener cadenas de una longitud específica i rellenar los espacios con un carácter definido. **LPAD** rellena por la izquierda y **RPAD** por la derecha. Reciben como parámetros la cadena de caracteres, el número de caracteres y el carácter con el cual rellenar los espacios.

Ejemplos:

`SELECT LPAD('12345',10,'0') AS izquierda, RPAD('12345',10,'0') AS derecha FROM tabla;`

se obtendría



izquierda	derecha
'0000012345'	'1234500000'

LENGTH / INSTR / REPLACE / REVERSE

Estas funciones no tienen una relación entre ellas como en los casos anteriores. Son muy útiles y se suelen utilizar con frecuencia.

- **LENGTH(str)**: Retorna la longitud de la cadena *str*.
- **INSTR(str, substr)**: Retorna la posición de la primera ocurrencia de la subcadena *substr* en la cadena *str*.
- **REPLACE(str, from, to)**: Reemplaza en *str* la cadena *from* por *to*.
- **REVERSE(str)**: Retorna la cadena en orden invertido.

```
SELECT LENGTH('casa') AS l, INSTR('restaurant','ura') AS i,  
       REPLACE('casa','a','o') AS r, REVERSE('jamon') AS re FROM tabla;
```

se obtendría



l	i	r	re
4	6	coso	nomaj

CONCAT / STRCMP

Estas funciones también son muy útiles y se suelen utilizar con frecuencia:

- **CONCAT(str1,str2,...)**: Concatena las cadenas pasadas como parámetro.
- **STRCMP(str1,str2)**: Compara dos cadenas. Retorna **0** si son iguales, **-1** si el primer argumento es más pequeño que el segundo según el orden de clasificación actual, y **1** en caso contrario.

```
SELECT CONCAT('Hola ','1o ','de ','DAM') AS c, STRCMP('hola','hola') AS iguales,  
STRCMP('casa','hogar') AS menor, STRCMP('hogar','casa') AS mayor FROM tabla;
```

se obtendría



c	iguales	menor	mayor
Hola 1o de DAM	0	-1	1

SUBSTRING

Esta función extrae una subcadena de la cadena pasada como parámetro.

Ejemplos:

- `SELECT SUBSTRING('Matemáticas',5) FROM tabla;`
 - Retorna: 'máticas'
- `SELECT SUBSTRING('Abogado',4) FROM tabla;`
 - Retorna: 'gado'
- `SELECT SUBSTRING('Alimentación',5,6) FROM tabla;`
 - Retorna: 'entaci'
- `SELECT SUBSTRING('Música', -3) FROM tabla;`
 - Retorna: 'ica'
- `SELECT SUBSTRING('Música', -5, 3) FROM tabla;`
 - Retorna: 'úsi'

Funciones sobre datos de tipo fecha

Las funciones más comunes para trabajar con datos de tipo fecha son:

- **CURDATE():** Obtiene la fecha actual.

```
select curdate();
```

- **CURTIME():** Obtiene la hora actual.

```
select curtime();
```

- **NOW():** Obtiene la fecha y hora actual.

```
select now();
```

- **DAY(*fecha*)**: Retorna el día de la fecha.
- **MONTH(*fecha*)**: Retorna el mes de la fecha.
- **YEAR(*fecha*)**: Retorna el año de la fecha.

- **DAYOFWEEK(*fecha*)**: Obtiene el número de día de la semana.
- **DAYOFMONTH(*fecha*)**: Obtiene el día del mes.
- **DAYOFYEAR(*fecha*)**: Obtiene el número de día del año.
- **LAST_DAY(*fecha*)**: Obtiene el último día del mes.

- **DATE_ADD, DATE_SUB**: Calcula una fecha con un intervalo determinado.
- **DATE_FORMAT**: Obtiene la fecha en un formato determinado.
- **STR_TO_DATE**: Convierte una cadena en formato fecha.

- **DATEDIFF(f1, f2)**: Devuelve el número de días entre f1 y f2.

```
SELECT DATEDIFF(NOW(), '2010-11-02'); #cuantos días han pasado  
SELECT DATEDIFF(NOW(), '2030-03-20'); #Cuantos días faltan
```

DATE_ADD / DATE_SUB

Ejemplos de uso de las funciones para calcular fechas:

- `SELECT DATE_ADD('2018-05-01',INTERVAL 1 DAY) FROM tabla;`
 - Retorna: '2018-05-02'
- `SELECT DATE_SUB('2018-05-01',INTERVAL 1 YEAR) FROM tabla;`
 - Retorna: '2017-05-01'
- `SELECT DATE_ADD('2020-12-31 23:59:59',INTERVAL 1 SECOND) FROM tabla;`
 - Retorna: '2021-01-01 00:00:00'
- `SELECT DATE_ADD('2018-12-31 23:59:59',INTERVAL 1 DAY) FROM tabla;`
 - Retorna: '2019-01-01 23:59:59'
- `SELECT DATE_SUB('1998-01-02', INTERVAL 31 DAY) FROM tabla;`
 - Retorna: '1997-12-02'

DATE_FORMAT

Formatea el valor de la fecha de acuerdo con la cadena de formato:

Esp.	Descripción
%a	Día de la semana abreviado (Sun..Sat)
%b	Día del mes abreviado (Jan..Dec)
%d	Día del mes, numérico (00..31)
%H	Hora (00..23)
%h	Hora (01..12)
%i	Minutos, numérico (00..59)
%M	Mes nombre (January..December)
%m	Mes, numérico (00..12)
%p	AM or PM
%s	Segundos (00..59)
%Y	Año, cuatro dígitos
%y	Año, dos dígitos

Ejemplos:

- `SELECT DATE_FORMAT(now(), '%d/%m/%Y') FROM tabla;`
 - Retorna: '24/10/2015'
- `SELECT DATE_FORMAT(now(), '%M %y') FROM tabla;`
 - Retorna: 'December 18'

Para obtener las fechas en español se debe cambiar el valor de la variable **lc_time_names**:

`SET lc_time_names = 'es_ES';`

now()	%a	%b	%d	%H	%h	%i	%M	%m	%p	%s	%Y	%y
2018-12-30 16:21:14	Sun	Dec	30	16	04	21	December	12	PM	14	2018	18

STR_TO_DATE

Convierte una cadena en formato fecha.

Esp.	Descripción
%a	Día de la semana abreviado (Sun..Sat)
%b	Día del mes abreviado (Jan..Dec)
%d	Día del mes, numérico (00..31)
%H	Hora (00..23)
%h	Hora (01..12)
%i	Minutos, numérico (00..59)
%M	Mes nombre (January..December)
%m	Mes, numérico (00..12)
%p	AM or PM
%s	Segundos (00..59)
%Y	Año, cuatro dígitos
%y	Año, dos dígitos

Ejemplos:

- `SELECT STR_TO_DATE('01,5,2013', '%d,%m,%Y') FROM tabla;`
 - Retorna: '2013-05-01'
- `SELECT STR_TO_DATE('May 1 2013', '%M %d %Y') FROM tabla;`
 - Retorna: '2013-05-01'

Filtrar valores NULL

Algunos registros pueden tener uno o más campos con el valor NULL.
Para poder filtrar los registros con campos NULL hay dos opciones:

- **IS NULL:** Para seleccionar los registros con valores nulos.
 - `SELECT per_nombre FROM personas WHERE per_apellido IS NULL;`
- **IS NOT NULL:** Para seleccionar los registros con valores que no son nulos.
 - `SELECT per_nombre FROM personas WHERE per_apellido IS NOT NULL;`

nombre	edad	apellido
Mario	20	Pericas
Juan	26	Montoro
Alvaro	28	<NULL>



IS NULL

nombre
Alvaro

IS NOT NULL

nombre
Mario
Juan

BETWEEN

Operador que permite filtrar registros dentro de un rango de valores establecido.

Ej.: `SELECT per_nombre FROM personas WHERE per_edad BETWEEN 26 AND 29 ;`

Mismo resultado:

`SELECT per_nombre FROM personas WHERE per_fecha BETWEEN '2006-03-02' AND '2009-04-01';`

nombre	edad	fecha
Mario	20	02/03/2000
Juan	26	20/03/2006
Alvaro	28	08/03/2008
Jorge	29	12/03/2009
Hugo	30	02/12/2009

nombre
Juan
Alvaro
Jorge

IN

Operador que permite filtrar registros dentro de una lista de valores.

Ej.: **SELECT** per_nombre **FROM** personas **WHERE** per_nombre **IN** ('Juan','Hugo');

nombre	edad	fecha
Mario	20	02/03/2000
Juan	26	20/03/2006
Alvaro	28	08/03/2008
Jorge	29	12/03/2009
Hugo	30	02/12/2009

Se obtendría



nombre
Juan
Hugo

IN

Dentro de una lista de valores se puede también especificar una SELECT.

Ej.: **SELECT** per_nombre **FROM** personas **WHERE** per_nombre **IN** (**SELECT** per_nombre **FROM** personas **WHERE** per_edad **BETWEEN** 28 **AND** 29);

nombre	edad	fecha
Mario	20	02/03/2000
Juan	26	20/03/2006
Alvaro	28	08/03/2008
Jorge	29	12/03/2009
Hugo	30	02/12/2009

Se obtendría



nombre
Alvaro
Jorge

IFNULL

La función **IFNULL** devuelve un valor por defecto cuando el registro es NULL.

Ej.: **SELECT** per_edad, **IFNULL**(per_edad,0) **AS** fecha **FROM** personas;

nombre	edad
Mario	20
Juan	null
Alvaro	null
Jorge	29
Hugo	30

Se obtendría



edad	fecha
20	20
null	0
null	0
29	29
30	30