Unidad 5
Data Query Language

DQL

Contenido:

- SELECT
- OPERADORES
- ALIAS
- DISTINCT
- CONSULTAS AGRUPADAS – GROUP BY/HAVING
- FUNCIONES - MISCELANEA

SELECT

- **SELECT**: comando para la consulta de tablas. Se **seleccionan** las columnas que se quiere mostrar como resultado de la consulta.
- Como mínimo es necesario incluir las cláusulas SELECT y FROM

```
SELECT *
  FROM sala
 WHERE idsala > 3
 ORDER BY idcine DESC , idsala ASC;
```

```
SELECT idSala, idCine, butacasSala
  FROM sala
 WHERE butacasSala > 100;
```

  - *Con * se seleccionan todas las columnas de las tablas incluidas en la cláusula FROM*

- Referencia MySQL: MySQL :: MySQL 5.7 Reference Manual :: 13.2.9 SELECT Statement

SELECT

- **SELECT:** Permite seleccionar las columnas que se quieren mostrar como resultado.
  - Para mostrar todas las columnas: *
- **FROM**: lista de tablas de las que se quiere obtener la información
- **WHERE**: Cláusula que se utiliza para aplicar filtros al resultado de la consulta
  - Se pueden utilizar expresiones lógicas (AND, OR, NOT, = , <,>, LIKE)
- **ORDER BY**: Para establecer un orden en el resultado
  - Se indica la columna por la que se quiere ordenar los registros resultado de la consulta.
  - De forma ascendente **ASC** o descendente **DESC**
  - Se puede aplicar el orden por varias columnas: 1º por población, 2º por código postal, etc.. En este caso, se indican las columnas separadas por coma ‘,’ el orden en el que aparezcan las columnas es en el que se aplicarán los criterios de ordenación.

SELECT

- **LIMIT**: Se utiliza para limitar el **número de registros a mostrar** como resultado. Es posible indicar una posición inicial para el rango del resultado.
  - Cláusula que se utiliza para aplicar filtros al resultado de la consulta

```
SELECT *
  FROM sala
 WHERE idsala > 3
 ORDER BY idcine DESC , idsala ASC
 LIMIT 5, 8; -- a partir del 5º registro, 8 registros

SELECT *
  FROM cine
 WHERE poblacionCine like 'A%'
 ORDER BY idCine DESC
 LIMIT 2; -- se devuelven únicamente los 2 primeros registros
          -- que cumplen con el filtro
```

OPERADORES

- **ARITMÉTICOS**

Operador	Descripción
+	Suma
-	Resta
*	Multiplicación
/	División
%	Módulo (resto)

```
SELECT precio, precio +precio*10/100 AS precioIVA
FROM habitacion
```

```
SELECT *
  FROM habitacion
 WHERE precio + 50 < 300;
```

- Podemos utilizar los operadores aritméticos en:
  - Cláusula SELECT: Para mostrar como una **nueva columna un valor calculado**, que puede utilizar o no el valor de otras columnas.
  - Cláusula WHERE: Para aplicar filtros sobre los registros a mostrar considerando algún cálculo

OPERADORES

- **COMPARACIÓN y LÓGICOS**

Operador	Descripción
<	Menor
<=	Menor igual
>	Mayor
>=	Mayor igual
=	Igual
<>	Distinto (!=)
BETWEEN	Entre
IS NULL	Es nulo
LIKE	Comparación texto

Operador	Descripción
AND	Y
OR	O
NOT	NEGACIÓN

- **Comodines del LIKE:**
  - **%** : cualquier valor cualquier número de veces
  - _ : cualquier valor una única vez

```
SELECT *
  FROM habitacion
 WHERE camas <> 1
   AND m2 >= 40;
```

```
SELECT *
  FROM habitacion
 WHERE m2 BETWEEN 30 and 60
    OR m2 > 90;
```

ALIAS

- **AS (alias) :** Se utilizan para dar un nombre a las columnas de la consulta.
  - Para cambiar el nombre que se muestra por defecto (columna)
  - Para dar un nombre a la expresión que se devuelve en la consulta

```
SELECT idHabitacion AS 'Número de Habitación', precio
  FROM habitacion;
```

  - *Se utilizan las comillas simples para el nombre del alias si está compuesto de varias palabras*

```
SELECT precio, precio +precio*10/100 AS precioIVA
FROM habitacion
```

DISTINCT

- **DISTINCT:** Se utiliza para eliminar duplicados en el resultado de una consulta.

```
SELECT DISTINCT idHabitacion
  FROM habitacion;

SELECT COUNT(DISTINCT idHabitacion)
  FROM habitacion;
```

GROUP BY

- **GROUP BY:** Se utiliza para agrupar el resultado de las consultas atendiendo a alguno(s) de los campos.
- Al crear las agrupaciones es posible calcular totales utilizando diferentes funciones de cálculo (funciones de agregación).
- La cláusula GROUP BY se coloca entre WHERE y ORDER BY
- Se puede agrupar por más de un campo.
- El campo o campos por los que se va a agrupar el conjunto de resultados deben colocarse en la select, habitualmente en primer lugar.

```
Select campo1, campo2, funcAgregación()
From tabla
Group by campo1, campo2;
```

GROUP BY

- Funciones para el cálculo de valores agrupados:
  - **AVG** (promedio)
  - **COUNT** (cuenta)
  - **SUM** (suma)
  - **MAX** (máximo)
  - **MIN** (mínimo)

```
SELECT MAX(m2)
  FROM habitacion;
```

```
SELECT SUM(precio)
  FROM habitacion
 WHERE m2 > 90;
```

```
-- La media del precio de las habitaciones
SELECT AVG(precio)
  FROM habitacion;
```

```
-- La habitación con más metros cuadrados
-- agrupando las habitaciones por el nº de camas
SELECT  camas, idHabitacion,max(m2)
  FROM habitacion
 GROUP BY camas;
```

HAVING

- **HAVING:** Cláusula utilizada para añadir filtros en la agrupación.
  - Se coloca después del GROUP BY y antes del ORDER BY
  - En la condición del HAVING solo pueden intervenir los campos indicados en la SELECT, ya que el filtro del HAVING se aplica sobre el resultado de la agrupación.

```
-- cuántas salas tiene cada cine
-- considerando únicamente aquellos
-- cines que tienen más de 2 salas
SELECT idCine, count(*)
  FROM sala
 GROUP BY idCine
HAVING count(*) > 2;
```

ROUND

- **ROUND:** Función para mostrar un valor decimal redondeado
  - ROUND(valor, precisión)
    - Si se invoca solo con el valor → trunca el valor decimal
    - Si se invoca con la precisión → Redondea con los decimales indicados

```
SELECT 2.345667 , ROUND(2.345667) , ROUND(2.345667 , 2)
  FROM DUAL;
```

2.345667	ROUND(2.345667)	ROUND(2.345667 , 2)
2.345667	2	2.35

IFNULL()

- **IFNULL():** Función para evaluar si una expresión es NULL y reemplazarla por otro valor si sí es NULL
- Sintaxis: IFNULL(expresión, valor_predeterminado)
  - *expresión*: valor que se evalúa, si es NULL se reemplaza por *valor_predeterminado*
  - Si *expresión* no es NULL, se mantiene el valor de *expresión*

```
select nombre, prapellido, sgapellido
  from profesor
 where dni = 48300100;
```

nombre	prapellido	sgapellido
Ian	Oxley	*NULL*

```
select nombre, prapellido, ifnull(sgapellido, '') as sgApellido
  from profesor
 where dni = 48300100;
```

nombre	prapellido	sgApellido
Ian	Oxley	

NVL()

- **NVL():** Mismo funcionamiento que IFNULL(). Se introdujo en MySQL para facilitar la migración desde Oracle a MySQL/MariaDB.

```
select nombre, prapellido, NVL(sgapellido, '') as sgApellido
  from profesor
 where dni = 48300100;
```

nombre	prapellido	sgApellido
Ian	Oxley	

LPAD / RPAD

- **LPAD() / RPAD().** Left/Right Padding: Estas funciones se utilizan para dar un determinado formato (**longitud y relleno**) a un valor o **expresión**.
- Sintaxis: LPAD(expresión, longitud, caracterRelleno)
  - *expresión*: puede ser un valor, el resultado de una función, una columna de una tabla…
  - *longitud*: la longitud que se quiere alcanzar para 'expresión'. Si la longitud es menor que la cadena 'expresión', esta se trunca desde la izqda/dcha.
  - *caracterRelleno*: cómo se rellenan las posiciones que le falten a 'expresión' para alcanzar la 'longitud'.

```
select nombre, dni, lpad(dni, 12, '0')
  from profesor
 where dni = 48300100;
```

nombre	dni	lpad(dni, 12, '0')
Ian	48300100	000048300100

```
select codcf, rpad(codCF, 5, '*')
  from ciclo
 where siglas = 'DAM';
```

codcf	rpad(codCF, 5, '*')
3	3****

```
select nombre, dni, lpad(dni, 5, '0')
  from profesor
 where dni = 48300100;
```

nombre	dni	lpad(dni, 5, '0')
Ian	48300100	48300