Repaso de SQL

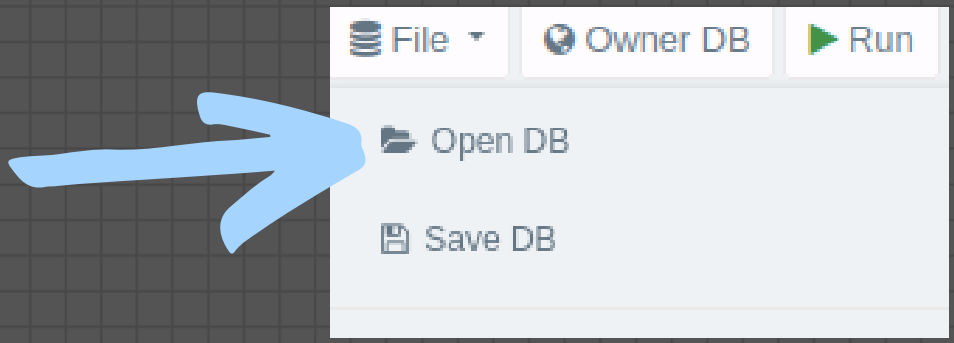
IES BENIGASLÓ

Instrucciones

- *Descargar el fichero con la BD de Aules*
- *Entrar en:*

`https://sqliteonline.com/`

- *Abrir la base de datos en la herramienta anterior:*



- *Para ejecutar consultar escribir la consulta y pulsar RUN*

BUSCAR

```
SELECT columnaN, columnaM FROM tabla;

SELECT * FROM tabla;
```

FILTRAR

```
SELECT [columnas] FROM [tabla]
WHERE [condicionales] ORDER BY [columna]
ASC/DESC LIMIT [posición]
OFFSET [número de filas];
```

`WHERE` → *crea condicionales*

`ORDER BY` → *ordena las columnas*

`LIMIT` → *limita el número de filas devueltas*

`OFFSET` → *indica la posición por la cual empezará a contar*

EJEMPLOS DE FILTRADO

LIMITAR Y ORDENAR

→ *Para listar los 5 primeros resultados:*

```
SELECT TrackId, Name, Composer
FROM Track LIMIT 5 OFFSET 0;
```

→ *¿Cómo podría obtener la fila 16, 17 y 18?*

```
SELECT TrackId, Name, Composer
FROM Track LIMIT 3 OFFSET 15;
```

→ *Ordenar por el nombre de forma ASCendente:*

```
SELECT name FROM Playlist
ORDER BY name ASC;
```

→ *Ordenar por nombre de forma DESCendente:*

```
SELECT name FROM Playlist
ORDER BY name DESC;
```

ACTIVIDAD 1

ORDENANDO A MI MANERA

- Ordena la tabla de géneros (Genre) por orden alfabético.
- Corta los resultados para que solo se muestren los 6 primeros.

Pro:

- Muestra las últimas 2 filas de la tabla.

FILTRAR

Comparaciones

```
SELECT [columnas] FROM [tabla]
WHERE [columna] = [valor];
```

Buscar patrones

```
SELECT [columnas] FROM [tabla]
WHERE [columna] LIKE 'texto%';
```

¿Quieres ignorar mayusculas y minúsculas?
Añade al final `COLLATE NOCASE`

FILTRAR

Operadores Lógicos

```
SELECT [columnas] FROM [tabla]
WHERE [columna] [> < >= <=] [valor];
```

AND / OR

```
SELECT [columnas] FROM [tabla]
WHERE [condicion]
AND/OR [condicion] AND/OR ...;
```

FILTRAR

Between

```
SELECT [columnas] FROM [tabla] WHERE
([columna] BETWEEN [valor] AND [valor]);
```

EJEMPLOS DE FILTRADO

CONDICIONALES

→ *Ciudades que han gastado un total de 1.98:*

```
SELECT BillingCity FROM Invoice
WHERE Total = 1.98;
```

→ *Ciudades que empiezan por R:*

```
SELECT DISTINCT BillingCity FROM
Invoice WHERE BillingCity LIKE 'R%';
```

→ *Facturas de Londres con gasto inferior a 5:*

```
SELECT InvoiceId, BillingCity, Total
FROM Invoice WHERE Total < 5 AND
BillingCity = 'London';
```

→ *Facturas donde el gasto está entre 10 y 12:*

```
SELECT InvoiceId, BillingCity, Total
FROM Invoice
WHERE (Total BETWEEN 10 AND 12);
```

ACTIVIDAD 2

EN BUSCA DEL PEOR CLIENTE

Volvemos a usar la tabla `Customer`.

1. Muestra los usuarios que han reportado más de 3 incidencia (columna donde se cuentan es `SupportRepid`)
2. Filtra los resultados anteriores por los que viven en Brasil.
3. Muestra a los usuarios que tengan un código postal empezado en **7**.
4. Muestra a los usuarios que tengan un email de **hotmail**.
5. Muestra los usuarios nacidos en Estados Unidos (USA) o Canadá (Canada).
6. De los resultados anteriores, muestra los que tengan un email de **gmail**.
7. Muestra al usuario que trabaja en **Apple** (columna **Company**). Te aviso que no sabes el formato de la compañía, podría ser: Apple SL, Company Apple, APPLE...
8. Muestra los usuario que han reportado entre 3 y 4 incidencias.

FILTRAR

GROUP BY

```
SELECT [tabla.columnas] FROM [tabla_1]
GROUP BY [tabla.columnas];
```

HAVING

```
SELECT [tabla.columnas] FROM [tabla_1]
GROUP BY [tabla.columnas]
HAVING [condicinales];
```

EJEMPLOS DE FILTRADO

AGRUPANDO

→ *Agrupar canciones por precio:*

```
SELECT TrackId, UnitPrice FROM Track
GROUP BY UnitPrice;
```

→ *¿Cuántas canciones hay de cada precio?*

```
SELECT COUNT(*), UnitPrice FROM Track
GROUP BY UnitPrice;
```

→ *Media de precio por canción de cada género, de mayor a menor:*

```
SELECT GenreId, AVG(UnitPrice) AS
price FROM Track GROUP BY GenreId,
UnitPrice ORDER BY price DESC;
```

→ *Media de precio por género solamente cuando sea mayor de 1:*

```
SELECT GenreId, AVG(UnitPrice) AS
price FROM Track GROUP BY GenreId,
UnitPrice HAVING price > 1;
```

ACTIVIDAD 3

LIMPIEZA DE PERSONAL

Usaremos la tabla `Employee` (empleados).

1. Cuenta la cantidad de empleados que hay por ciudad.
2. Cuenta la cantidad de empleados que hay por departamento(`Title`).

Pro:

1. Muestra la edad de cada empleado.
2. Calcula la media de edad por departamento (`Title`).
3. Cuenta los empleados que fueron contratados por año.

SUBCONSULTAS

Las tablas de clientes y facturas están relacionadas mediante el ID del cliente.

```
SELECT CustomerId FROM Invoice
WHERE InvoiceId = 12;
```


```
SELECT FirstName FROM Customer
WHERE CustomerId = 2;
```

SUBCONSULTAS

```
SELECT FirstName FROM Customer
WHERE CustomerId = (SELECT CustomerId
                    FROM Invoice
                    WHERE InvoiceId = 12);
```

¡Ojo! lo que hay entre paréntesis se ejecuta primero

SUBCONSULTAS

IN

```
SELECT InvoiceId, BillingCity, Total
FROM Invoice
WHERE InvoiceId IN (1, 2, 3);
```

Si una subconsulta puede devolver más de un valor:

```
SELECT name FROM Track WHERE AlbumId
IN (SELECT AlbumId
    FROM Album
    WHERE Title LIKE 'b%' COLLATE NOCASE);
```

ACTIVIDAD 4

EXPERTO EN CANCIONES

Volvemos a usar la tabla `Track` (canción).

1. Muestra todas las canciones con el `MediaType` **Protected AAC audio file**.
2. Muestra todas las canciones que contengan algún `MediaType` con **AAC**.
3. Muestra todas las canciones que duren más de 2 minutos.
4. Muestra todas las canciones de Jazz.
5. Averigua cual es la canción más pesada.

Pro:

1. ¿Cuantos discos tiene **Led Zeppelin**?
2. De entre sus discos, ¿cuanto cuesta el disco **Houses Of The Holy**?

FUNCIONES

MIN( ) - mínimo

```
SELECT MIN(Total) FROM Invoice;
```

MAX( ) - máximo

```
SELECT MAX(Total) FROM Invoice;
```

COUNT( ) - cuenta los elementos

```
SELECT COUNT(*) FROM Employee;
```

FUNCIONES

AVG( ) - media

```
SELECT AVG(Total) FROM Invoice;
```

SUM( ) - suma

```
SELECT SUM(Total) FROM Invoice;
```

AS - crear alias

```
SELECT InvoiceLineId AS Id,
       TrackId AS Song,
       UnitPrice AS Price
FROM InvoiceLine WHERE InvoiceId = 4;
```

ACTIVIDAD 5

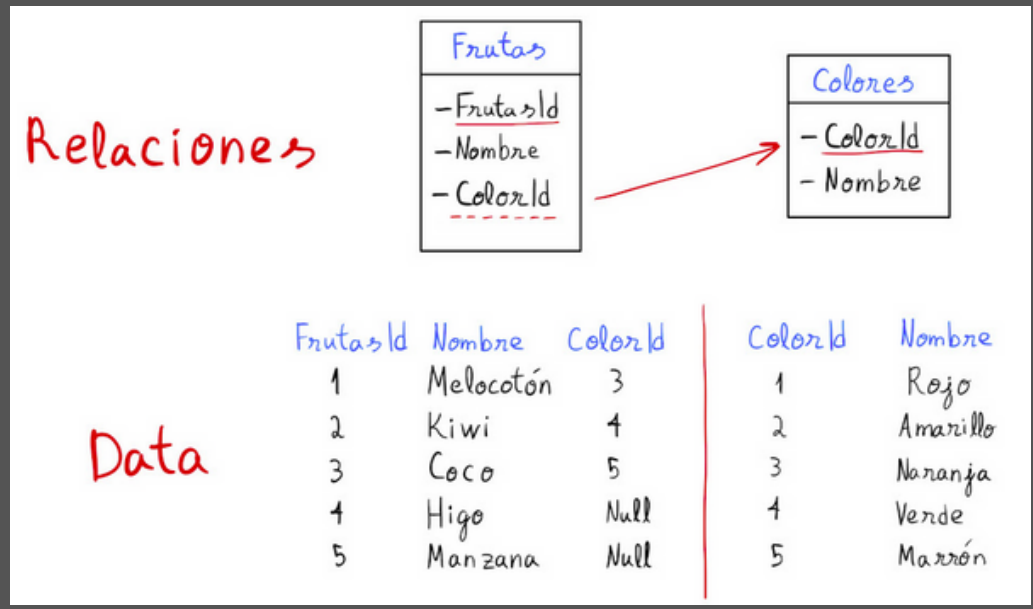
DAME LA FACTURA

De la tabla `Track`, consigue la siguiente información.

1. Cual es el título de la canción que menos pesa (`Bytes`).
2. Cual es el título de la canción que más dura (`Miliseconds`).
3. Cuantas canciones cuestan 1$ o más.
4. Cuantas canciones hay de Queen.
5. Cual es la media de duración entre todas las canciones.
6. Cual es la media de peso entre todas las canciones de U2.
7. Cuantas canciones esta Bill Berry como `Composer` (Compositor). → LIKE 'BILL BERRY%'
8. Un Mb son: Bite / 1024 / 1024. Muestra todos los `Track`s calculando, y renombrando, la columna `Bytes` en `Mb`.

JOINS

Permiten realizar consultas cuyo resultado puede estar formado por columnas de diferentes tablas.



Deben utilizarse con precaución ya que consumen muchos recursos.

JOINS

INNER JOIN

```
SELECT[tabla.columnas]
FROM [tabla_1]
INNER JOIN [tabla_2]
ON [tabla_1.columna] = [tabla_2.columna];
```

Une los valores que coinciden en ambas tablas si hay relación. En caso contrario omite la fila.

JOINS

LEFT JOIN

```
SELECT[tabla.columnas]
FROM [tabla_1]
LEFT JOIN [tabla_2]
ON [tabla_1.columna] = [tabla_2.columna];
```

Une los valores de la primera tabla con la segunda, aunque encuentre relaciones rotas o Nulas.
No omite ninguna fila de la primera tabla.

JOINS

RIGHT JOIN


```
SELECT[tabla.columnas]
FROM [tabla_1]
RIGHT JOIN [tabla_2]
ON [tabla_1.columna] = [tabla_2.columna];
```

Es la versión inversa del LEFT JOIN.
Une los valores de la segunda tabla con la primera, aunque encuentre relaciones rotas o Nulas.
No omite filas de la segunda tabla.

JOINS

FULL JOIN


```
SELECT[tabla.columnas]
FROM [tabla_1]
FULL JOIN [tabla_2]
ON [tabla_1.columna] = [tabla_2.columna];
```

No ignora ninguna fila, ni de la primera tabla ni de la segunda.

UNION

Combina los valores de diferentes consultas en una sola columna o varias. Además suprime las repeticiones.

```
SELECT[tabla.columnas] FROM [tabla_1]
UNION
SELECT [tabla.columnas] FROM [tabla_2];
```

Para mostar los datos sin eliminar las repeticiones se utiliza `UNION ALL` .

EJEMPLOS DE JOINS

→ *Muestra el nombre de las frutas y sus colores:*

```
SELECT Frutas.Nombre AS Nombre,
       Colores.Nombre AS Color
FROM Frutas
INNER JOIN Colores
ON Frutas.ColorId = Colores.ColorId;
```

→ *Muestra el nombre de las frutas y sus colores (incluyendo valores nulos):*

```
SELECT Frutas.Nombre AS Nombre,
       Colores.Nombre AS Color
FROM Frutas
FULL JOIN Colores
ON Frutas.ColorId = Colores.ColorId;
```

EJEMPLO DE UNION

→ *Muestra el nombre de las frutas y el nombre de los colores:*

```
SELECT Frutas.nombre
FROM Frutas
UNION
SELECT Colores.nombre
FROM Colores;
```

ACTIVIDAD 6

BUSCANDO FACTURAS

De la tabla `Invoice`, consigue la siguiente información.

1. Muestra: `InvoiceId`, nombre del cliente y `BillingCountry`.
2. Ordena de mayor a menor por `Total`.
3. ¿Cuánto ha facturado cada ciudad?

AÑADIR DATOS

```
INSERT INTO [tabla]
VALUES (valor1, valor2, …), (valor1,
valor2, …), …;
```

Especificando columnas

```
INSERT INTO [tabla] (column1, column2, …)
VALUES (valor1, valor2, …),
       (valor1, valor2, …), …;
```

EJEMPLOS DE INTRODUCCIÓN DE DATOS

→ *Añadir el género Trap:*

```
INSERT INTO Genre
VALUES (NULL, 'Trap');
```

NULL se usa para no especificar ningún valor, ya que se trata de una columna autoincremental

→ *Añadir los géneros Trap, Ska y Reguetón:*

```
INSERT INTO Genre VALUES (NULL,
'Trap'), (NULL, 'Ska'), (NULL,
'Reguetón');
```

→ *Añadir un cliente con solamente Nombre, Apellido y Email:*

```
INSERT INTO Customer (FirstName,
LastName, Email) VALUES ('Mickey',
'Mouse', 'mickey@disney.com');
```

ACTIVIDAD 7

ALIMENTANDO LA PRODUCTORA

1. Añade 5 artistas que te gusten en la tabla `Artist`.
2. Introduce el `MediaType` Wav.
3. Crea 2 discos que estén relacionados con los artistas que has creado.

BORRAR DATOS

```
DELETE FROM [tabla]
WHERE [condicionales];
```

Si no se usa WHERE se borrará toda la información de la tabla

ACTIVIDAD 8

ARTISTAS DESAPARECIDOS

Volvemos a usar la tabla `Artist` (canción).

(usa `PRAGMA foreign_keys = OFF` para desactivar la comprobación)

- Borra a `U2`.
- Borra los artistas que tengan en su nombre el símbolo `&`.
- Borra los artistas con una Id entre 201 y 230.
- Borra toda la tabla de `Track`.

MODIFICAR DATOS

```
UPDATE [tabla]
SET [columna] = [nuevo valor],
    [columna] = [nuevo valor], ...
WHERE [condicion];
```

EJEMPLOS DE MODIFICACIÓN DE DATOS

→ *Cambiar compañía del cliente con ID=10:*

```
UPDATE Customer
SET Company = 'Facebook'
WHERE CustomerId = 10;
```

→ *Cambiar los datos de la vivienda del cliente con ID=10:*

```
UPDATE Customer
SET City = 'Valencia', State =
'Comunidad Valenciana', Country =
'Spain' WHERE CustomerId = 10;
```

→ *Cambiar 'USA' por 'EEUU' :*

```
UPDATE Customer
SET Country = 'EEUU' WHERE Country =
'USA';
```

ACTIVIDAD 9

SE ACABARON LAS REBAJAS

Volvemos a usar la tabla `Track` (canción).

- Sube el precio de todas las canciones de 0.99 a 2.99.
- Sube el precio de todas las canciones de 1.99 a 4.99.
- Cambia el Compositor (`Composer`) por `Sara` del `TrackId` número 2000.
- Cambia el Compositor (`Composer`) por `clasico` de todas las canciones que empiecen con el nombre `Concerto`.