Gestiona bases de datos con MySQL

Día 2

Gerald Kogler

geraldo@servus.at

Instrucciones preliminares

Podéis descargar este documento aquí:
https://github.com/geraldo/curso_mysql/blob/main/mysql_dia2.pdf

Posteriormente hace falta descargar el instalador de XAMPP desde aquí:
https://www.apachefriends.org/download.html

Seguimos los pasos de instalación de XAMPP para el sistema operativo usado. Si usáis Windows entonces recomiendo instalarlo en la carpeta predefinida:
c:\xampp

Por cualquier problema que nos encontramos podemos consultar la ayuda online:
https://www.apachefriends.org/faq_windows.html

Temario día 2

4 Comandos de SQL para gestionar MySQL
4.1 INSERT, UPDATE, DELETE
4.2 CREATE, DROP

5 Funciones y vistas
5.1 Funciones de MySQL
5.2 Vistas

6 Unions, subqueries y joins de SQL
6.1 Unions
6.2 Subqueries
6.3 Joins

Herramientas de administración

Existen una multitud de herramientas de administración para MySQL/MariaDB:

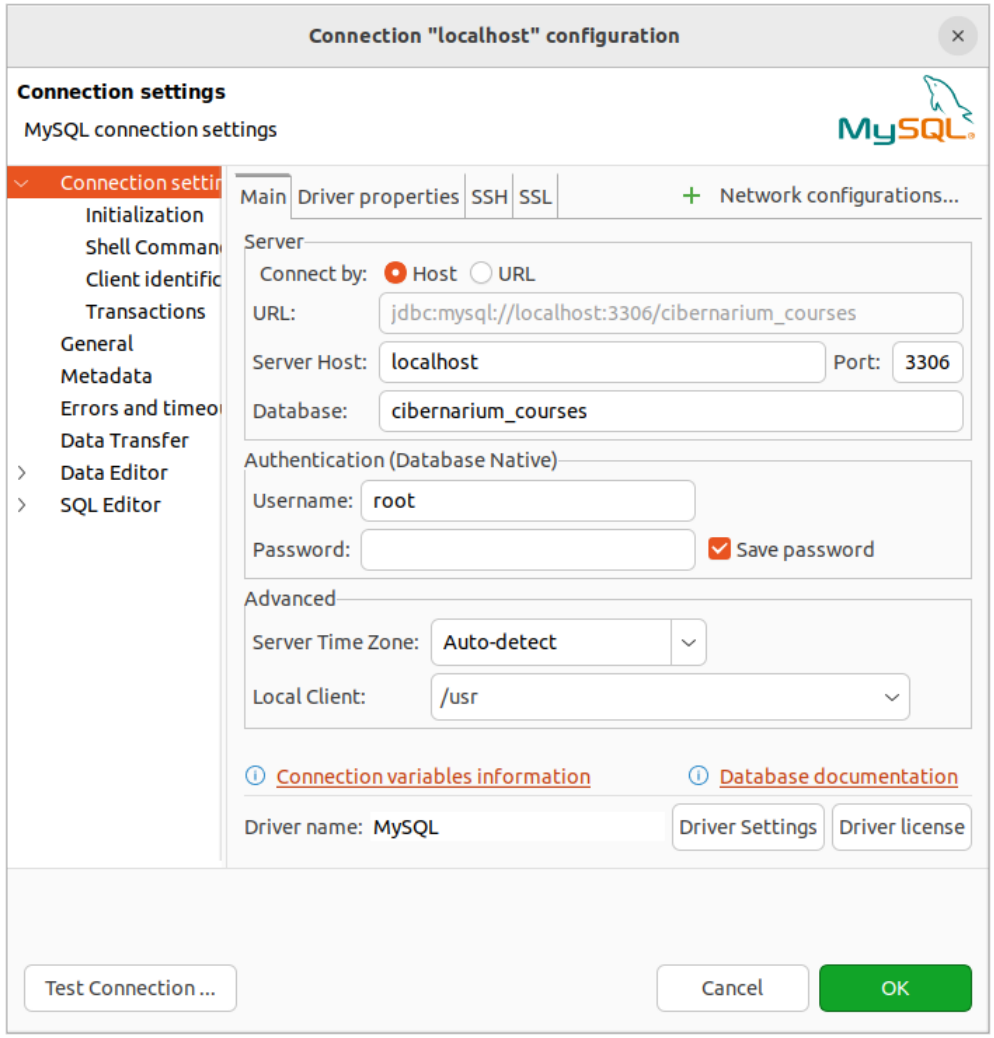
- MariaDB/MySQL Shell: Viene por definición y se gestiona desde la línea de comandos
- Adminer: Gestor ultra sencillo distribuido en un único archivo PHP (https://www.adminer.org/)
- phpMyAdmin: Gestor escrito en PHP muy popular (https://www.phpmyadmin.net/)
- Dbeaver: Herramienta muy potente para gestionar una multitud de bases de datos (https://dbeaver.io/)
- ... (https://en.wikipedia.org/wiki/MySQL#Other_GUI_tools)

Para conectar desde cualquier cliente a nuestra base de datos MySQL, necesitamos los siguientes datos:

- Server Host: *localhost*
- Port: *3306*
- Username: *root*
- Password: *[mysql no tiene contraseña por definición en XAMPP]*

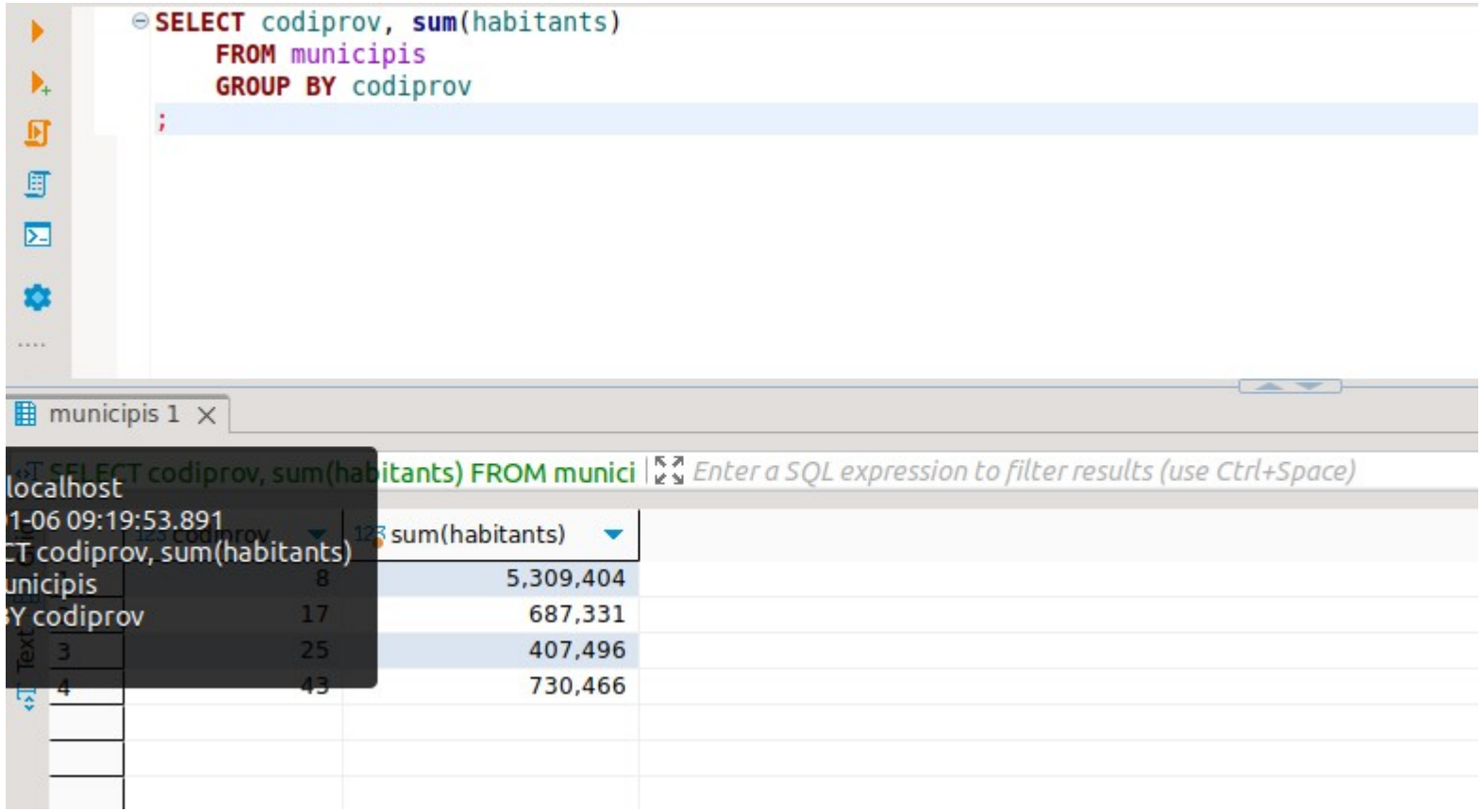
Herramientas de administración

Ahora usaremos **DBeaver** para conectarnos a nuestra base de datos *cibernarium_courses*:



Herramientas de administración

DBeaver tiene su propio **SQL Editor** para ejecutar consultas de SQL:



Herramientas de administración

Ahora utilizaremos **DBeaver** para resolver los ejercicios que nos quedaron pendientes de la primera clase.

Importamos el fichero SQL **municipis.sql** que contiene todos los municipios de Cataluña.

Resuelve los siguientes **tareas** empleando los diferentes ordenes SQL que aprendimos:

1. ¿Qué id tiene la comarca del Bages?
2. ¿Cuántos municipios hay en la comarca del Baix Llobregat?
3. ¿Cuántos municipios tienen más de 1000 metros de altitud y más de 1000 habitantes?
4. ¿Cuántos municipios tiene más de 200 km2?
5. ¿Cuáles son los 10 municipios de Cataluña con más habitantes ordenados con el más poblado primero?
6. ¿Cuáles son las 5 comarcas de Cataluña con más área (en km²) ordenados con la más grande primera?
7. ¿Cuántos municipios hay en cada comarca?
8. ¿Cuántos habitantes hay en cada provincia de Cataluña?

4 - Comandos de SQL para gestionar MySQL

MySQL – SQL Alter Table

Utilizaremos ALTER TABLE para modificar la estructura de una tabla existente. Podemos tanto modificar una columna existente como añadir una nueva.

Añadimos una columna nueva que contenga el teléfono de los participantes de los cursos:

```
ALTER TABLE participantes
ADD COLUMN telefono varchar(20);
```

Para borrar una columna hacemos lo siguiente:

```
ALTER TABLE participantes
DROP COLUMN telefono;
```

MySQL – SQL Alter Table

Para definir la id como auto-incrementar, hacemos lo siguiente:

```
ALTER TABLE participantes
MODIFY COLUMN participante_id INT NOT NULL AUTO_INCREMENT;
```

Para dar un valor por defecto a la columna *direccion,* hacemos lo siguiente:

```
ALTER TABLE participantes
MODIFY COLUMN direccion VARCHAR(250) DEFAULT NULL;
```

Para definir *curso_id* como índice de otra tabla, hacemos lo siguiente:

```
ALTER TABLE participantes
ADD CONSTRAINT fk_curso_id FOREIGN KEY (curso_id) REFERENCES cursos(curso_id);
```

Ejercicio:

- Añadimos una columna nueva a la tabla de municipios nombrada *nomprov* para guardar el nombre de las provincias

MySQL – SQL Constraints

Podemos definir reglas como se insertan los datos en las columnas usando los siguientes limitadores:

- NOT NULL - Aseguramos que la columna no puede ser NULL.
- UNIQUE - Aseguramos que cada valor de la columna es único.
- PRIMARY KEY - Es una combinación de NOT NULL y UNIQUE. Indentifica cada valor de la columna.
- FOREIGN KEY - Evita destruir enlaces entre tablas.
- CHECK - Asegura que la condición de la columna cumpla una cierta condición.
- DEFAULT - Define un valor predefinido en caso que no haya valor al insertar.
- CREATE INDEX - Usado para crear y conseguir datos de una tabla muy rápido.

MySQL – SQL Insert

Sirve para añadir filas a una tabla. Básicamente hay dos formas de llamar al INSERT:

1. Especificar nombres de columnas y valores a insertar:

```
INSERT INTO table_name (column1, column2, column3, ...)
VALUES (value1, value2, value3, ...);
```

2. Si vas a insertar valores para todas las columnas, entonces no hace falta nombrarlas. En este caso hay que asegurar que el orden de los valores es igual al orden de columnas en la tabla:

```
INSERT INTO table_name
VALUES (value1, value2, value3, ...);
```

En este caso si hace falta que las columnas para los valores que no vamos a insertar tengan un valor por defecto.

MySQL – SQL Insert

Para probar insertamos nuevos registros a la tabla de participantes de cursos:

```
INSERT INTO participantes
VALUES (4, "Ana", "Tarragona", 2);
```

En el caso que no queremos rellenar todas las columnas - y incluso podemos cambiar el orden de valores a insertar) – tenemos que definir los nombres de columnas de la siguiente manera:

```
INSERT INTO participantes (curso_id, nombre)
VALUES (2, "Joan");
```

Ejercicios:

- Añadimos un nuevo curso definiendo los valores de todas las columnas.
- Añadimos un nuevo curso solamente definiendo el nombre y el tutor.

MySQL – SQL Update

UPDATE actualiza valores de la tabla:

```
UPDATE table_name
SET column1 = value1, column2 = value2, ...
WHERE condition;
```

Para modificar la dirección de una participante de curso hacemos lo siguiente:

```
UPDATE participantes
SET direccion="Tarragona"
WHERE participante_id=5;
```

Cuidado con UPDATE! En caso de usarlo sin un WHERE todos los registros se sobreescriben!

Ejercicios:

- Actualizamos la tabla *municipis* insertando todos los nombres de las provincias a la columna *nomprov* utilizando los códigos oficiales de https://www.idescat.cat/codis/?id=50&n=11&lang=es

MySQL – SQL Delete

Elimina filas o registros de la tabla según el criterio elegido. Si no hay criterio, elimina todos los valores de la tabla (pero no la tabla en si).

```
DELETE FROM table_name
WHERE condition;
```

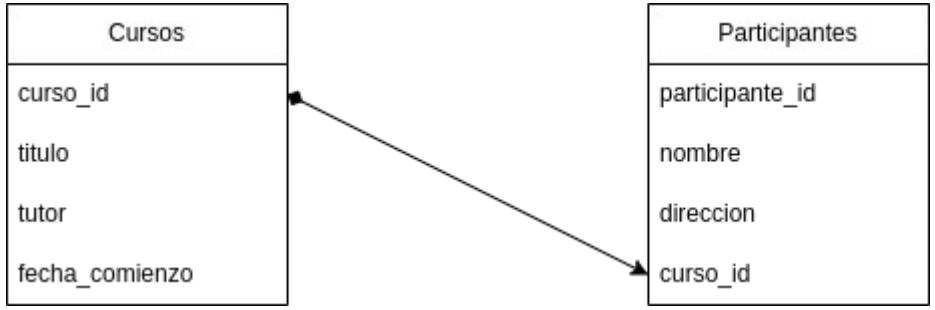
Para borrar un participante de un curso, hagamos lo siguiente:

```
DELETE FROM participantes
WHERE participante_id=6;
```

MySQL – SQL Create

Aquí repetimos la orden de crear una tabla, ahora con varios constraints definidos:

```
CREATE TABLE participantes (
        participante_id INT NOT NULL AUTO_INCREMENT,
        nombre VARCHAR(100) NOT NULL,
        direccion VARCHAR(100) DEFAULT NULL,
        curso_id INT NOT NULL FOREIGN KEY (curso_id),
        PRIMARY KEY (participante_id),
        FOREIGN KEY (curso_id) REFERENCES cursos(curso_id)
);
```



En caso que ya existe una tabla con el mismo nombre, CREATE no se ejecuta. Si queremos sobrescribir una tabla existente, podemos escribir CREATE OR REPLACE.

MySQL – SQL Drop

DROP TABLE sirve para eliminar una tabla completa, incluyendo estructura y datos:

```
DROP TABLE table_name;
```

En el caso que queremos eliminar todos los datos, pero no la tabla misma, usamos TRUNCATE TABLE:

```
TRUNCATE TABLE table_name;
```

Para borrar toda una base de datos, usamos DROP DATABASE:

```
DROP DATABASE databasename;
```

5 – Vistas y funciones

MySQL – SQL Views

Vistas son consultas en forma de tabla. Externamente funcionan idénticamente a las tables. Esto permite hacer tables dinámicas que obtienen los valores de otras tables.

Muy útil para hacer una capa con información y cargarla en una aplicación web (Wordpress) o escritorio (QGIS). Eso tiene el efecto que la información puede estar guardada en diferentes lugares pero la aplicación lo ve como una sola tabla con información.

Una vista siempre muestra la información actualizada de las tablas utilizadas, es decir, cada vez que se actualizan datos en una tabla utilizada por una vista, esta vista se vuelve a crear.

```
CREATE VIEW view_name AS
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

Igual que las tablas y las bases de datos, podemos borrar una vista usando DROP VIEW:

```
DROP VIEW view_name;
```

MySQL – SQL Views

Creamos una vista de los municipios con más de 10.000 habitantes en la provincia de Girona:

```
CREATE OR REPLACE VIEW `[Municipios grandes en Girona]` AS
SELECT codimuni, nommuni, habitants
FROM municipis
WHERE codiprov=17 AND habitants>10000;
```

Ejercicios:

- Crear una vista de todos los municipios de la comarca del Alt Penedès.
- Crear una vista de todos los municipios de Catalunya en una altura de más de 1000 metros.

MySQL – MySQL Functions

MySQL tiene funciones para tratar cadenas de texto, números y fechas. Aquí trataremos algunos para entender su aplicación, un listado completo de las funciones encuentras en https://www.w3schools.com/mysql/mysql_ref_functions.asp

Algunas funciones de cadenas de texto:

UPPER() convierte una cadena de texto en mayúsculas:

```
SELECT UPPER("Aprendemos SQL y más!");
SELECT UPPER(titulo) from cursos;
```

LENGTH() devuelve la cantidad de caractéres de una cadena de texto:

```
SELECT LENGTH("SQL");
SELECT nombre, LENGTH(nombre) AS caracteres from participantes;
```

CONCAT() junta varias cadenas de texto en una cadena única:

```
SELECT CONCAT(" Aprendemos ", "SQL ", "y ", "más!");
SELECT CONCAT(tutor, " imparte el curso ", titulo) from cursos;
```

MySQL – MySQL Functions

Algunas funciones de números:

Anteriormente ya vimos las funciones de COUNT(), SUM(), AVG(), MIN() y MAX(), aquí otros funciones de tratamiento de números:

FLOOR() devuelve un número entero:

```
SELECT FLOOR(25.75);
```

ROUND() ronda un número con la cantidad de decimales indicadas:

```
SELECT ROUND(135.375, 2);
```

MySQL – MySQL Functions

Algunas funciones de fechas:

NOW() devuelve la fecha y hora actual:

```
SELECT NOW();
```

DATE_FORMAT() formatea una fecha según el formato indicado:

```
SELECT DATE_FORMAT("2024-12-31", "%Y");
SELECT DATE_FORMAT(fecha_comienzo, "%W %M %e %Y") FROM cursos;
```

El listado completo de formatos a usar verás en este listado: https://www.w3schools.com/mysql/func_mysql_date_format.asp

MySQL – MySQL Functions

Algunas funciones avanzadas:

IF() permite especificar un condicional:

```
SELECT IF(500>1000, "YES", "NO");
SELECT titulo, IF(NOT STRCMP(tutor, "Gerald Kogler"),"soy yo","NO soy yo") FROM
cursos;
```

CASE() repasa múltiples condiciones y devuelve la primera que se cumpla:

```
SELECT count(*),
CASE
                    WHEN continent="South America" OR continent="North America" THEN "America"
                    ELSE continent
END as continent_group
FROM `countries`
GROUP BY continent_group;
```

ISNULL() comprueba si un valor es NULL:

```
SELECT ISNULL(NULL);
```

6 – Unions, subqueries y joins de SQL

MySQL – SQL Union

La instrucción UNION une las filas de los resultados de múltiples queries en la primera. Para hacerlo es obligatorio que cada SELECT tiene la misma cantidad de columnas, que cada columna tenga el mismo tipo de datos y que estén en el mismo orden en cada SELECT. Mantendrán el nombre de las columnas de la primera query.

Para aplicar instrucciones al resultado se tiene que usar una subquery.

```
SELECT 1 AS resultat, id FROM tabla 1
UNION
        SELECT 2, id FROM tabla 2
;
```

Ejercicios:

- Listar los nombres de municipios con sus códigos de las tablas municipis y de centres_docents en una sola tabla.

MySQL – SQL Subqueries

Las subquerys permiten utilizar el resultado de una query en otra query (de aquí su nombre). Estas subquerys pueden servir de tablas hijas (si es limita a una fila por cada subquery) o valores (si tienen una hija y una columna).

```
SELECT m.nomprov
FROM municipis as m
WHERE codimuni = (
    SELECT `Codi municipi_6`
    FROM centres_docents
    WHERE BATX="BATX"
    GROUP BY `Codi municipi_6`
    ORDER BY count(*) DESC
    LIMIT 1
);
```

Nota: La subquery no acaban en punto y coma (;) ya que el comando no se ejecuta hasta que la query principal se acaba.

MySQL – SQL JOIN

Importamos el fichero **centres_docents.sql** con todos los centros educativos de Cataluña. Este fichero tiene una columna llamada *codi_municipi_6* que nos permite relacionarlo con la tabla de municipios, que tiene el código del municipio en la columna *codimuni*.

Tipos de JOIN:

- INNER JOIN: Devuelve los registros que tiene valores que cumplan las condiciones en las dos tablas.
- LEFT JOIN: Devuelve todos los registros de la tabla izquierda y solamente los que cumplan las condiciones de la tabla derecha.
- RIGHT JOIN: Devuelve todos los registros de la tabla derecha y solamente los que cumplan las condiciones de la tabla izquierda.
- CROSS JOIN: Devuelve todos los registros de las dos tablas.

MySQL – SQL JOIN

SQL JOIN permite agrupar dos (o más) tablas en una sola en la misma consulta. La condición que agrupa las diferentes filas de las tables es la condición ON.

```
SELECT m.nommuni, c.`denominació completa`
FROM municipis AS m
LEFT JOIN centres_docents AS c
ON m.codimuni = c.`Codi municipi_6`
GROUP BY m.nommuni, c.`denominació completa`
ORDER BY m.nommuni;
```

Como se puede ver en el ejemplo se pueden utilizar alias por las tablas. Esto es muy útil para no tener que reescribir el nombre de la tabla entera cada vez.

MySQL – SQL JOIN

Miremos la diferencia de la misma query usando un INNER JOIN:

```
SELECT m.nommuni, c.`denominació completa`
FROM municipis AS m
INNER JOIN centres_docents AS c
ON m.codimuni = c.`Codi municipi_6`
GROUP BY m.nommuni, c.`denominació completa`
ORDER BY m.nommuni;
```

En este caso solamente nos devuelve los municipios que tengan un centro asociado, es decir, no muestra municipios con el centro con valor NULL.

En este caso el resultado devuelve los mismos resultados que haciendo un RIGHT JOIN o un CROSS JOIN ya que todos los centros docentes tienen un municipio asociado.

MySQL – SQL JOIN

Se pueden encadenar múltiples JOIN ... ON ... JOIN ... ON ... para juntar más de dos tables. La manera de como se juntaran dependerá de la relación especificada en ON.

Los joins son muy comunes en los modelos relacionales ya que los datos finales suelen estar en otra tabla (para evitar repeticiones).

```
SELECT *
        FROM taula1 AS p
                LEFT JOIN taula2 AS s
                        ON p.t2 = s.id
                LEFT JOIN taula3 AS t
                        ON s.t3 = t.id
        WHERE p.id > 100
        LIMIT 100
;
```

Ejercicio:

- Mostramos un listado de todos los bachilleratos (BATX) por municipio ordenado por cantidades.
- Mostramos un listado de todos los institutos (ESO) por provincia.

Ejercicios SQL

Para terminar la sesión practicamos algunos comandos SQL que aprendimos. Para eso cargamos un nuevo fichero de datos con consumos energéticos llamado **consums_electrics.sql**.

1. ¿Qué consumo eléctrico tiene el municipio Prat de Llobregat en el 2022?
2. ¿Cuál es el municipio de Cataluña que tiene más consumo eléctrico industrial en el 2019?
3. ¿Cuánto consumo eléctrico tiene la industria en la comarca del Baix Llobregat en el 2019?
4. Agrupa el consumo eléctrico de Cataluña por sectores.
5. Agrupa el consumo eléctrico de la comarca de Barcelona por municipios.
6. Agrupa el consumo eléctrico de la comarca de Barcelona por municipios y sectores.
7. Mostramos un listado de todos los consumos privados por provincia del 2022.
8. Mostramos un litado de todos los consumos privados por municipio y habitante en el 2022.

Espero que os ha agradado esta sesión.
Muchas gracias por vuestra asistencia.

Gerald Kogler

geraldo@servus.at

https://geraldo.github.io
https://github.com/geraldo
https://gitlab.com/geraldo1