**SQL**

**CONCEPTOS BÁSICOS SQL**

* **CREAR TABLAS:**

***-CREATE TABLE groceries (id INTEGER PRIMARY KEY, name TEXT, quantity INTEGER );🡪***primero el identificador, luego el nombre indicando de qué tipo es ese dato(texto), luego cantidad indicando de qué tipo es ese dato (numérico).

-INTRODUCIR COLUMNAS:***INSERT INTO groceries VALUES (1, "Bananas", 4)****;*🡪instrucción “insert into para introducir columnas, metemos id, luego nombre y luego cantida, que es el orden que hemos indicado al crear la tabla)

* ***INSERT INTO groceries VALUES*** *(2, "Peanut Butter", 1)*
* ***INSERT INTO groceries VALUES*** *(3, "Dark chocolate bars", 2);*
* **CONSULTAR TABLAS:**

***-SELECT name FROM groceries;****🡪* nos mostrará los nombres que componen la tabla.

***-SELECT \* FROM groceries****;🡪* añade todas las columnas.

***-SELECT \* FROM groceries ORDER BY aisle;🡪*** ordena la lista según la columna que le indiques

***- SELECT \* FROM groceries WHERE aisle > 5 ORDER BY aisle;🡪*** sacará las columnas con un valor mayor a 5 en la columna indicada (aisle).

**-SELECT SUM(quantity) FROM groceries;🡪** suma todas las cantidades de la columna “ quantity, en la tabla “ groceries”.

***SELECT MAX(quantity) FROM groceries;🡪*** selecciona el mayor elemento en la columna “ quantity”.

***-SELECT aisle, SUM(quantity) FROM groceries GROUP BY aisle;🡪*** agrupa los elementos de un mismo pasillo en este caso.( lo que seleccionas agrupas)

**CONSULTAS DE SQL MÁS AVANZADAS**

-**El id se puede autoincrementar** por el programa sin tener que hacerlo nosotros mediante id **INTEGER PRIMARY KEY AUTOINCREMENT** en cuyo caso tenemos que indicar cuando añadimos filas mediante “INSERT INTO”los nombres de las columnas y después los valores que se corresponden.

**-Si queremos unir dos o más condiciones usamos /\* AND \*/,** por ejemplo: SELECT \* FROM exercise\_logs WHERE calories > 50 AND minutes < 30;donde nos sacará la opción en la que se cumplen ambas condiciones ( calorías mayor que 50 y minutos menor a 30)

**-Si queremos que saque varias líneas que cumplan una u otra condición usamos /\* OR \*/**, por ejemplo SELECT \* FROM exercise\_logs WHERE calories > 50 OR heart\_rate > 100;donde nos sacará las filas que cumplan una u otra condición.

**-Se pueden usar varios AND y OR seguidos.** El AND tiene prioridad frente al OR

**-Para hacer una consulta anidada usamos IN:** esto quiere decir por ejemplo:*SELECT \* FROM exercise\_logs WHERE type IN (*

 *SELECT type FROM drs\_favorites)*;Dentro de la primera tabla de ejercicios sacará las que coincidan con los “type” que aparecen en la lista del doctor.

**-Having** se usa para referirnos a una lista creada previamente en la que se agrupan grupos “GROUP BY”. En el having podemos poner condiciones.Having se usa cuando hay que hacer operaciones intermedias, sin embargo where es para un valor concreto, sin operar.

**-** Para hacer la media usamos **AVG** seguido por el elemento del cual queremos obtener la media entre paréntesis.

**-GROUP BY type HAVING COUNT(\*) >= 2**;, para saber que actividades han aparecido 2 veces ó más.

**-Cálculo de resultados con Case:🡪** SELECT COUNT(\*) FROM exercise\_logs WHERE heart\_rate > 220 - 30; indica cuantas filas existen en este caso con un “heart rate” de 220-30.

**-CASE:es algo parecido a un “if”**

SELECT type, heart\_rate,

 **CASE**

 **WHEN** heart\_rate > 220-30 **THEN** "above max"(le damos una condición y un nombre)

 **WHEN** heart\_rate > ROUND(0.90 \* (220-30)) **THEN** "above target"(le damos otra condición y otro nombre)

 **WHEN** heart\_rate > ROUND(0.50 \* (220-30)) THEN "within target"

 **ELSE** "below target"( en caso de que no cumpla ninguna de las condiciones entrará en este caso)

 **END as "hr\_zone"🡪** siempre acaba así un case, y le damos el nombre a la nueva columna

FROM exercise\_logs;