***SQL***

SELECT \* FROM celebs;

CREATE TABLE table\_name (  
   column\_1 data\_type,   
   column\_2 data\_type,   
   column\_3 data\_type  
);

Ejemplo:

CREATE TABLE  celebs (

   id INTEGER,

   name TEXT,

   age INTEGER

 );

Rellenar con info las columnas:

INSERT INTO celebs (id, name, age)   
VALUES (1, 'Justin Bieber', 22);

Añadir columna:

ALTER TABLE celebs   
ADD COLUMN twitter\_handle TEXT;

Añadir info a la nueva columna para un ID en particular:

UPDATE celebs   
SET twitter\_handle = '@taylorswift13'   
WHERE id = 4;

Eliminar todas las filas donde un valor sea null:

DELETE FROM celebs

WHERE twitter\_handle IS NULL;

Crear tabla dando más restricciones/información al tipo de valor que vaya a haber en las columnas:

CREATE TABLE celebs (  
   id INTEGER PRIMARY KEY,   
   name TEXT UNIQUE,  
   date\_of\_birth TEXT NOT NULL,  
   date\_of\_death TEXT DEFAULT 'Not Applicable'  
);

LLamar a una columna con un alias:

SELECT name AS 'Titles'  
FROM movies;

Buscar los distintos valores de una variable:

SELECT DISTINCT genre   
FROM movies;

Buscar en una columna bajo un condicional:

* = equal to
* != not equal to
* > greater than
* < less than
* >= greater than or equal to
* <= less than or equal to

SELECT \*   
FROM movies   
WHERE imdb\_rating < 5;

Buscar un patron en nombres:

La ‘\_’ funciona como comodin, en ejemplo funionaría para ‘seven’ y ‘se7en’

SELECT \*

FROM movies

WHERE name LIKE 'Se\_en';

El ‘%’ se utiliza para buscar todo lo que haya antes y/o después de la palabra:

En el ejemplo (‘%man%’) buscaremos todo lo que va antes y después de man, por ejemplo ‘Iron Man 2’ o ‘Batman’.

SELECT \*

FROM movies

WHERE name LIKE 'The %';

Como encontrar valores nulos o no nulos en una columna:

Pondremos ‘null’ o ‘not null’ para la clumna en la que busquemos

SELECT name

FROM movies

WHERE imdb\_rating IS NULL;

La palabra ‘BETWEEN’ sireve para acer busquedas entre dos valores como números, pero también palabras:

En el ejemplo buscaremos todas los nombres que empiecen por A, B,C.. hasta J. De tal manera que si hubiese un nombre que se llamase ‘J’ lo encontraría, pero ‘Jaws’ no lo encontraría.

SELECT \*

FROM movies

WHERE name BETWEEN 'A' AND 'J';

Para hacer varias restricciones de búsqueda a la vez usaremos ‘AND’:

SELECT \*

FROM movies

WHERE year BETWEEN 1970 AND 1979

  AND imdb\_rating > 8;

Al igual que ‘AND’ es si se dan todas las condiciones, con ‘OR’ haremos búquedas donde se cumpla alguna de las condiciones:

SELECT \*

FROM movies

WHERE genre = 'comedy'

   OR genre = 'romance';

Para Ordenar las búsquedas (ASC y DESC):

SELECT name, year

FROM movies

ORDER BY name DESC;

Para ver solo algunos ejemplos de la búsqueda pondremos un límite:

SELECT \*

FROM movies

ORDER BY imdb\_rating DESC

LIMIT 3;

Para coger los valores únicos o distintos de una columna:

SELECT DISTINCT year

FROM movies;

Generaremos una nueva columna condicional con un caso ‘CASE’ y ‘END’:

Con ‘when’ pondremos los condicionales y con ‘AS’ el nombre de la nueva columna

SELECT name,

 CASE

  WHEN genre = 'romance' THEN 'Chill'

  WHEN genre = 'comedy' THEN 'Chill'

  ELSE 'Intense'

 END AS 'Mood'

FROM movies;

Agregadores básicos:

* COUNT(): count the number of rows
* SUM(): the sum of the values in a column
* MAX()/MIN(): the largest/smallest value
* AVG(): the average of the values in a column
* ROUND(): round the values in the column

Así contaremos todas las apps que son gratis de nuestra DB

SELECT COUNT(\*)

FROM fake\_apps

where price = 0;

Así sumaremos todas las descargas de todas las apps:

SELECT SUM(downloads)

FROM fake\_apps;

Podremos redondear los valores de una columna:

Con ‘0’ tendremos cero valores decimales

SELECT name, ROUND(price, 0)

FROM fake\_apps;

Para hacer una doble restriccion y sacar la media con dos valores decimales

SELECT ROUND(AVG(price), 2)

FROM fake\_apps;

Para agrupar segun un valor utilizaremos el ‘Group By’:

Según los distintos precios, cuenta las aplicaciones que tuviesen mas d3 20.000 descargas

SELECT price, COUNT(\*)

FROM fake\_apps

WHERE downloads > 20000

GROUP BY price;

Otro ejemplo mas complicado, para cada categoría sacar la media de descargar para cada precio. Cada tipo de categoría, cada precio sacas media de desc:

SELECT category,

   price,

   AVG(downloads)

FROM fake\_apps

GROUP BY category, price;

Para unir tablas a través de IDs, escribiremos ‘FROM’ y ‘JOIN’ para unir las distintas tablas; ‘ON’ escribiremos nombre\_tabla.nombre\_variable = nombre\_tabla2.nombre\_variable2 sobre las que queremos unir.

SELECT orders.order\_id,

   customers.customer\_name

FROM orders

JOIN customers

  ON orders.customer\_id = customers.customer\_id;

Se pueden utilizer ‘left join’ (la primera sería la base a la que se añadiría la información) ‘right join’ al revés y ‘cross join’ en donde se mantendría toda la información de ambas tablas y donde no hay información lo dejará en null.

También puedes usar ‘UNION’ que uniría por debajo las dos tablas, sería como sumarla por debajo.

SELECT \*

FROM newspaper

UNION

SELECT \*

FROM online;

Con ‘WITH’ podemos hacer referencia a la query anterior, darla un nombre y poder trabajarla como si fuera una tabla temporal:

En este caso la damos el nombre de previous\_query a la tabla temporal con la que haremos un join con customers.

WITH previous\_query AS (

   SELECT customer\_id,

      COUNT(subscription\_id) AS 'subscriptions'

   FROM orders

   GROUP BY customer\_id

)

SELECT customers.customer\_name,

   previous\_query.subscriptions

FROM previous\_query

JOIN customers

  ON previous\_query.customer\_id = customers.customer\_id;