**GUIA BÁSICA SQL**

1. **CREAR TABLA (CREATE TABLE)**

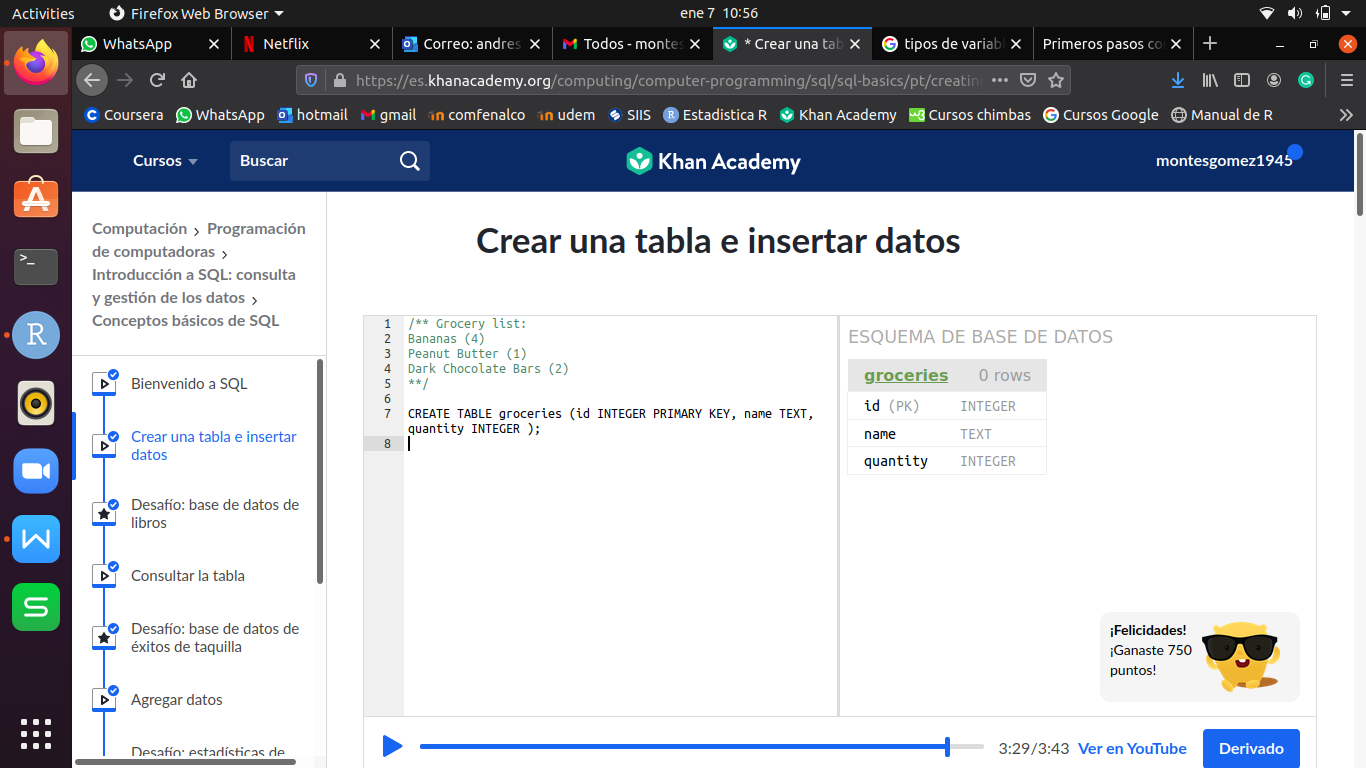
CREATE TABLE nombre\_de\_tabla (id INTEGER PRIMARY KEY, col1\_name TYPE, col2\_name TYPE, ...., coln\_name TYPE);

- Lo ideal es siempre agregar, inicialmente, un id como primary key.

- Los **TYPE** más utilizados son: **TEXT** (texto), **INTEGER** (número entero: no acepta decimales), **REAL** (acepta decimales) y **BLOB** (se almacenan con el mismo formato que se introduce).

**EJEMPLO:**

CREATE TABLE groceries (id INTEGER PRIMARY KEY, name TEXT, quantity INTEGER );



1. **INSERTAR VALORES EN LA TABLA (INSERT INTO)**

INSERT INTO nombre\_de\_tabla VALUES (‘text’, 5, 2,7, ....);

- Por cada linea que se desee ingresar, se debe hacer un INSERT.

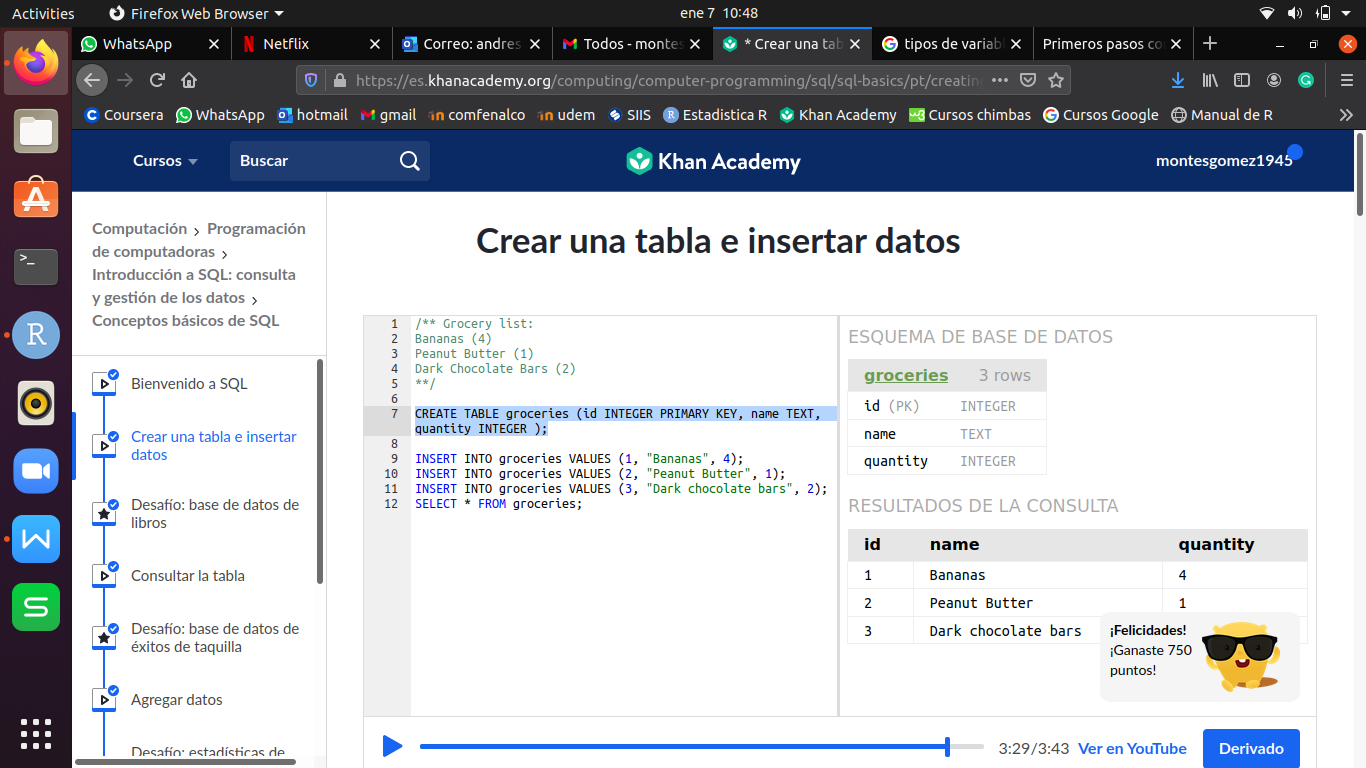
- Las columnas que sean texto deben ir entre comillas, no importa si son dobles (“ ”) o simples (‘ ’), SQL acepta cualquiera de las dos.

**EJEMPLO:**

INSERT INTO groceries VALUES (1, "Bananas", 4);

INSERT INTO groceries VALUES (2, "Peanut Butter", 1);

INSERT INTO groceries VALUES (3, "Dark chocolate bars", 2);

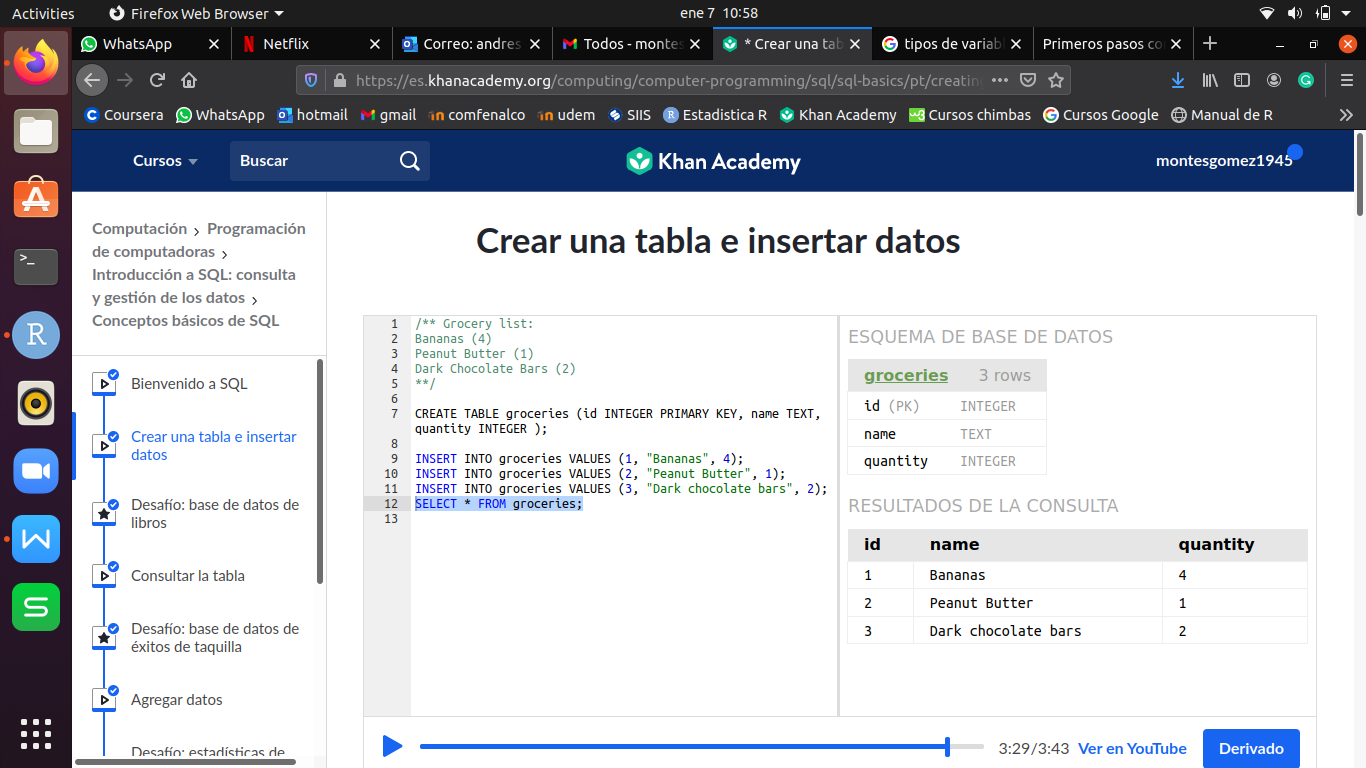


1. **VER TABLA COMPLETA (SELECT)**

SELECT \* FROM nombre\_de\_tabla;

**EJEMPLO:**

SELECT \* FROM groceries;

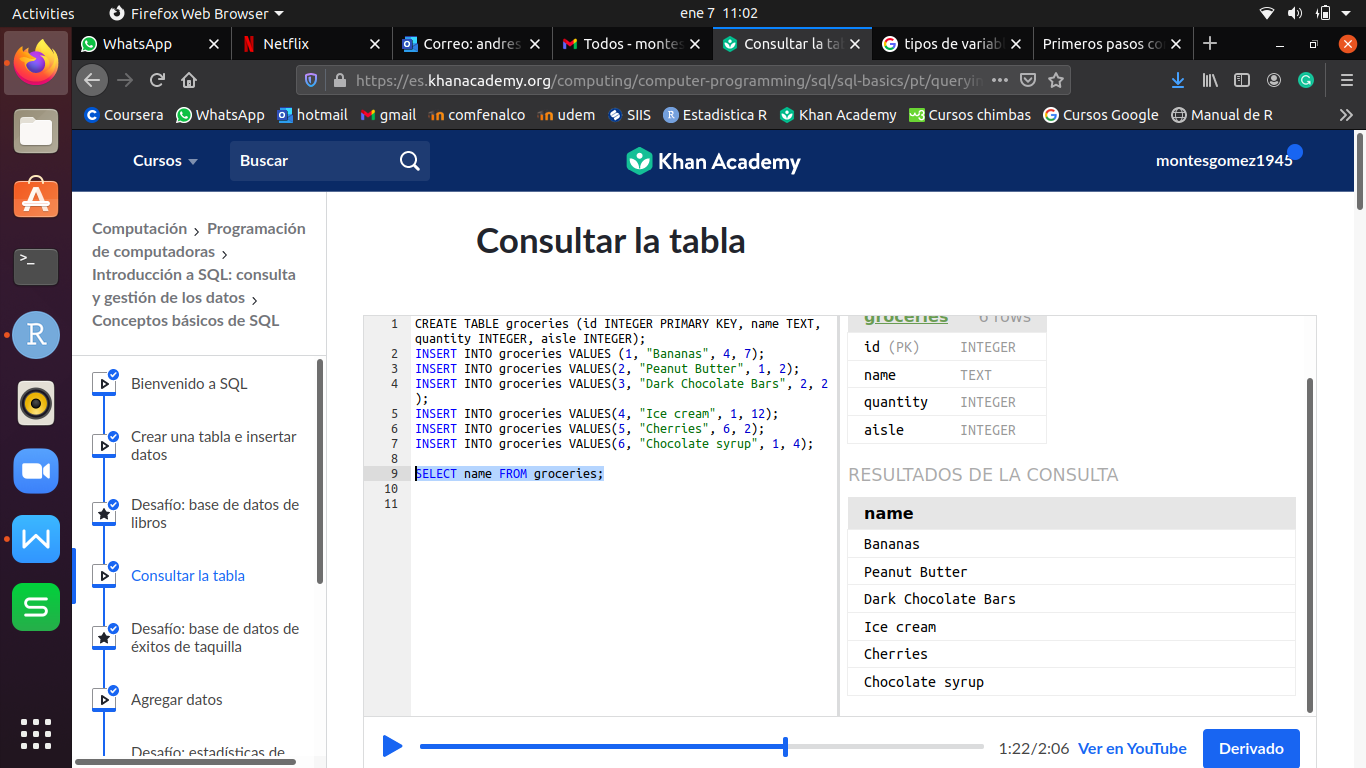


1. **VER UNA COLUMNA DETERMINADA**

SELECT nombre\_de\_columna FROM nombre\_de\_tabla;

**EJEMPLO:**

SELECT name FROM groceries;

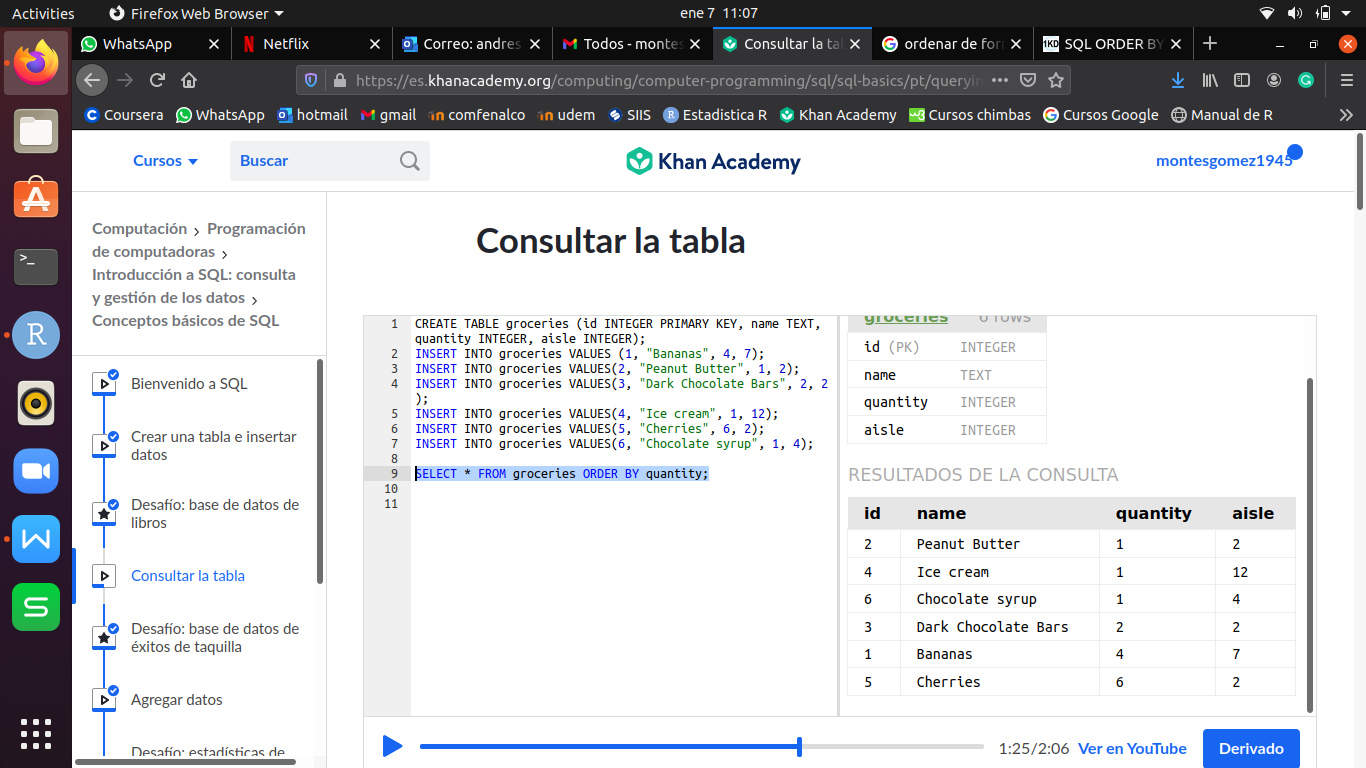


1. **ORDENAR TABLA DE ACUERDO A UNA COLUMNA (ORDER BY)**

SELECT \* FROM nombre\_de\_tabla ORDER BY nombre\_de\_columna DESC ó ASC;

**EJEMPLO:**

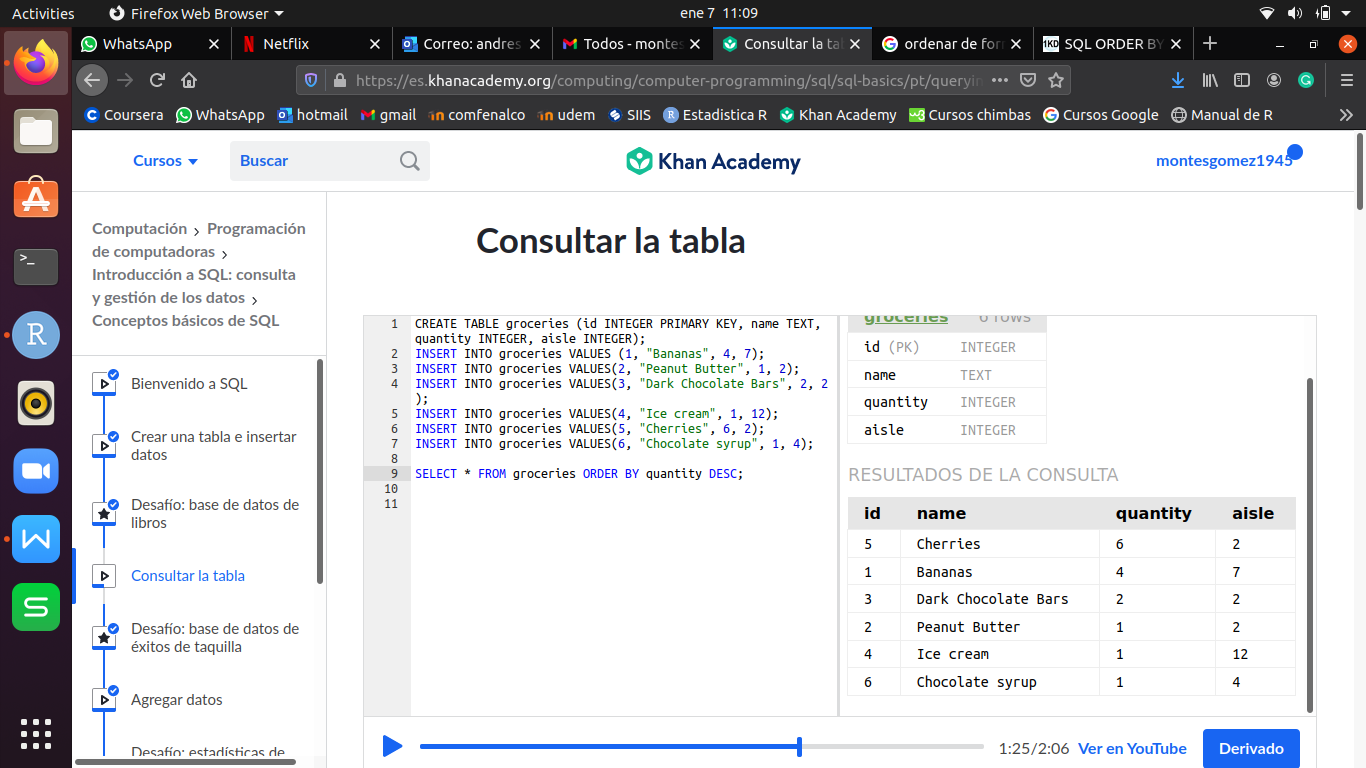
SELECT \* FROM groceries ORDER BY quantity;



- Por defecto, SQL ordena de forma ascendente, si se desea de forma descendente se debe agregar al final **DESC.**

**EJEMPLO:**

SELECT \* FROM groceries ORDER BY quantity DESC;



1. **FILTRAR RESULTADO (WHERE)**

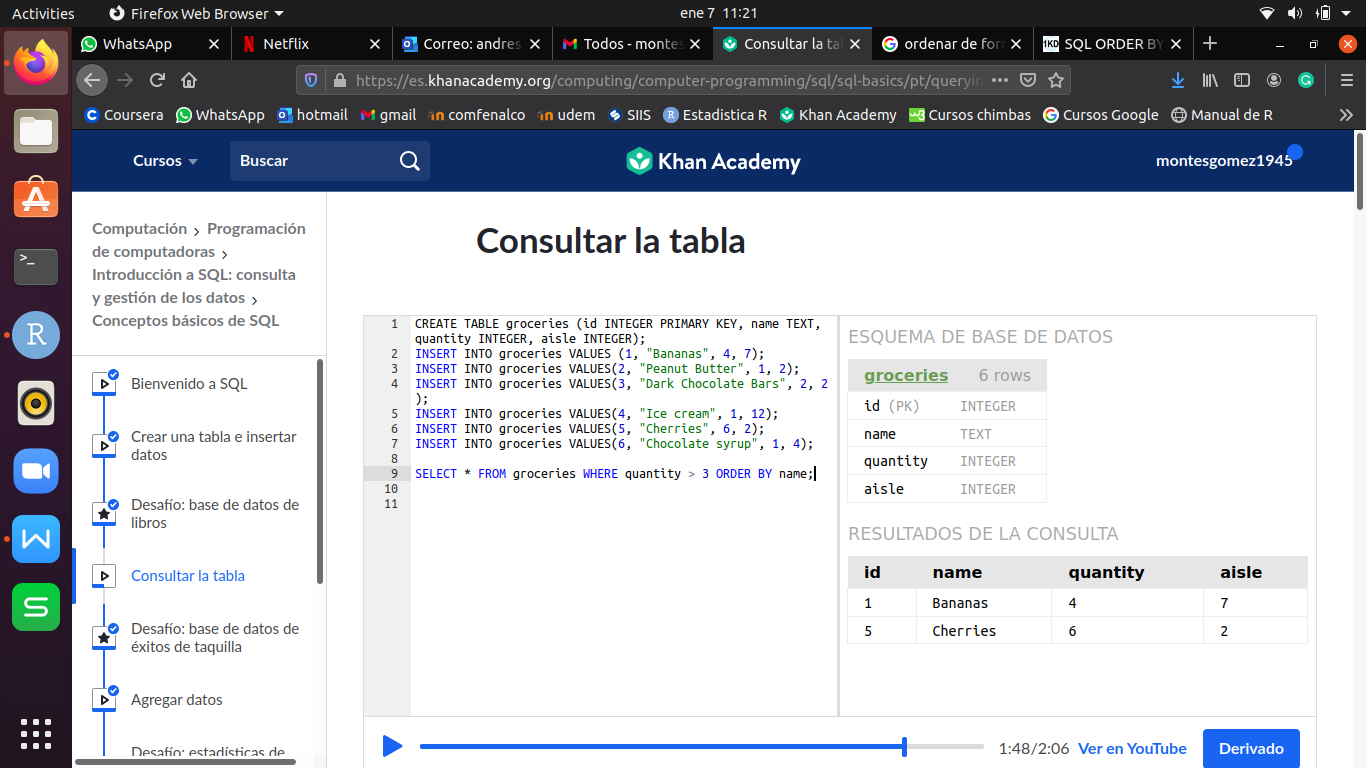
... WHERE operador 5, 3.2, ‘text’, ....

- El operador puede ser >, <, =, >=, <=, <>, !=, LIKE, BETWEEN, IS NOT, IS NULL, IN, NOT IN.

- El ORDER BY siempre debe ir al final.

**EJEMPLO:**

SELECT \* FROM groceries WHERE quantity > 3 ORDER BY name;



- También es posible aplicar filtros con varias condiciones con los operadores lógicos **OR, AND,** o **NOT.**

**EJEMPLO:**

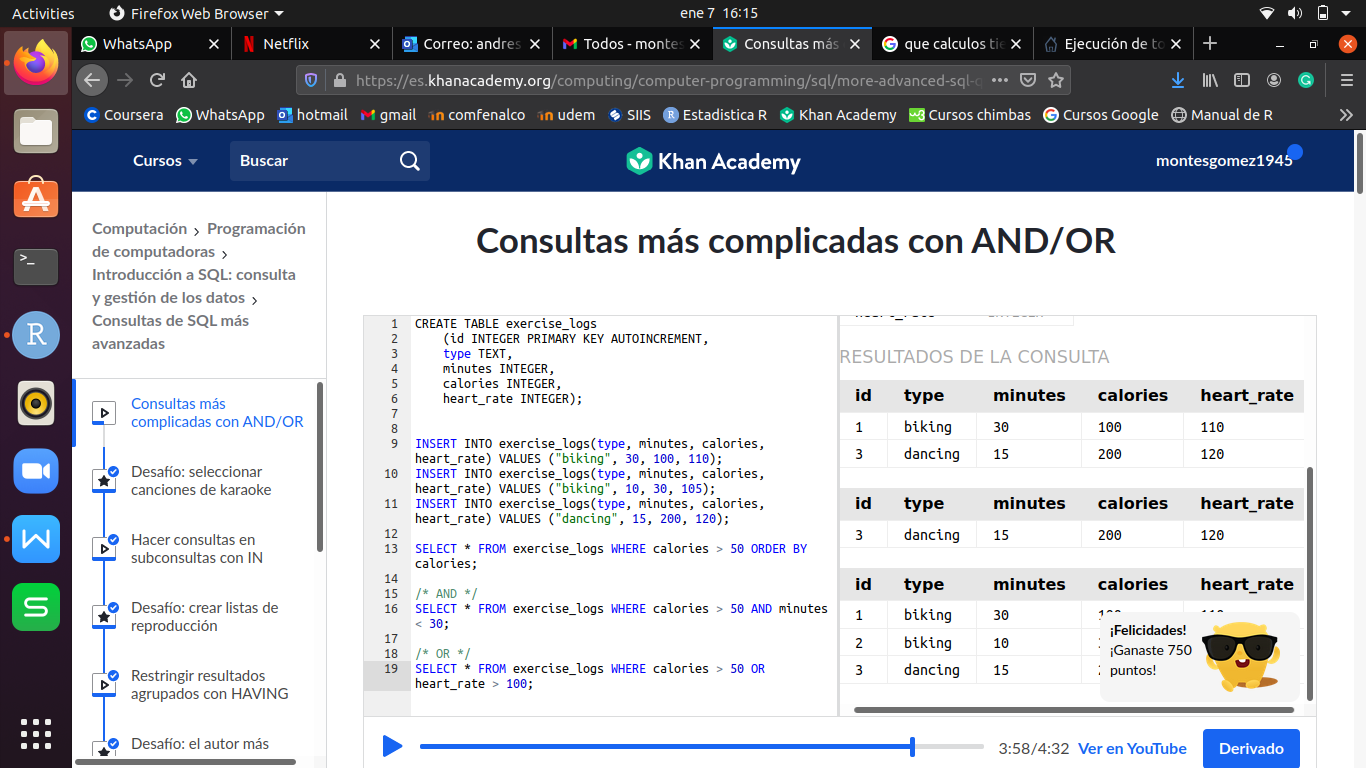
CREATE TABLE exercise\_logs (id INTEGER PRIMARY KEY AUTOINCREMENT, type TEXT, minutes INTEGER, calories INTEGER, heart\_rate INTEGER);

INSERT INTO exercise\_logs(type, minutes, calories, heart\_rate) VALUES ("biking", 30, 100, 110);

INSERT INTO exercise\_logs(type, minutes, calories, heart\_rate) VALUES ("biking", 10, 30, 105);

INSERT INTO exercise\_logs(type, minutes, calories, heart\_rate) VALUES ("dancing", 15, 200, 120);

SELECT \* FROM exercise\_logs WHERE calories > 50 AND minutes < 30;



SELECT \* FROM exercise\_logs WHERE calories > 50 OR heart\_rate > 100;



1. **CÁLCULOS**

SELECT FUNCION(nom\_col) FROM nom\_tabla;

- La **FUNCIÓN** puede ser: **SUM**(Suma), **AVG**(Promedio), **MAX**(Maximo), **MIN**(Minimo), **COUNT**(contar).

INSERT INTO groceries VALUES (1, "Bananas", 56, 7);

INSERT INTO groceries VALUES(2, "Peanut Butter", 1, 2);

INSERT INTO groceries VALUES(3, "Dark Chocolate Bars", 2, 2);

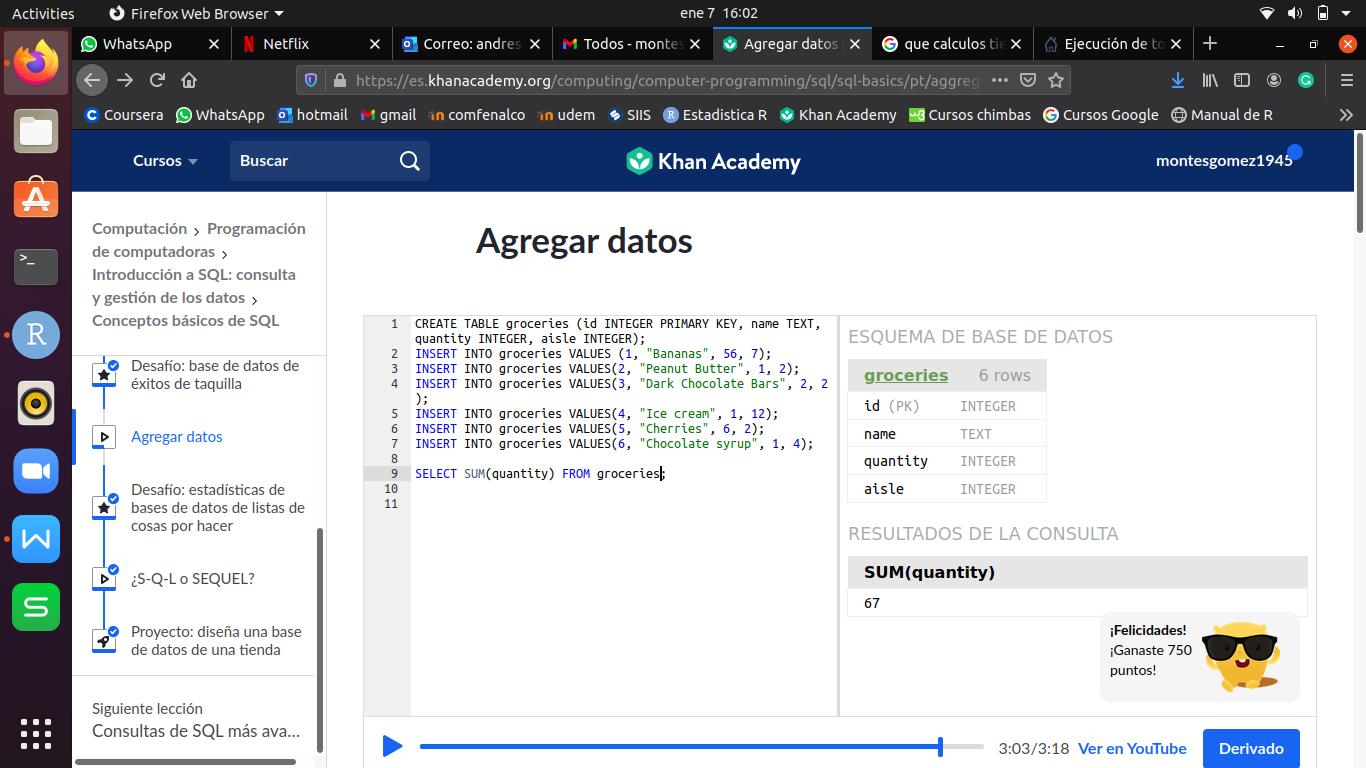
INSERT INTO groceries VALUES(4, "Ice cream", 1, 12);

INSERT INTO groceries VALUES(5, "Cherries", 6, 2);

INSERT INTO groceries VALUES(6, "Chocolate syrup", 1, 4);

**EJEMPLO:**

SELECT aisle, SUM(quantity) FROM groceries GROUP BY aisle;

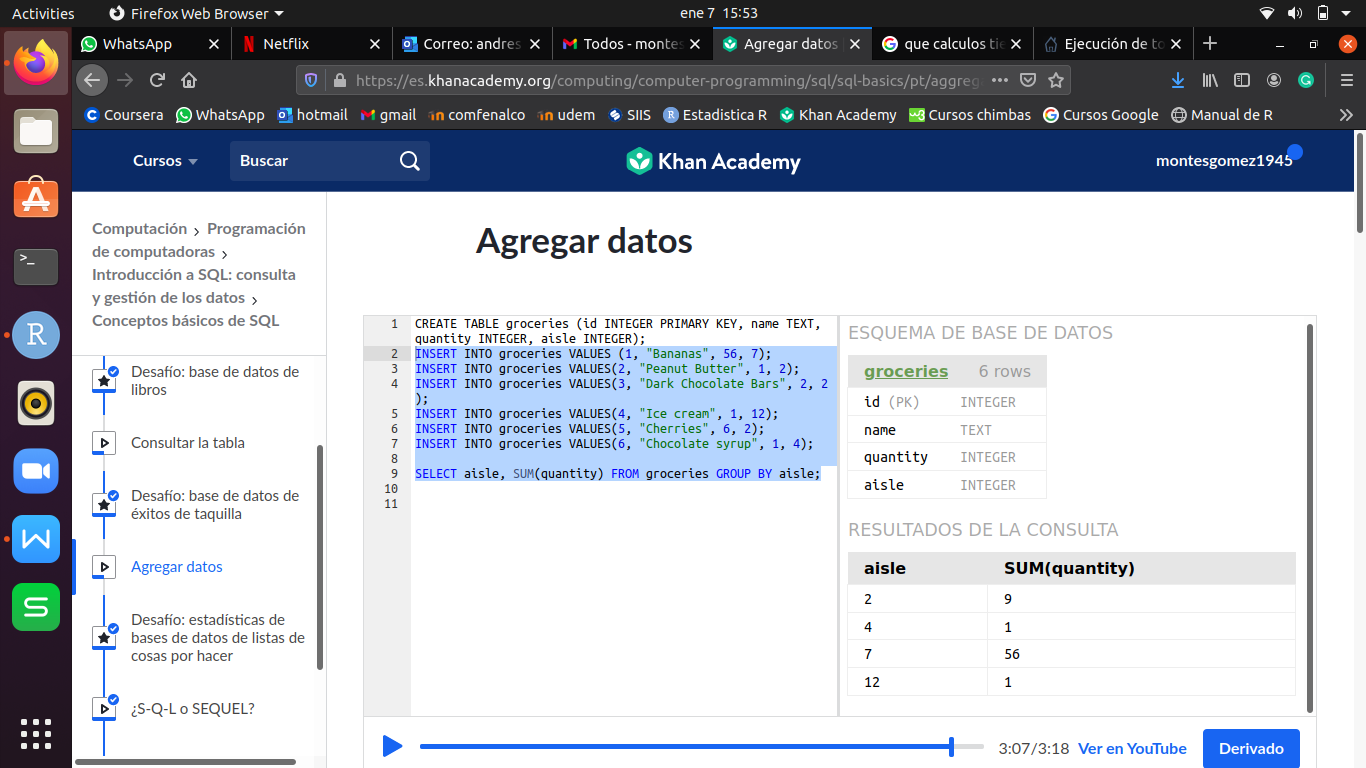


1. **CÁLCULOS CON FILTROS**

SELECT ***nombre\_columna\_a\_filtrar*** FUNCIÓN (nombre\_columna\_numérica) FROM nombre\_tabla GROUP BY ***nombre\_columna\_a\_filtrar***

**EJEMPLO:**

SELECT **aisle**, SUM(quantity) FROM groceries GROUP BY **aisle**;



1. **APLICAR VARIOS FILTROS: SUBCONSULTAS (IN, NOT IN)**

SELECT \* FROM nombre\_tabla WHERE nombre\_columna\_numérica IN (‘texto1’, ‘texto2’, ...);

**EJEMPLO:**

CREATE TABLE exercise\_logs (id INTEGER PRIMARY KEY AUTOINCREMENT, type TEXT, minutes INTEGER, calories INTEGER, heart\_rate INTEGER);

INSERT INTO exercise\_logs(type, minutes, calories, heart\_rate) VALUES ("biking", 30, 100, 110);

INSERT INTO exercise\_logs(type, minutes, calories, heart\_rate) VALUES ("biking", 10, 30, 105);

INSERT INTO exercise\_logs(type, minutes, calories, heart\_rate) VALUES ("dancing", 15, 200, 120);

INSERT INTO exercise\_logs(type, minutes, calories, heart\_rate) VALUES ("tree climbing", 30, 70, 90);

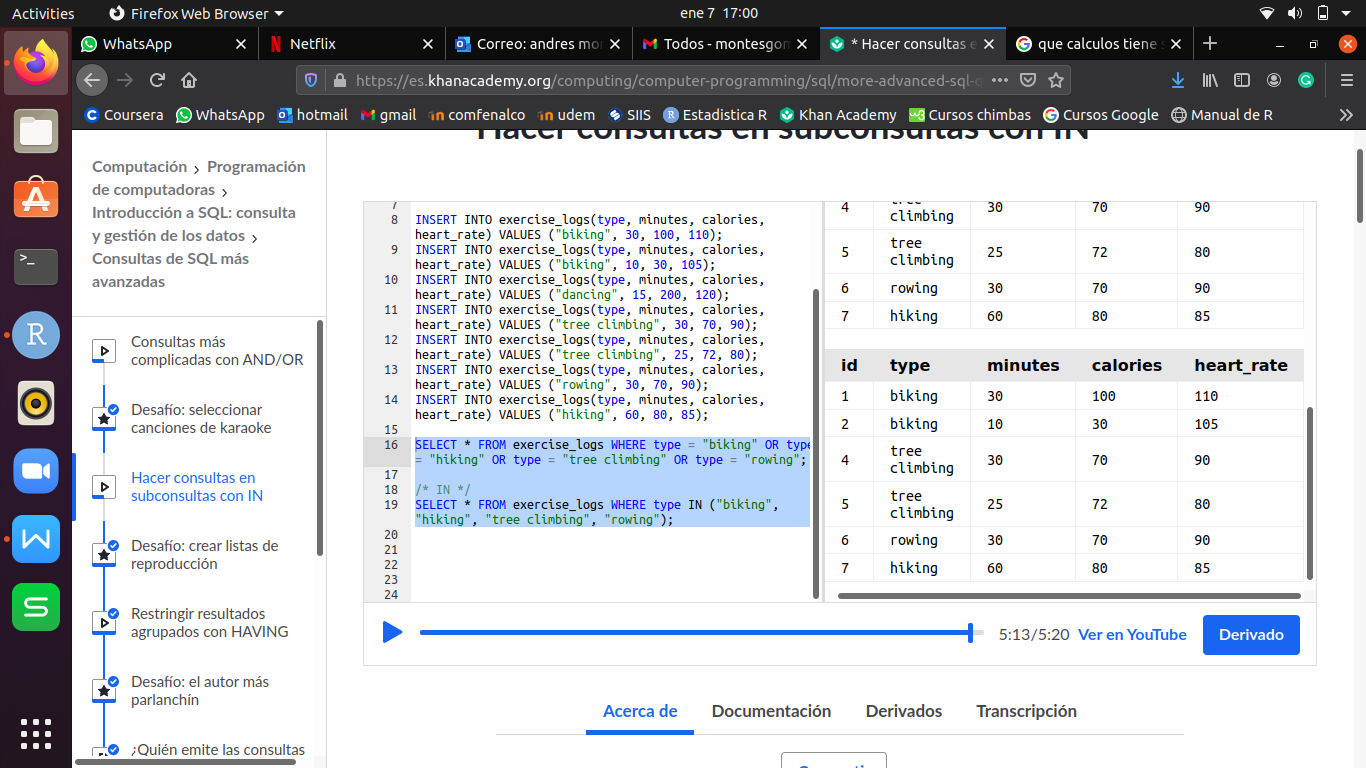
INSERT INTO exercise\_logs(type, minutes, calories, heart\_rate) VALUES ("tree climbing", 25, 72, 80);

INSERT INTO exercise\_logs(type, minutes, calories, heart\_rate) VALUES ("rowing", 30, 70, 90);

INSERT INTO exercise\_logs(type, minutes, calories, heart\_rate) VALUES ("hiking", 60, 80, 85);

SELECT \* FROM exercise\_logs WHERE type = "biking" OR type = "hiking" OR type = "tree climbing" OR type = "rowing";

SELECT \* FROM exercise\_logs WHERE type IN ("biking", "hiking", "tree climbing", "rowing");



- Como podemos ver, el **IN** se usa para no hacer varios **OR** y sea mas facil la consulta.

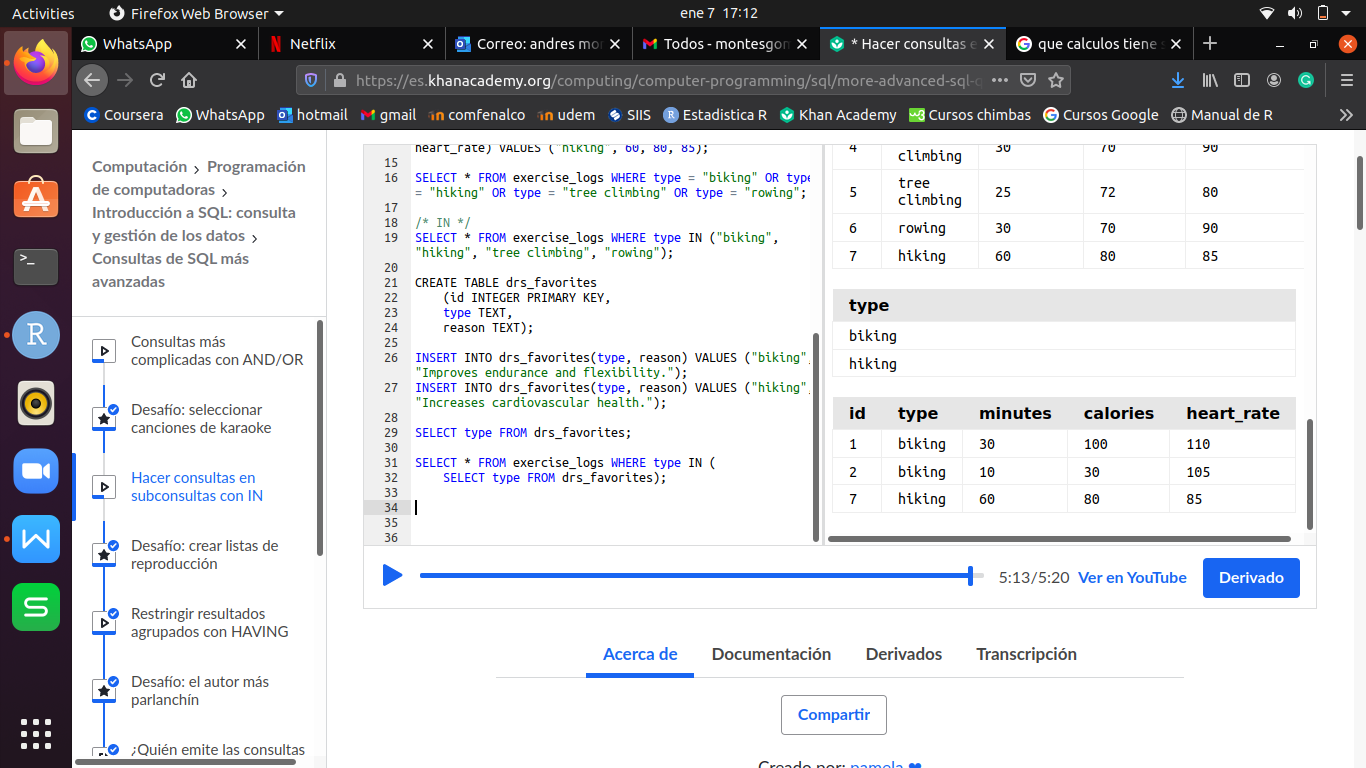
**EJEMPLO:**

CREATE TABLE drs\_favorites (id INTEGER PRIMARY KEY, type TEXT, reason TEXT);

INSERT INTO drs\_favorites(type, reason) VALUES ("biking", "Improves endurance and flexibility.");

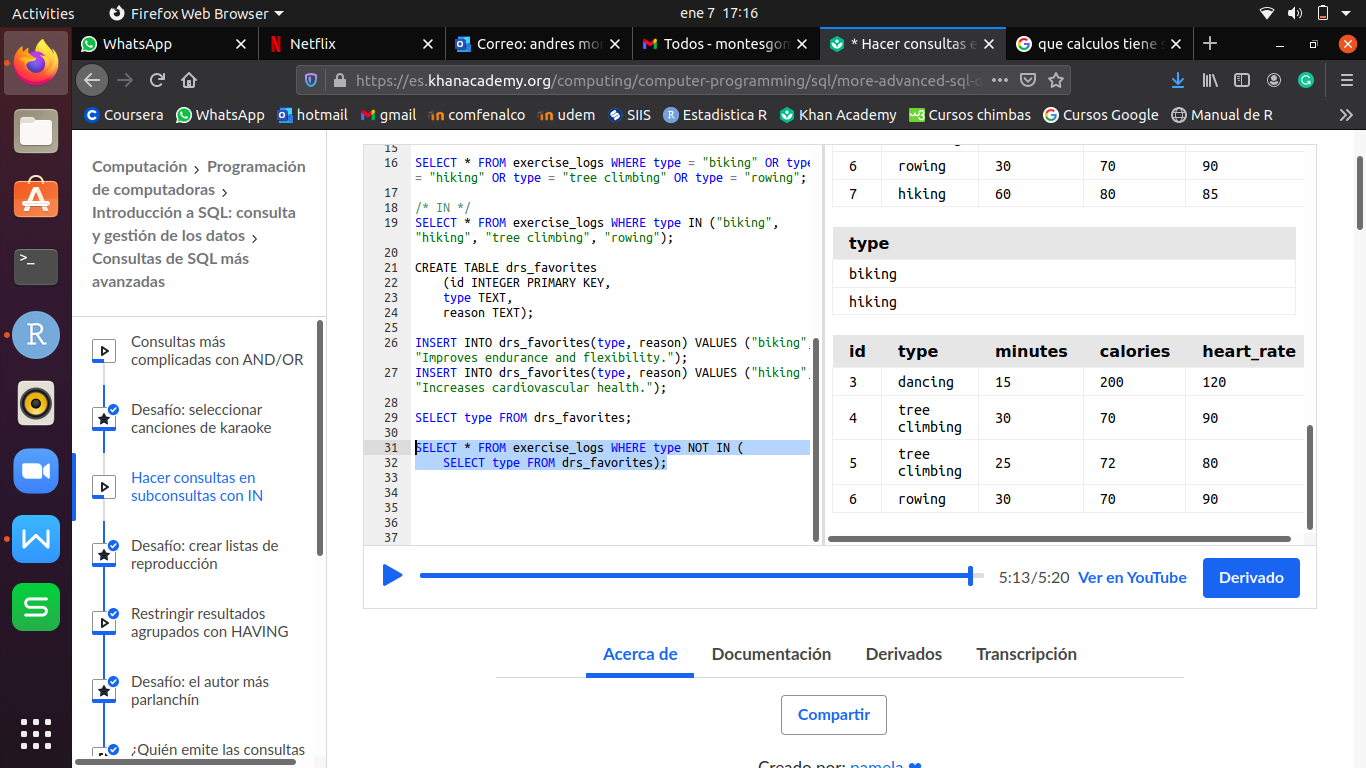
INSERT INTO drs\_favorites(type, reason) VALUES ("hiking", "Increases cardiovascular health.");

SELECT \* FROM exercise\_logs WHERE type IN (SELECT type FROM drs\_favorites);



**EJEMPLO:**

SELECT \* FROM exercise\_logs WHERE type NOT IN (SELECT type FROM drs\_favorites);



1. **CONSULTAS QUE CONTENGAN UN VALOR DESEADO (LIKE)**

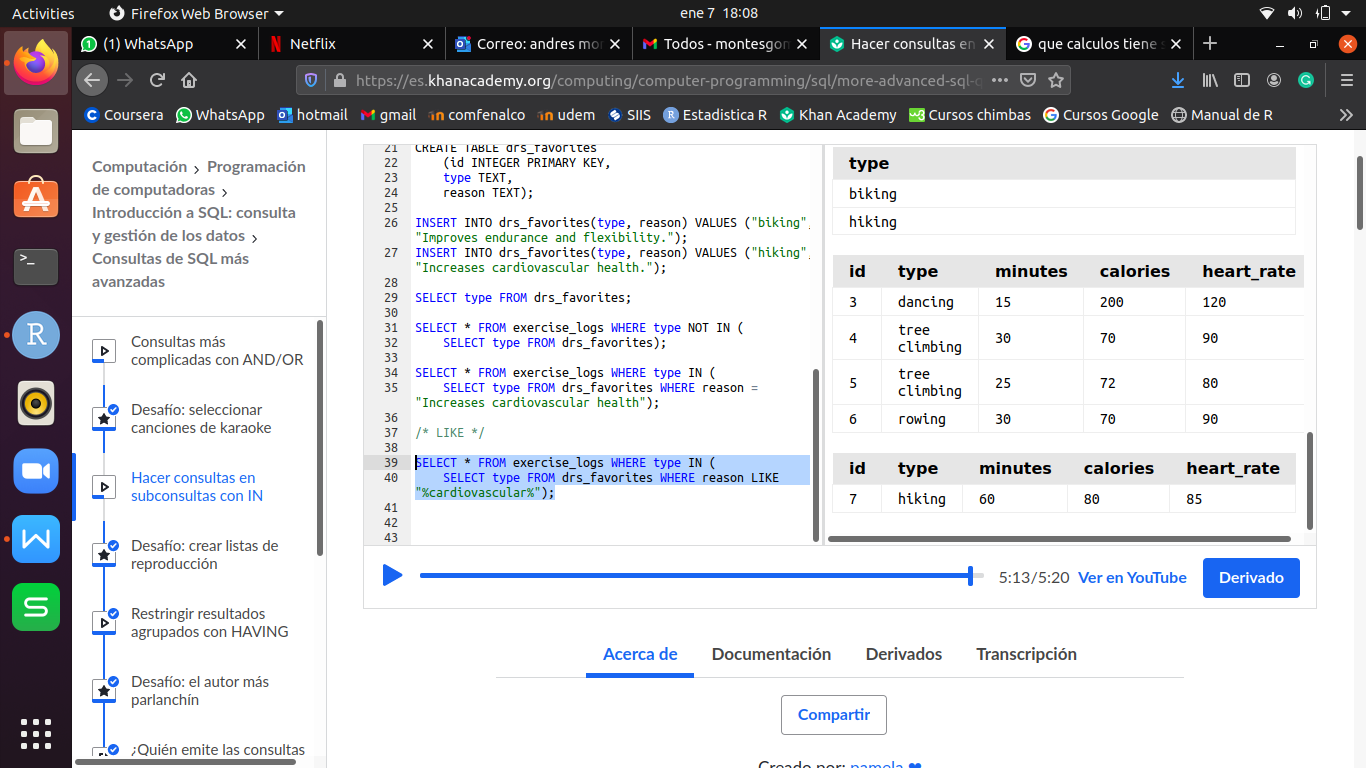
SELECT \* FROM nombre\_tabla1 WHERE nombre\_coumnal IN(SELECT nombre\_columna FROM nombre\_tabla2 WHERE nombre\_columna\_buscada LIKE ‘%valor\_bucado%);

- La segunda parte del **SELECT** es otra tabla relacionada por nombre\_columna.

- Después del **IN** va el filtro deseado.

**EJEMPLO:**

SELECT \* FROM exercise\_logs WHERE type IN (SELECT type FROM drs\_favorites WHERE reason LIKE "%cardiovascular%");



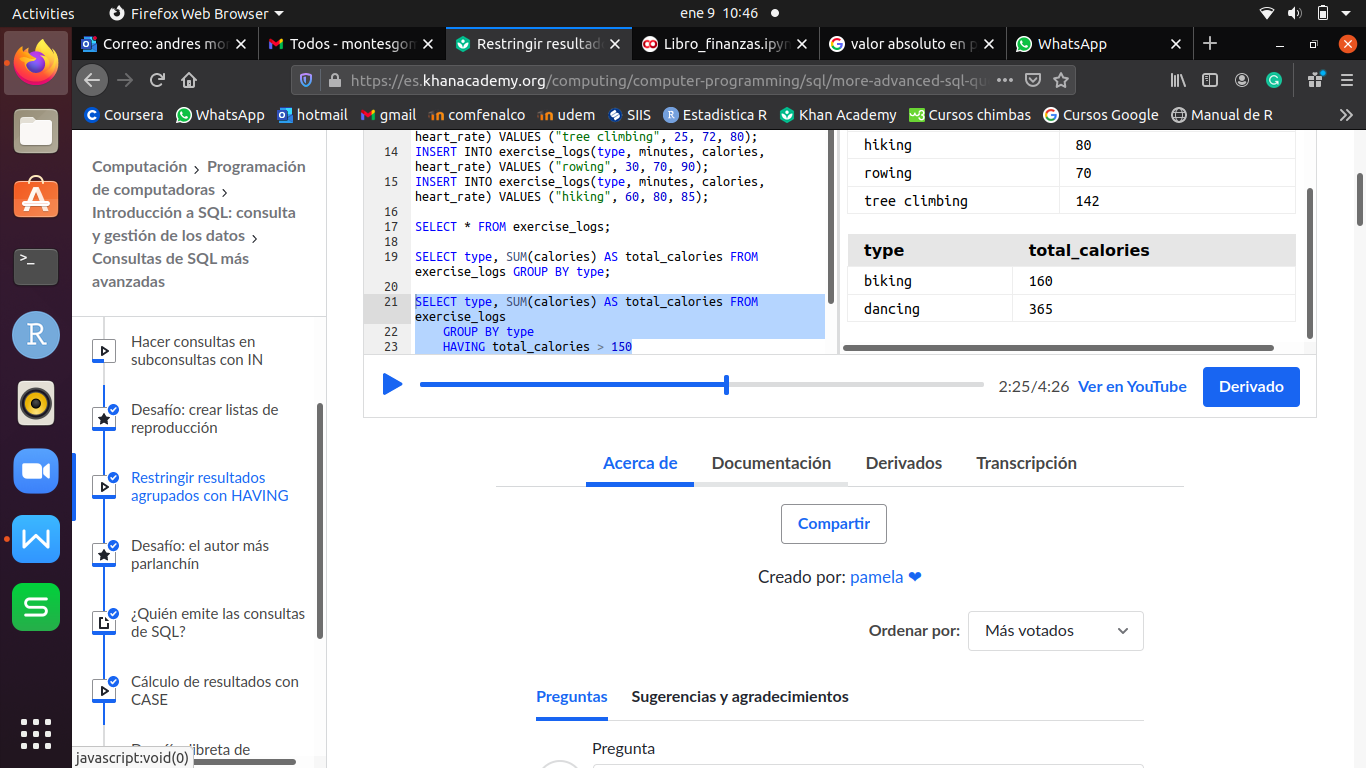
1. **HACER FILTRO A UNA OPERACIÓN (HAVING)**

SELECT nombre\_columna FUNCION(com\_columna\_numérica) AS nombre\_columna\_resultado FROM nombre\_tabla GROUP BY nombre\_columna HAVING nombre\_resultado;

**EJEMPLO:**

SELECT type, SUM(calories) AS total\_calories FROM exercise\_logs

GROUP BY type HAVING total\_calories > 150



1. **CASE**

SELECT OPERACION(\*)

CASE

WHEN nom\_col\_num **operador** # u operacion THEN ‘texto deseado1’

WHEN nom\_col\_num **operador** # u operacion THEN ‘texto deseado2’ ...

ELSE ‘texto si no cumple las condiciones’

END AS ‘nom\_col\_resultado’

FROM nom\_tabla GROUP BY nom\_col\_resultado;

- El **CASE** es el equivalente al **SWITCH** en programación.

- Con el **CASE** se crea una nueva columna ‘nombre\_columna\_resultado’ dependiendo de los rangos dados (**WHERE; ELSE**). Con esta nueva columna creada es más sencillo hacer la consulta.

**EJEMPLO:**

SELECT COUNT(\*),

CASE

WHEN heart\_rate > 220-30 THEN "above max"

WHEN heart\_rate > ROUND(0.90 \* (220-30)) THEN "above target"

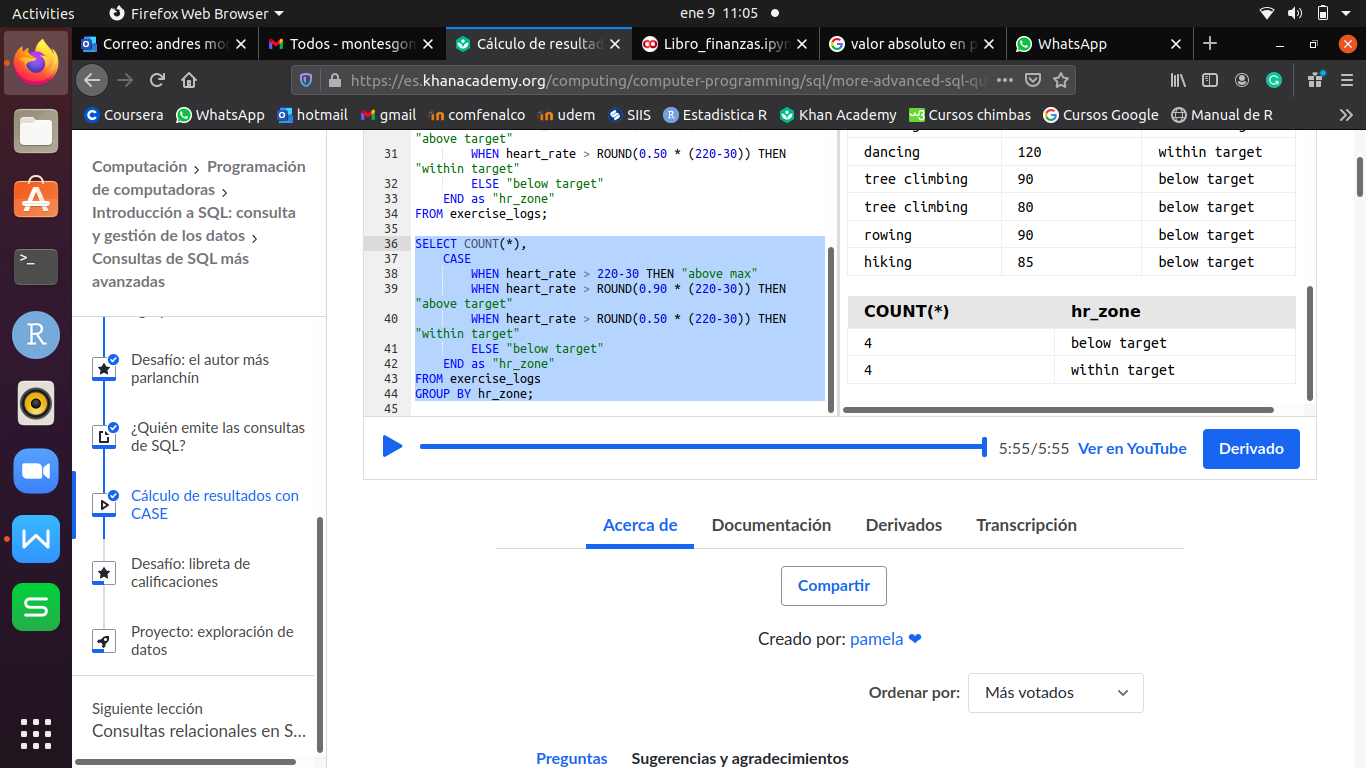
WHEN heart\_rate > ROUND(0.50 \* (220-30)) THEN "within target"

ELSE "below target"

END as "hr\_zone"

FROM exercise\_logs

GROUP BY hr\_zone;



1. **EXPLICIT INNER JOIN**

SELECT nombre\_tabla.nombre\_columna1, nombre\_tabla.nombre\_col2, ... FROM tabla1

JOIN tabla2 ON tabla1.id(PK) = tabla2.col\_id;

- Es importante colocar al inicio la tabla (nombre\_tabla) de donde se va a tomar la columna (nombre\_columna) ya que es posible que la otra tabla tenga columnas con el mismo nombre.

**EJEMPLO:**

CREATE TABLE students (id INTEGER PRIMARY KEY, first\_name TEXT, last\_name TEXT, email TEXT, phone TEXT, birthdate TEXT);

INSERT INTO students (first\_name, last\_name, email, phone, birthdate)

VALUES ("Peter", "Rabbit", "peter@rabbit.com", "555-6666", "2002-06-24");

INSERT INTO students (first\_name, last\_name, email, phone, birthdate)

VALUES ("Alice", "Wonderland", "alice@wonderland.com", "555-4444", "2002-07-04");

CREATE TABLE student\_grades (id INTEGER PRIMARY KEY, student\_id INTEGER, test TEXT, grade INTEGER);

INSERT INTO student\_grades (student\_id, test, grade) VALUES (1, "Nutrition", 95);

INSERT INTO student\_grades (student\_id, test, grade) VALUES (2, "Nutrition", 92);

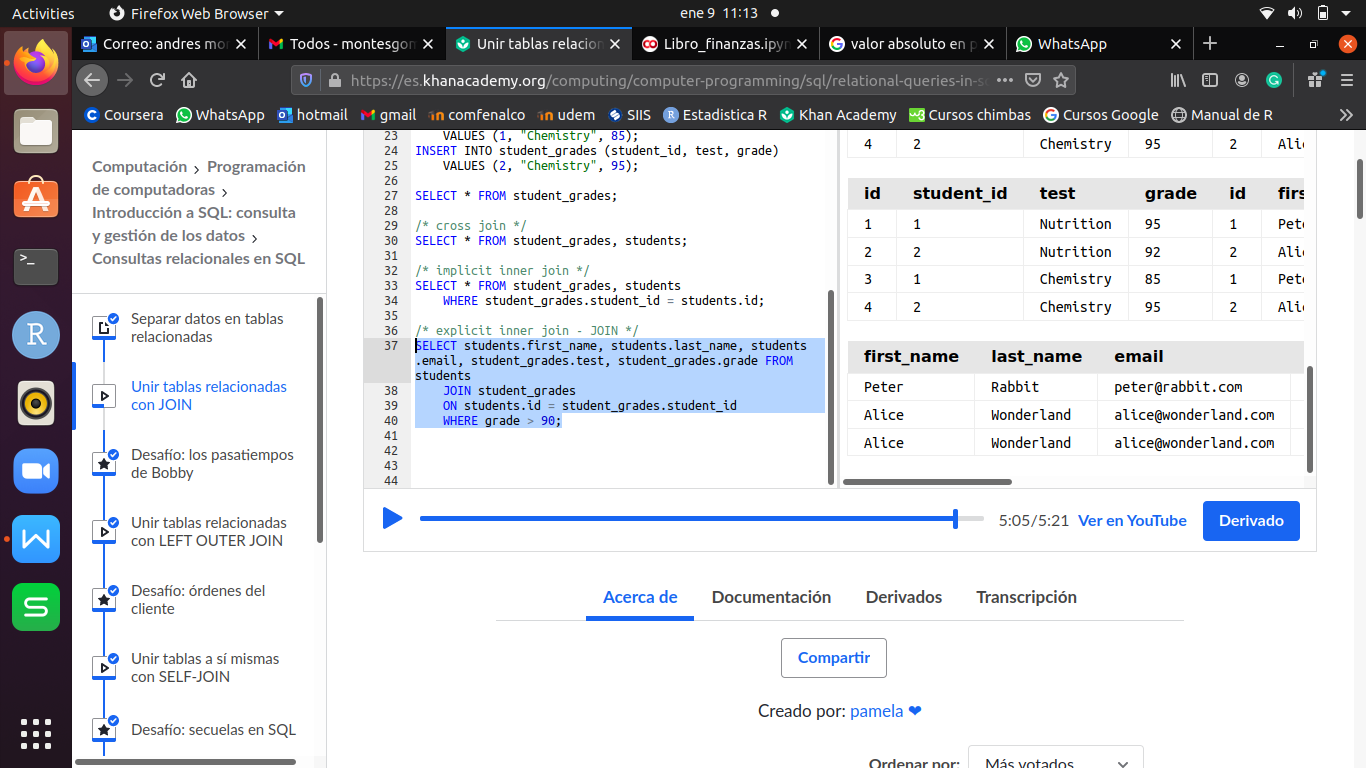
INSERT INTO student\_grades (student\_id, test, grade) VALUES (1, "Chemistry", 85);

INSERT INTO student\_grades (student\_id, test, grade) VALUES (2, "Chemistry", 95);

SELECT students.first\_name, students.last\_name, students.email, student\_grades.test, student\_grades.grade FROM students

JOIN student\_grades ON students.id = student\_grades.student\_id

WHERE grade > 90;



1. **LEFT OUTER JOIN**

SELECT nombre\_columna1, nombre\_columna2, ... FROM tabla1 LEFT OUTER JOIN tabla2

ON tabla1.id(PK) = tabla2.columna\_id;

- El **LEFT OUTER JOIN** se utiliza cuando se quiere muestrar todos los valores de las columnas seleccionadas, así estas no tengan registro en la otra tabla.

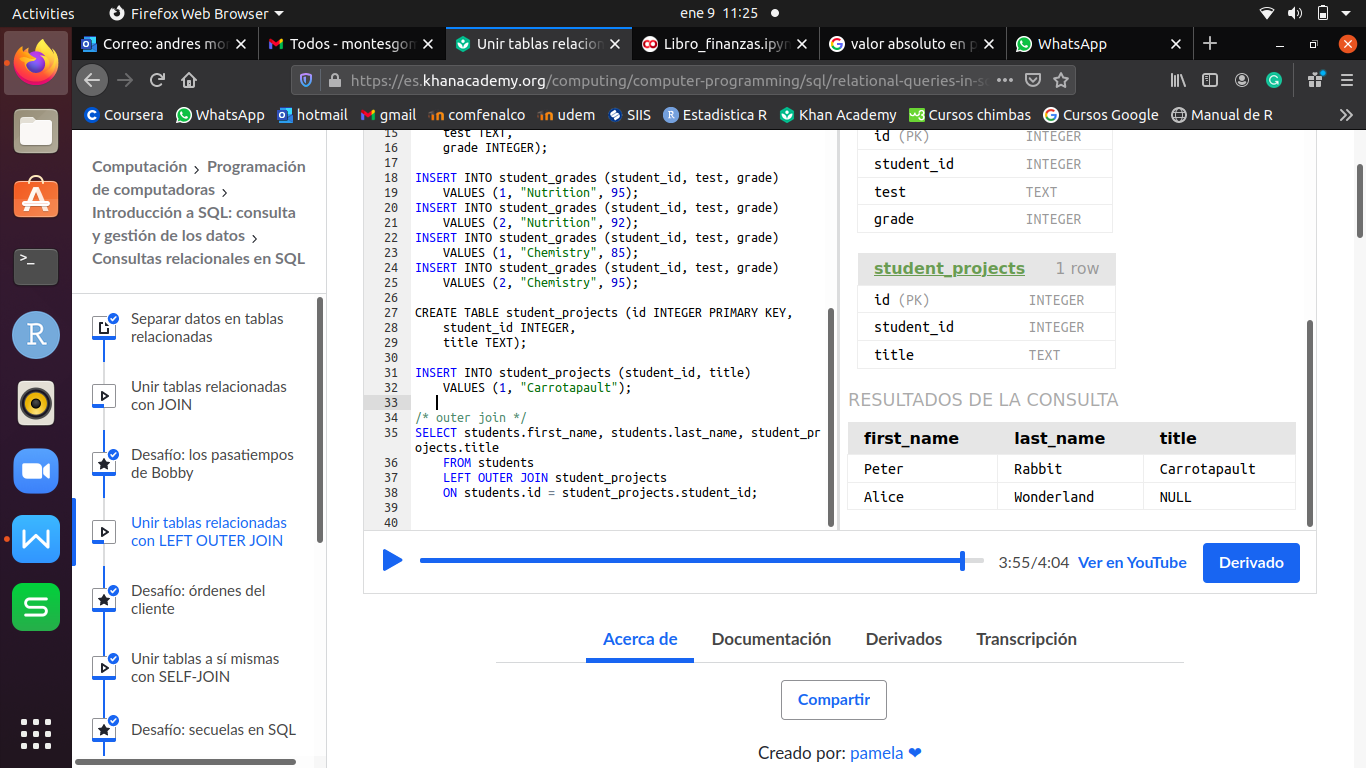
**EJEMPLO:**

CREATE TABLE student\_projects (id INTEGER PRIMARY KEY, student\_id INTEGER, title TEXT);

INSERT INTO student\_projects (student\_id, title) VALUES (1, "Carrotapault");

SELECT students.first\_name, students.last\_name, student\_projects.title FROM students LEFT OUTER JOIN student\_projects

ON students.id = student\_projects.student\_id;



1. **SELF JOIN**

SELECT nombre\_columna1, nombre\_columna2, ..., AS alias.nombre\_columna AS nombre\_com\_deseada FROM tabla JOIN tabla alias

ON tabla.columna\_id = alias.id(PK);

- Se utliza cuando se desea relacionar columnas de una misma tabla.

**EJEMPLO:**

CREATE TABLE students (id INTEGER PRIMARY KEY AUTOINCREMENT, first\_name TEXT, last\_name TEXT, email TEXT, phone TEXT, birthdate TEXT,

buddy\_id INTEGER);

INSERT INTO students VALUES (1, "Peter", "Rabbit", "peter@rabbit.com", "555-6666", "2002-06-24", 2);

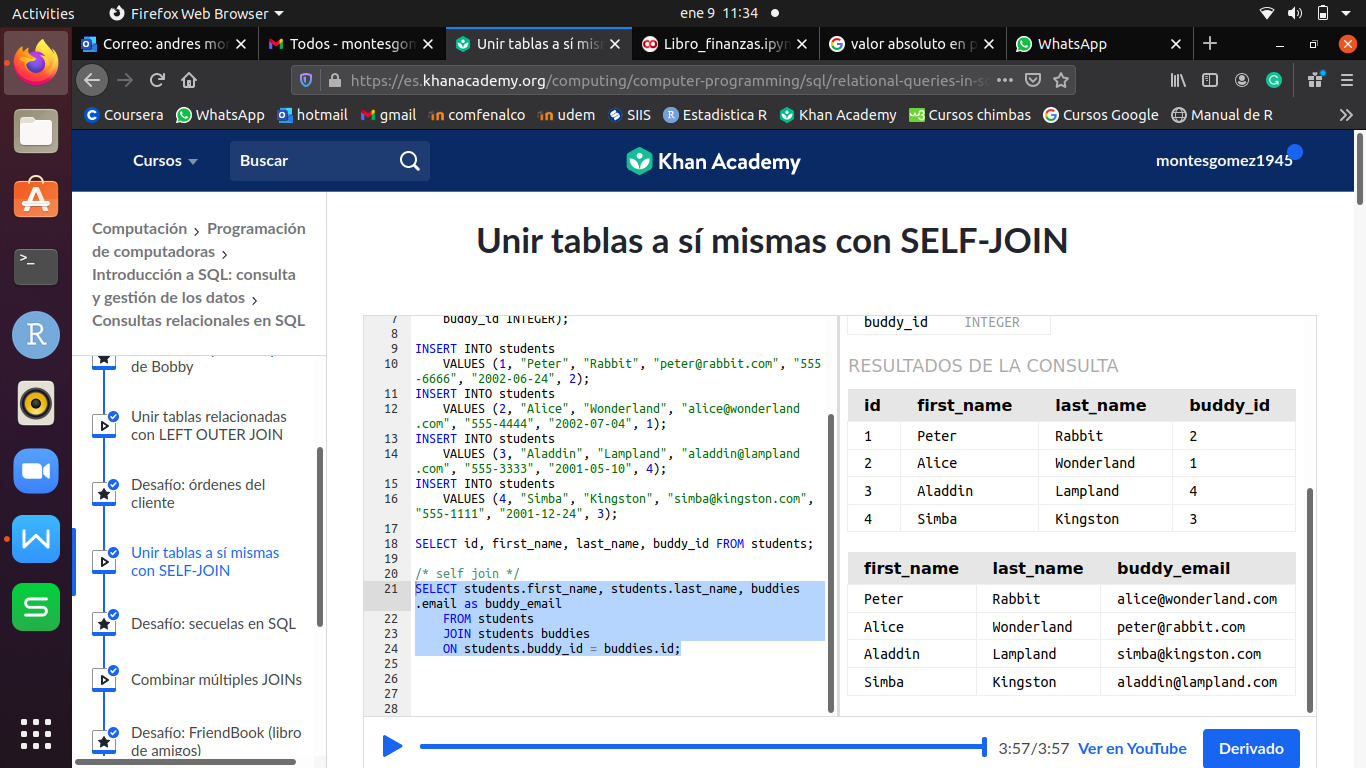
INSERT INTO students VALUES (2, "Alice", "Wonderland", "alice@wonderland.com", "555-4444", "2002-07-04", 1);

INSERT INTO students VALUES (3, "Aladdin", "Lampland", "aladdin@lampland.com", "555-3333", "2001-05-10", 4);

INSERT INTO students VALUES (4, "Simba", "Kingston", "simba@kingston.com", "555-1111", "2001-12-24", 3);

SELECT students.first\_name, students.last\_name, buddies.email as buddy\_email FROM students JOIN students buddies

ON students.buddy\_id = buddies.id;



1. **MULTIPLES JOIN**

SELECT a.nombre\_columna, b.nombre\_columna FROM tabla2

JOIN tabla1 a ON tabla2.columna\_id = a.id(PK)

JOIN tabla1 b ON tabla2.columna\_id = b.id(PK)

**EJEMPLO:**

CREATE TABLE students (id INTEGER PRIMARY KEY, first\_name TEXT, last\_name TEXT, email TEXT, phone TEXT, birthdate TEXT);

INSERT INTO students (first\_name, last\_name, email, phone, birthdate)

VALUES ("Peter", "Rabbit", "peter@rabbit.com", "555-6666", "2002-06-24");

INSERT INTO students (first\_name, last\_name, email, phone, birthdate)

VALUES ("Alice", "Wonderland", "alice@wonderland.com", "555-4444", "2002-07-04");

INSERT INTO students (first\_name, last\_name, email, phone, birthdate)

VALUES ("Aladdin", "Lampland", "aladdin@lampland.com", "555-3333", "2001-05-10");

INSERT INTO students (first\_name, last\_name, email, phone, birthdate)

VALUES ("Simba", "Kingston", "simba@kingston.com", "555-1111", "2001-12-24");

CREATE TABLE student\_projects (id INTEGER PRIMARY KEY, student\_id INTEGER, title TEXT);

INSERT INTO student\_projects (student\_id, title) VALUES (1, "Carrotapault");

INSERT INTO student\_projects (student\_id, title) VALUES (2, "Mad Hattery");

INSERT INTO student\_projects (student\_id, title) VALUES (3, "Carpet Physics");

INSERT INTO student\_projects (student\_id, title) VALUES (4, "Hyena Habitats");

CREATE TABLE project\_pairs (id INTEGER PRIMARY KEY, project1\_id INTEGER, project2\_id INTEGER);

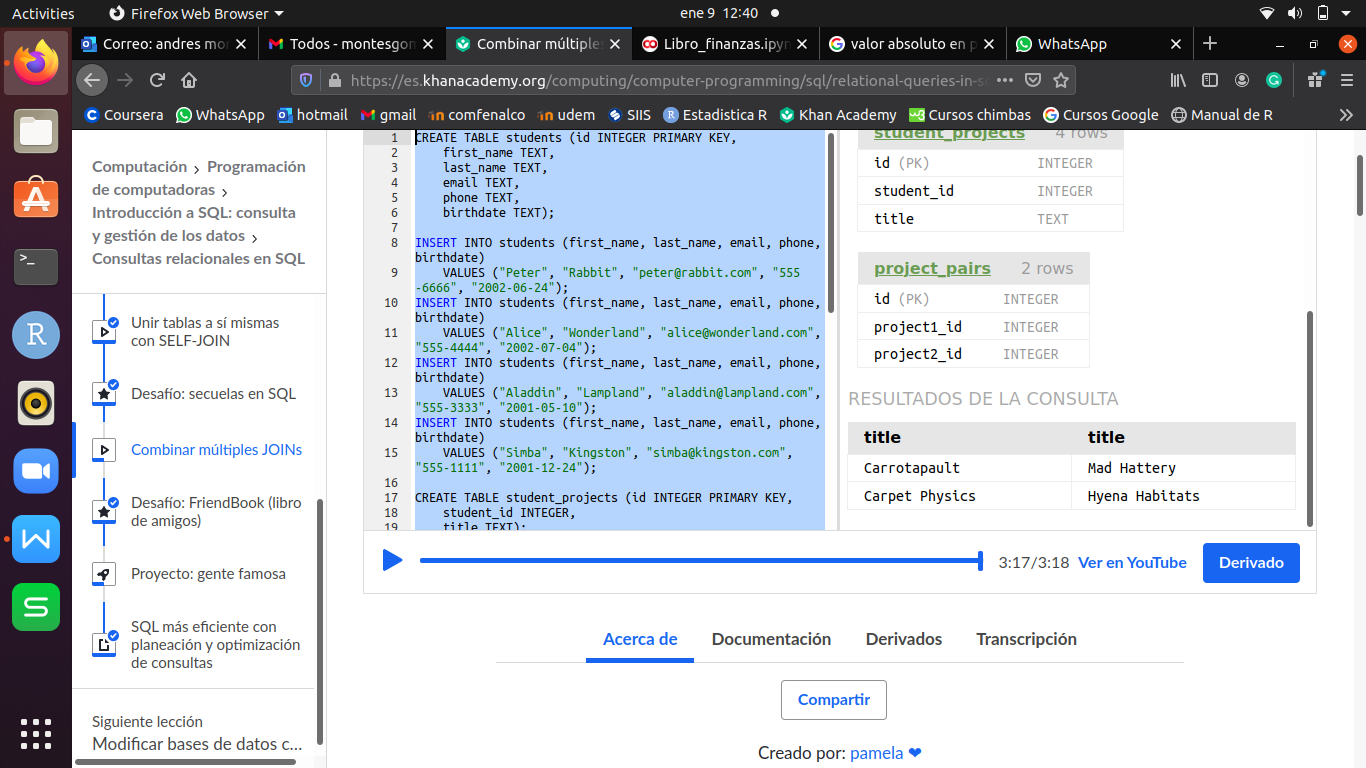
INSERT INTO project\_pairs (project1\_id, project2\_id) VALUES(1, 2);

INSERT INTO project\_pairs (project1\_id, project2\_id) VALUES(3, 4);

SELECT a.title, b.title FROM project\_pairs

JOIN student\_projects a ON project\_pairs.project1\_id = a.id

JOIN student\_projects b ON project\_pairs.project2\_id = b.id;



1. **ACTUALIZAR REGISTROS (UPDATE)**

UPDATE tabla SET nombre\_columna = ‘valor a actualizar’ WHERE nombre.columnax = #, o ‘char’ AND nombre\_columnay = #, o‘char’, o simplemente el id(PK) de conozcerlo.

**EJEMPLO:**

CREATE TABLE users (id INTEGER PRIMARY KEY, name TEXT);

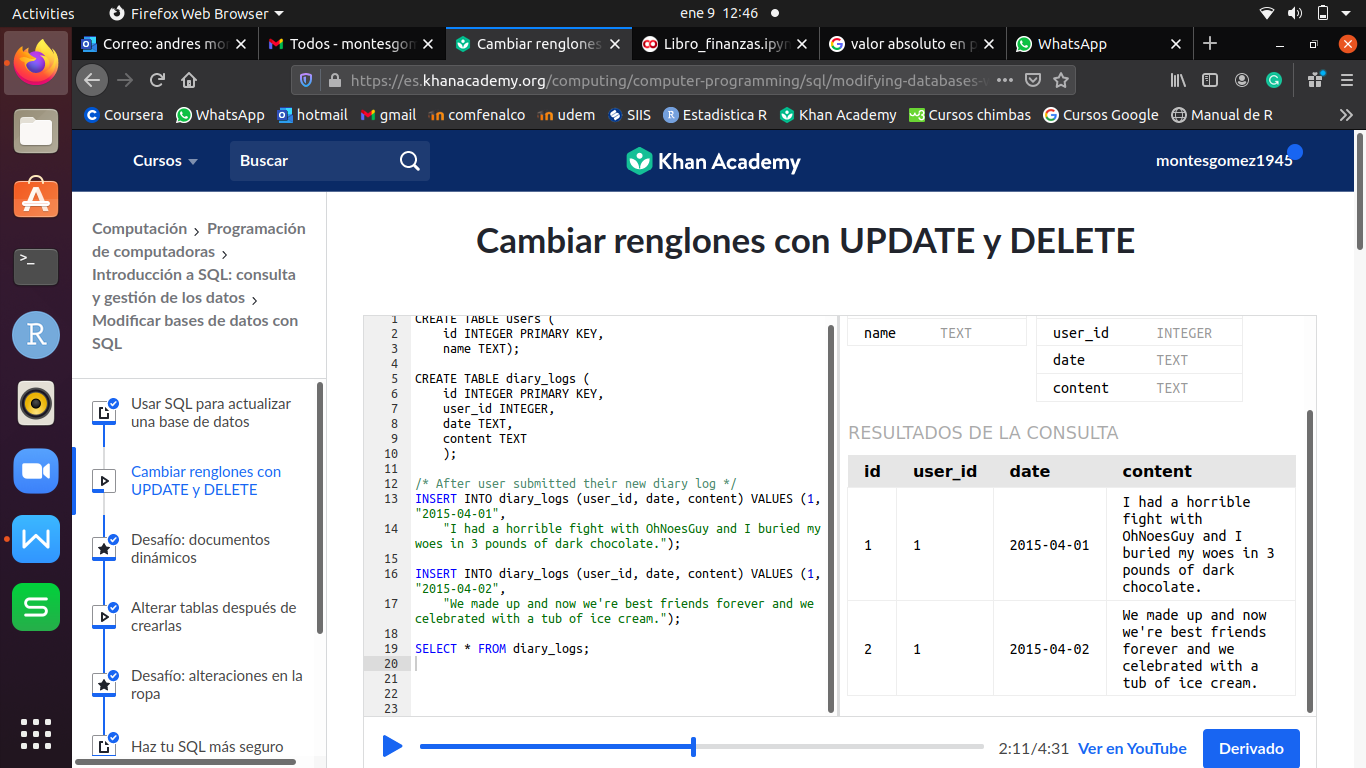
CREATE TABLE diary\_logs (id INTEGER PRIMARY KEY, user\_id INTEGER, date TEXT, content TEXT);

INSERT INTO diary\_logs (user\_id, date, content) VALUES (1, "2015-04-01",

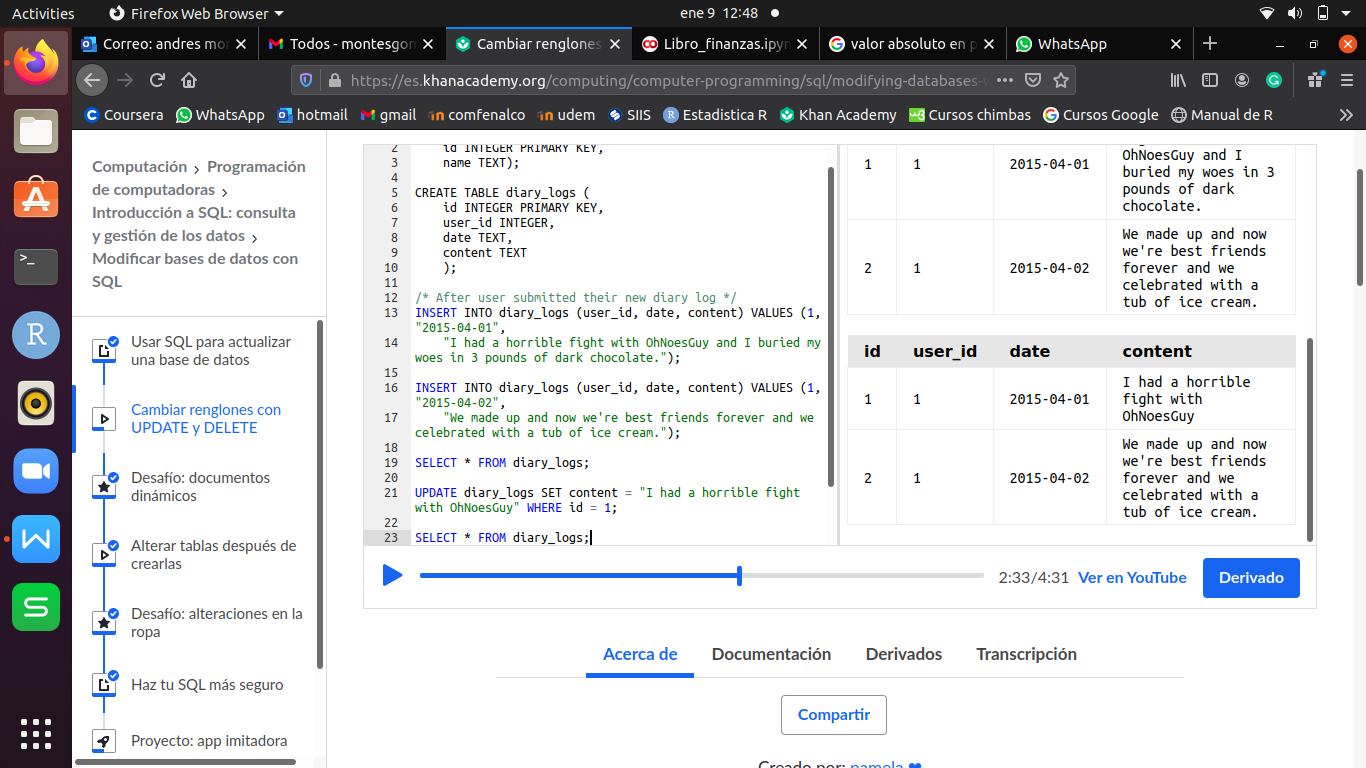
"I had a horrible fight with OhNoesGuy and I buried my woes in 3 pounds of dark chocolate.");

INSERT INTO diary\_logs (user\_id, date, content) VALUES (1, "2015-04-02",

"We made up and now we're best friends forever and we celebrated with a tub of ice cream.");



UPDATE diary\_logs SET content = "I had a horrible fight with OhNoesGuy" WHERE user\_id=1 AND date = "2015-04-01";

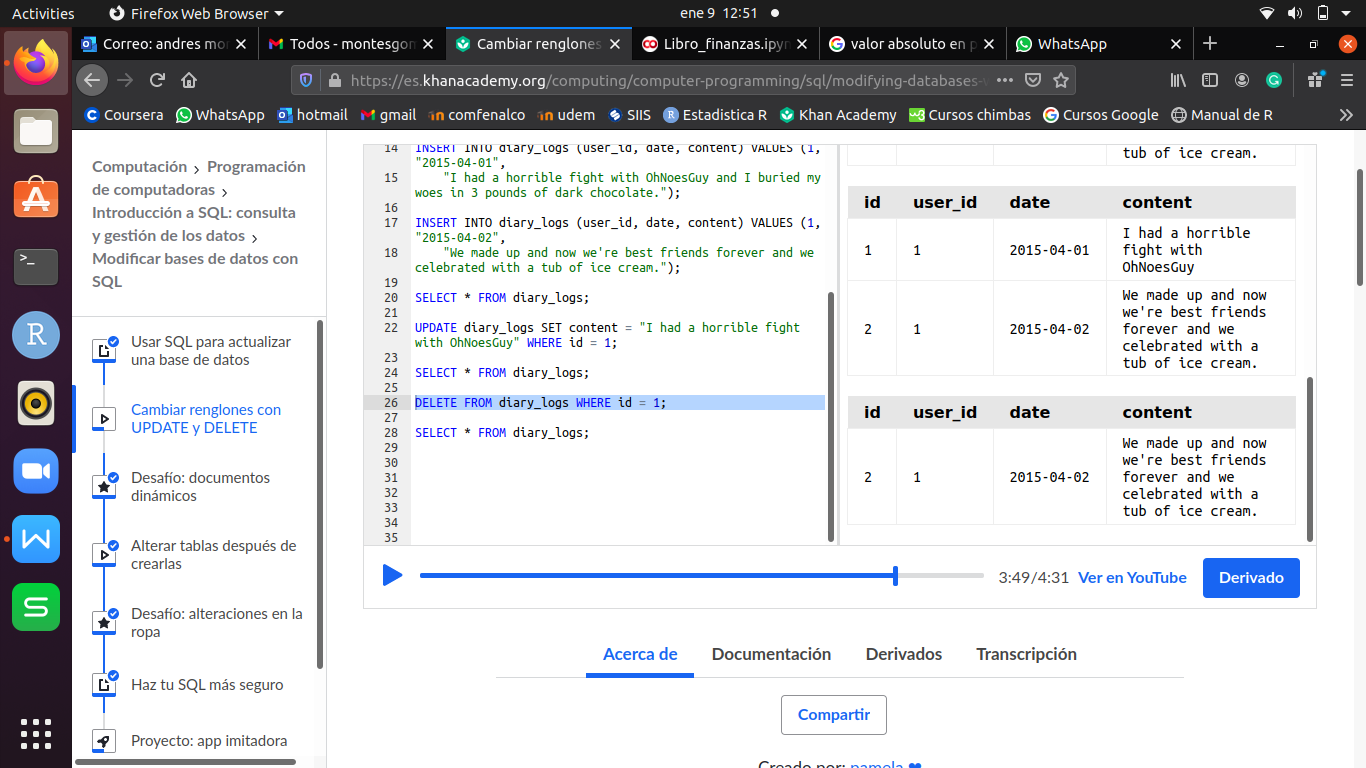


1. **ELIMINAR REGISTROS (DELETE)**

DELETE FROM tabla WHERE id(PK) = # o filtrado;

**EJEMPLO:**

DELETE FROM diary\_logs WHERE id = 1;

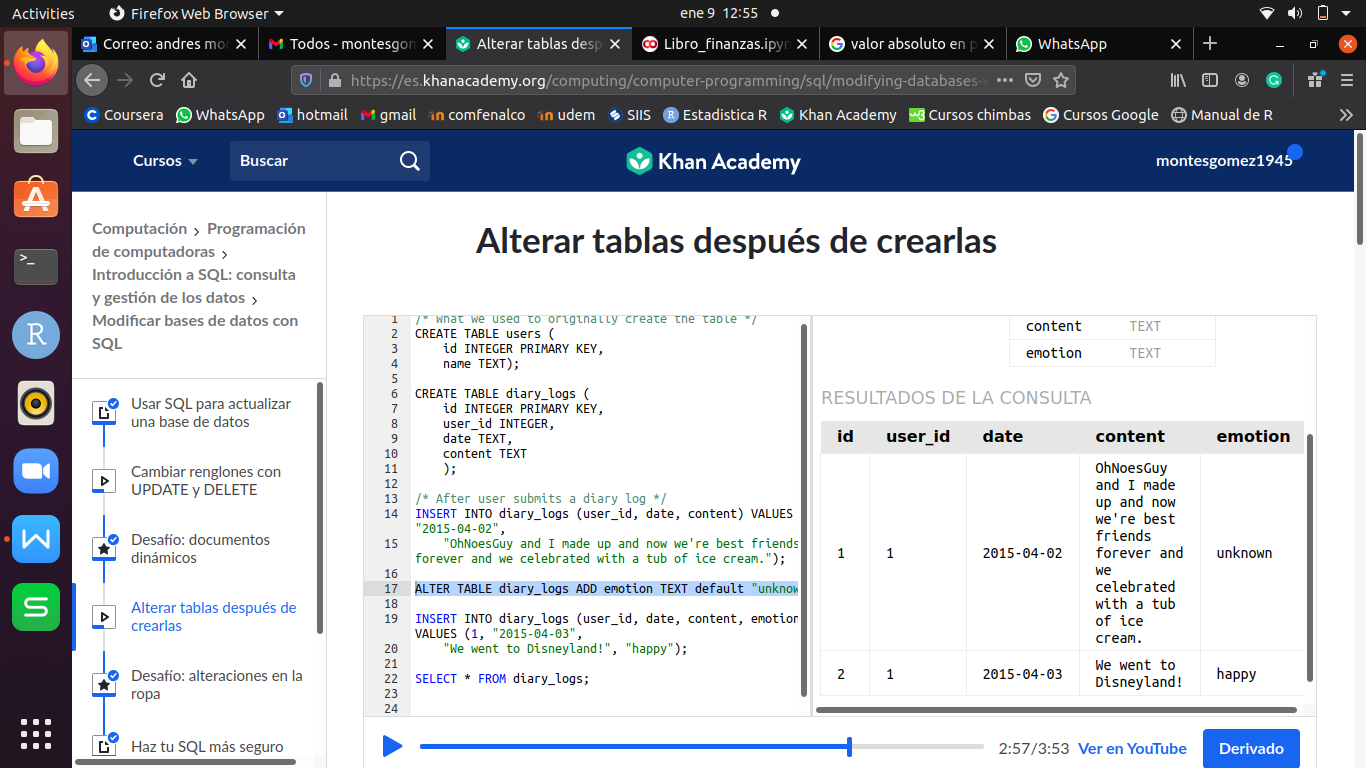


1. **AGREGAR UNA NUEVA COLUMNA (ALTER TABLE)**

ALTER TABLE tabla ADD nombre\_columna TIPO default ‘lo que se desea agregar a la columna nueva de los registros anteriores’;

**EJEMPLO:**

ALTER TABLE diary\_logs ADD emotion TEXT default "unknown";



1. **ELIMINAR UNA TABLA (DROP TABLE)**

DROP TABLE tabla;