/** Grocery list:
Bananas(4)
Peanut Butter (1)
Dark Chocolate Bars (2)
**/

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

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

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

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

SELECT * FROM groceries;

DATABASE SCHEMA

groceries 3 rows

id (PK) INTEGER
name TEXT
quantity INTEGER

QUERY RESULTS

id	name	quantity
1	Bananas	4
2	Peanut Butter	1
3	Dark chocolate bars	2

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

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

SELECT name FROM groceries;
SELECT * FROM groceries;

SELECT * FROM groceries ORDER BY aisle;

SELECT * FROM groceries WHERE aisle > 5 ORDER BY aisle;
```

DATABASE SCHEMA

groceries 6 rows
id (PK) INTEGER
name TEXT
quantity INTEGER
aisle INTEGER

QUERY RESULTS

name
Bananas
Peanut Butter
Dark chocolate bars
Ice cream
Cherries
Chocolate syrup

id	name	quantity	aisle
1	Bananas	4	7
2	Peanut Butter	1	2
3	Dark chocolate bars	2	2

id	name	quantity	aisle
4	Ice cream	1	12
5	Cherries	6	2
6	Chocolate syrup	1	4

id	name	quantity	aisle
2	Peanut Butter	1	2
3	Dark chocolate bars	2	2
5	Cherries	6	2
6	Chocolate syrup	1	4
1	Bananas	4	7
4	Ice cream	1	12

id	name	quantity	aisle
1	Bananas	4	7
4	Ice cream	1	12

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

INSERT INTO groceries VALUES(1, "Bananas", 10, 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);

/*Aggregation function*/

SELECT SUM(quantity) FROM groceries;

SELECT MAX(quantity) FROM groceries;

SELECT * FROM groceries;

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

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

/*
SELECT aisle, SUM(quantity) FROM groceries GROUP BY aisle;
//this will have a query result which is not sensible.
*/
```

DATABASE SCHEMA

groceries 6 rows

id (PK) INTEGER
name TEXT
quantity INTEGER
aisle INTEGER

QUERY RESULTS

SUM(quantity)
21

MAX(quantity)
10

id	name	quantity	aisle
1	Bananas	10	7
2	Peanut Butter	1	2
3	Dark chocolate bars	2	2
4	Ice cream	1	12
5	Cherries	6	2
6	Chocolate syrup	1	4

SUM(quantity)
9
1
10
1

aisle	SUM(quantity)
2	9
4	1
7	10
12	1

Now my database table set up

/*More queries */

```
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, 20, 105);

INSERT INTO exercise_logs(type, minutes, calories, heart_rate) VALUES("dancing", 15, 200, 120);

SELECT * FROM exercise_logs WHERE calories > 50 ORDER BY calories;

/* AND */
SELECT * FROM exercise_logs WHERE calories > 50 AND minutes < 30;
```

```
/* OR */
SELECT * FROM exercise_logs WHERE calories > 50 OR heart_rate > 100;
```

DATABASE SCHEMA

exercise_logs 3 rows
id (PK) INTEGER
type TEXT
minutes INTEGER
calories INTEGER
heart_rate INTEGER

QUERY RESULTS

id	type	minutes	calories	heart_rate
1	biking	30	100	110
3	dancing	15	200	120

id	type	minutes	calories	heart_rate
3	dancing	15	200	120

id	type	minutes	calories	heart_rate
1	biking	30	100	110
2	biking	10	20	105
3	dancing	15	200	120

```
/*More queries */

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,
20, 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);

/*Query data from column "type"*/

SELECT * FROM exercise_logs;

SELECT * FROM exercise_logs WHERE type = "biking";

SELECT * FROM exercise_logs WHERE type = "biking" OR type = "hiking" OR type =
"tree climbing" OR type = "rowing";

/* IN */
SELECT * FROM exercise_logs WHERE type IN("biking", "hiking", "tree climbing",
"rowing");

SELECT * FROM exercise_logs WHERE type NOT IN("biking", "hiking", "tree climbing",
"rowing");

/*More work on IN*/

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 drs_favorites;

 SELECT type FROM drs_favorites;
```

```
SELECT * FROM exercise_logs WHERE type IN("biking", "hiking");

/*select updated data when tables changes
use subquery
*/

SELECT * FROM exercise_logs WHERE type IN(SELECT type FROM drs_favorites);

SELECT * FROM exercise_logs WHERE type IN(SELECT type FROM drs_favorites
WHERE reason = "Increases cardiovascular health");

/*unexact match using LIKE*/

SELECT * FROM exercise_logs WHERE type IN
(SELECT type FROM drs_favorites WHERE reason LIKE "%cardiovascular%");
```

DATABASE SCHEMA

exercise_logs 7 rows

id (PK) INTEGER
type TEXT
minutes INTEGER
calories INTEGER
heart_rate INTEGER

drs_favorites 2 rows

id (PK) INTEGER
type TEXT
reason TEXT

QUERY RESULTS

id	type	minutes	calories	heart_rate
1	biking	30	100	110
2	biking	10	20	105

id	type	minutes	calories	heart_rate
3	dancing	15	200	120
4	tree climbing	30	70	90
5	tree climbing	25	72	80
6	rowing	30	70	90
7	hiking	60	80	85

id	type	minutes	calories	heart_rate
1	biking	30	100	110
2	biking	10	20	105

id	type	minutes	calories	heart_rate
1	biking	30	100	110
2	biking	10	20	105
4	tree climbing	30	70	90
5	tree climbing	25	72	80
6	rowing	30	70	90
7	hiking	60	80	85

id	type	minutes	calories	heart_rate
1	biking	30	100	110
2	biking	10	20	105
4	tree climbing	30	70	90
5	tree climbing	25	72	80
6	rowing	30	70	90
7	hiking	60	80	85

id	type	minutes	calories	heart_rate
3	dancing	15	200	120

id	type	reason
1	biking	Improves endurance and flexibility
2	hiking	Increases cardiovascular health

type
biking
hiking

id	type	minutes	calories	heart_rate
1	biking	30	100	110
2	biking	10	20	105
7	hiking	60	80	85

id	type	minutes	calories	heart_rate
1	biking	30	100	110
2	biking	10	20	105
7	hiking	60	80	85

id	type	minutes	calories	heart_rate
7	hiking	60	80	85

id	type	minutes	calories	heart_rate
7	hiking	60	80	85

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,
20, 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;

SELECT type FROM exercise_logs;

SELECT type FROM exercise_logs GROUP BY type;

SELECT type, SUM(calories) FROM exercise_logs GROUP BY type;

SELECT type, SUM(calories) AS total_calories FROM exercise_logs GROUP BY type;

/*HAVING*/
SELECT type, SUM(calories) AS total_calories FROM exercise_logs
   GROUP BY type
   HAVING total_calories > 150;

SELECT type, AVG(calories)
    AS avg_calories FROM exercise_logs
    GROUP BY type
    HAVING avg_calories > 70;

/*select type with the number of logged exercises greater and equal to 2  */
SELECT type FROM exercise_logs
   GROUP BY type
   HAVING COUNT(*) >= 2;
```

DATABASE SCHEMA

exercise_logs 7 rows
id (PK) INTEGER
type TEXT
minutes INTEGER
calories INTEGER
heart_rate INTEGER

QUERY RESULTS

id	type	minutes	calories	heart_rate
1	biking	30	100	110
2	biking	10	20	105
3	dancing	15	200	120
4	tree climbing	30	70	90
5	tree climbing	25	72	80
6	rowing	30	70	90
7	hiking	60	80	85

type
biking
biking
dancing
tree climbing
tree climbing
rowing
hiking

type
biking
dancing

type
hiking
rowing
tree climbing

type	SUM(calories)
biking	120
dancing	200
hiking	80
rowing	70
tree climbing	142

type	total_calories
biking	120
dancing	200
hiking	80
rowing	70
tree climbing	142

type	total_calories
dancing	200

type	avg_calories
dancing	200
hiking	80
tree climbing	71

type
biking
tree climbing

```
/*A few more advanced SQL features*/

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,
20, 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("dancing", 15,
165, 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;

/*Count rows above max heart_rate*/
SELECT COUNT(*) FROM exercise_logs WHERE heart_rate > 220-30;

/*Count rows above the target 50%-90% of max heart_rate */
SELECT COUNT(*) FROM exercise_logs WHERE
heart_rate >= ROUND(0.50*(220-30))
AND heart_rate <= ROUND(0.90*(220-30));

/* CASE */
SELECT type, heart_rate,
   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;


/*count the number of rows for heart_rate zone*/
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
```

DATABASE SCHEMA

exercise_logs 8 rows

id (PK) INTEGER
type TEXT
minutes INTEGER
calories INTEGER
heart_rate INTEGER

QUERY RESULTS

id	type	minutes	calories	heart_rate
1	biking	30	100	110
2	biking	10	20	105
3	dancing	15	200	120
4	dancing	15	165	120
5	tree climbing	30	70	90
6	tree climbing	25	72	80

id	type	minutes	calories	heart_rate
7	rowing	30	70	90
8	hiking	60	80	85

COUNT(*)
0

COUNT(*)
4

type	heart_rate	hr_zone
biking	110	within target
biking	105	within target
dancing	120	within target
dancing	120	within target
tree climbing	90	below target
tree climbing	80	below target
rowing	90	below target
hiking	85	below target

COUNT(*)	hr_zone
4	below target
4	within target

```
/*one table: students*/
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", "555-666", "2002-07-04");
INSERT INTO students(first_name, last_name, email, phone, birthdate)
VALUES("Alice", "Wonderland", "alice@wonderland.com", "555-4444", "2002-07-04");

/*another table: student_grades*/
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 * FROM students;

SELECT * FROM student_grades;


/*Cross join related tables*/

/*for every row in the first table,
it creates with rows in other table, we ended with rows grouped by the rows in the first
table*/
/*but we only want the rows which match together if student_id matches the id in the student
table*/
SELECT * FROM students, student_grades;



/*Inner join related tables*/
SELECT * FROM students, student_grades
  WHERE students.id = student_grades.student_id;


/*Explicit inner joint --JOIN*/
SELECT * FROM students
  JOIN student_grades
  ON students.id = student_grades.student_id;


SELECT first_name, last_name, test, grade FROM students
  JOIN student_grades
  ON students.id = student_grades.student_id;



SELECT first_name, last_name, test, grade
FROM students
```

```
    JOIN student_grades
    ON students.id = student_grades.student_id
    WHERE grade > 90;
```

```
SELECT students.first_name, students.last_name, student_grades.test, student_grades.grade
FROM students
    JOIN student_grades
    ON students.id =student_grades.student_id
    WHERE grade > 90;
```

DATABASE SCHEMA

students 2 rows

id (PK) INTEGER
first_name TEXT
last_name TEXT
email TEXT
phone TEXT
birthdate TEXT

student_grades 4 rows

id (PK) INTEGER
student_id INTEGER
test TEXT
grade INTEGER

QUERY RESULTS

id	first_name	last_name	email	phone	birthdate
1	Peter	Rabbit	peter@rabbit.com	555-666	2002-07-04
2	Alice	Wonderland	alice@wonderland.com	555-4444	2002-07-04

id	student_id	test	grade
1	1	Nutrition	95
2	2	Nutrition	92
3	1	Chemistry	85
4	2	Chemistry	95

id	first_name	last_name	email	phone	birthdate
1	Peter	Rabbit	peter@rabbit.com	555-666	2002-07-04
1	Peter	Rabbit	peter@rabbit.com	555-666	2002-07-04
1	Peter	Rabbit	peter@rabbit.com	555-666	2002-07-04
1	Peter	Rabbit	peter@rabbit.com	555-666	2002-07-04
2	Alice	Wonderland	alice@wonderland.com	555-4444	2002-07-04
2	Alice	Wonderland	alice@wonderland.com	555-4444	2002-07-04
2	Alice	Wonderland	alice@wonderland.com	555-4444	2002-07-04
2	Alice	Wonderland	alice@wonderland.com	555-4444	2002-07-04

id	first_name	last_name	email	phone	birthdate
1	Peter	Rabbit	peter@rabbit.com	555-666	2002-07-04
2	Alice	Wonderland	alice@wonderland.com	555-4444	2002-07-04
1	Peter	Rabbit	peter@rabbit.com	555-666	2002-07-04
2	Alice	Wonderland	alice@wonderland.com	555-4444	2002-07-04

id	first_name	last_name	email	phone	birthdate
1	Peter	Rabbit	peter@rabbit.com	555-666	2002-07-04
2	Alice	Wonderland	alice@wonderland.com	555-4444	2002-07-04
1	Peter	Rabbit	peter@rabbit.com	555-666	2002-07-04
2	Alice	Wonderland	alice@wonderland.com	555-4444	2002-07-04

first_name	last_name	test	grade
Peter	Rabbit	Nutrition	95
Alice	Wonderland	Nutrition	92

first_name	last_name	test	grade
Peter	Rabbit	Chemistry	85
Alice	Wonderland	Chemistry	95

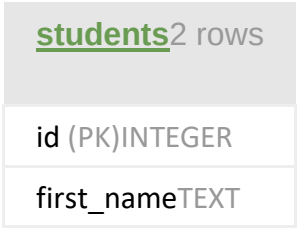
first_name	last_name	test	grade
Peter	Rabbit	Nutrition	95
Alice	Wonderland	Nutrition	92
Alice	Wonderland	Chemistry	95

first_name	last_name	test	grade
Peter	Rabbit	Nutrition	95
Alice	Wonderland	Nutrition	92
Alice	Wonderland	Chemistry	95

```
/*one table: students*/
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", "555-666", "2002-07-04");
INSERT INTO students(first_name, last_name, email, phone, birthdate)
VALUES("Alice", "Wonderland", "alice@wonderland.com", "555-4444", "2002-07-04");

/*another table: student_grades*/
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);
```

```
/*the third table: student_projects*/
CREATE TABLE student_projects(id INTEGER PRIMARY KEY, student_id INTEGER,
        title  TEXT);

INSERT INTO student_projects(student_id, title)   VALUES(1, "Carrotapault");


/*Explicit Inner JOIN*/

SELECT students.first_name, students.last_name, student_projects.title
   FROM students
   JOIN student_projects
   ON students.id = student_projects.student_id;


/*outer join*/
/*LEFT means roatating along rows in the left table*/
/*OUTER means retaining the rows in the left table when there is no matches in the right
table and fills in 'NULL' if it cannot find matches in right table*/

SELECT students.first_name, students.last_name, student_projects.title
   FROM students
   LEFT OUTER JOIN student_projects
   ON students.id = student_projects.student_id;


/*Similarly, RIGHT OUT JOIN; often just exchange the positions between the left table and
the right table*/

/*Similarly, FULL OUTER JOIN; that matches rows, if it can, on both the left and the right
side, and fills in 'NULL' when it cannot on either side*/
```

DATABASE SCHEMA



students 2 rows

last_name TEXT
email TEXT
phone TEXT
birthdate TEXT

student_grades 4 rows

id (PK) INTEGER
student_id INTEGER
test TEXT
grade INTEGER

student_projects 1 row

id (PK) INTEGER
student_id INTEGER
title TEXT

QUERY RESULTS

first_name	last_name	title
Peter	Rabbit	Carrotapault

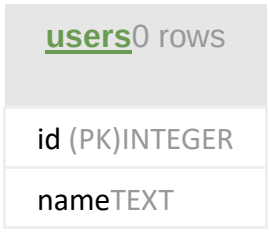
first_name	last_name	title
Peter	Rabbit	Carrotapault
Alice	Wonderland	NULL

/*Changing rows with UPDATE and DELETE*/

/*user table*/
CREATE TABLE users(
  id INTEGER PRIMARY KEY,

```
  name TEXT);

CREATE TABLE diary_logs(
  id INTEGER PRIMARY KEY,
  user_id INTEGER,
  date TEXT,
  content TEXT);

/*After user submitted their new diary log*/
INSERT INTO diary_logs(user_id, date, content) VALUES(1, "2015-04-01", "I had a
horrible fight with OhNoesGuy and 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 are best forever and we celebrated with a tub of ice cream.");

SELECT * FROM diary_logs;

UPDATE diary_logs SET content = "I had a horrible fight with OhNoesGuy"
    WHERE id =1;

UPDATE diary_logs SET content = "I had a horrible fight" WHERE user_id = 1 AND date =
"2015-04-01";

SELECT * FROM diary_logs;

/*delete specified roles*/

DELETE FROM diary_logs WHERE id = 1;

SELECT * FROM diary_logs;


/*or add a 'delete' column in the database, then they will do something like set "deleted" to
"TRUE"*/
/*If the user want to delete, they filter based on 'deleted = FALSE' in the SELECT queries.*/
/*this means you actually not deleting data, which can be a little bit safer*/
```

DATABASE SCHEMA



diary_logs 1 row
id (PK) INTEGER
user_id INTEGER
date TEXT
content TEXT

QUERY RESULTS

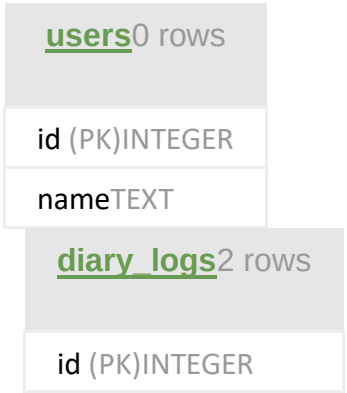
id	user_id	date	content
1	1	2015-04-01	I had a horrible fight with OhNoesGuy and buried my woes in 3 pounds of dark chocolate.
2	1	2015-04-02	We made up and now we are best forever and we celebrated with a tub of ice cream.

id	user_id	date	content
1	1	2015-04-01	I had a horrible fight
2	1	2015-04-02	We made up and now we are best forever and we celebrated with a tub of ice cream.

id	user_id	date	content
2	1	2015-04-02	We made up and now we are best forever and we celebrated with a tub of ice cream.

```
/*What we used to originally create the table*/
CREATE TABLE users(
    id INTEGER PRIMARY KEY,
    name TEXT);

CREATE TABLE diary_logs(
    id INTEGER PRIMARY KEY,
    user_id INTEGER,
    date TEXT,
    content TEXT);

/*After user submits a diary log*/
INSERT INTO diary_logs(user_id, date, content) VALUES(1, "2015-04-02",
        "OhNoseGuy and I made up and now we are best friends forever and we celebrated
with a tub of ice cream");

ALTER TABLE diary_logs ADD emotion TEXT default "unknown";

INSERT INTO diary_logs(user_id, date, content, emotion)
VALUES(1, "2015-04-03", "We went to Disneyland!", "happy");

SELECT * FROM diary_logs;


/*or give a new column unfilled data filled with "unknown"*/

/*
DROP TABLE diary_logs;

SELECT * FROM diary_logs;
*/
```

DATABASE SCHEMA



diary_logs 2 rows
user_id INTEGER
date TEXT
content TEXT
emotion TEXT

QUERY RESULTS

id	user_id	date	content	emotion
1	1	2015-04-02	OhNoseGuy and I made up and now we are best friends forever and we celebrated with a tub of ice cream	unknown
2	1	2015-04-03	We went to Disneyland!	happy

/*Joining tables to themselves with self-joins*/

```
CREATE TABLE students(
    id INTEGER PRIMARY KEY,
    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", "555-666",
"2002-06-04", 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 id, first_name,last_name, buddy_id     FROM students;

/*self join*/
/*give an alias for one table*/
SELECT students.first_name,
  students.last_name, buddies.email
   FROM students
   JOIN students buddies
   ON students.buddy_id = buddies.id;


/*self join*/
/*set an alias for one column in the querying result*/
SELECT students.first_name, students.last_name, buddies.email as buddy_email
   FROM students
   JOIN students buddies
   ON students.buddy_id = buddies.id;
```

DATABASE SCHEMA

students 4 rows

id (PK)INTEGER
first_nameTEXT
last_nameTEXT
emailTEXT
phoneTEXT
birthdateTEXT
buddy_idINTEGER

QUERY RESULTS

id	first_name	last_name	buddy_id
1	Peter	Rabbit	2

id	first_name	last_name	buddy_id
2	Alice	Wonderland	1
3	Aladdin	Lampland	4
4	Simba	Kingston	3

first_name	last_name	email
Peter	Rabbit	alice@wonderland.com
Alice	Wonderland	peter@rabbit.com
Aladdin	Lampland	simba@kingston.com
Simba	Kingston	aladdin@lampland.com

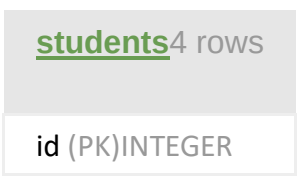
first_name	last_name	buddy_email
Peter	Rabbit	alice@wonderland.com
Alice	Wonderland	peter@rabbit.com
Aladdin	Lampland	simba@kingston.com
Simba	Kingston	aladdin@lampland.com

/*Combining multiple joins*/

/*one table: students*/
```
CREATE TABLE students(
    id INTEGER PRIMARY KEY,
    first_name TEXT,
    last_name TEXT,
    email TEXT,
    phone TEXT,
    birthdate TEXT);

INSERT INTO students VALUES(1, "Peter", "Rabbit", "peter@rabbit.com", "555-666",
"2002-06-24");

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

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

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

/*second table: student_projects*/

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");

/*third table*/
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 * FROM project_pairs;

/*multiple joins, combination of join and self join*/
/*Pair up the project titles*/
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;
```

DATABASE SCHEMA



students 4 rows
first_name TEXT
last_name TEXT
email TEXT
phone TEXT
birthdate TEXT

student_projects 4 rows
id (PK) INTEGER
student_id INTEGER
title TEXT

project_pairs 2 rows
id (PK) INTEGER
project1_id INTEGER
project2_id INTEGER

QUERY RESULTS

id	project1_id	project2_id
1	1	2
2	3	4

title	title
Carrotapault	Mad Hattery
Carpet Physics	Hyena Habitats