Extra - More on JOIN and stuff

A joint venture

Find names that don’t have a unique initial

How do we list all the names of students whose name starts with the same letter as some other student?

```
sqlite> SELECT DISTINCT name FROM students ORDER BY name;
name
Chip
Dale
Dewey
Donald
Goofy
Hewey
Louie
Mickey
Minnie
Pluto
Scrooge
```

Self-join

```
SELECT DISTINCT s1.name
           FROM students s1
           JOIN students s2
             ON SUBSTR(s1.name, 1, 1) = SUBSTR(s2.name, 1, 1)
            AND s1.student_id != s2.student_id -- why do we need this?
       ORDER BY s1.name;
name
Dale
Dewey
Donald
Mickey
Minnie
```

Finding names with unique initial?

```
SELECT name FROM
   ( SELECT name, SUBBSTR(name,1,1) AS "letter", COUNT(*) AS number
       FROM students
   GROUP BY letter
     HAVING number = 1 );
name
Chip
Goofy
Hewey
Louie
Pluto
Scrooge
```

What colors doesn’t any car have?

```
CREATE TABLE color(color_id INTEGER PRIMARY KEY, color TEXT UNIQUE NOT
NULL);
CREATE TABLE car(id INTEGER PRIMARY KEY, license TEXT UNIQUE NOT NULL,
  color_id INTEGER, FOREIGN KEY(color_id) REFERENCES color(color_id));
```

```
sqlite> select * from car;
id          license     color_id
----------  ----------  ----------
1           AAA 001     1
2           AAA 002     2
3           AAA 003     3
```

```
sqlite> select * from color;
color_id    color
----------  ----------
1           Red
2           Blue
3           White
4           Green
5           Silver
6           Gold
7           Pink
8           Grey
9           Black
10          Orange
11          Purple
```

Use an outer join (keep all rows in first table)

```
sqlite>  SELECT color
           FROM color
LEFT OUTER JOIN car
             ON car.color_id = color.color_id
          WHERE license is null;
color
----------
Green
Silver
Gold
Pink
Grey
Black
Orange
Purple
```

Huh?

```
sqlite>  SELECT color, license
           FROM color
LEFT OUTER JOIN car        -- OUTER JOIN: keep  all rows from color!
             ON car.color_id = color.color_id;
color       license
----------  ----------
Red         AAA 001
Blue        AAA 002
White       AAA 003
Green       NULL
Silver      NULL
Gold        NULL
Pink        NULL
Grey        NULL
Black       NULL
Orange      NULL
Purple      NULL
```

So we filter using WHERE license is null

```
sqlite>  SELECT color
           FROM color
LEFT OUTER JOIN car
             ON car.color_id = color.color_id
          WHERE license is null;
color
----------
Green
Silver
Gold
Pink
Grey
Black
Orange
Purple
```

SQLite3 doesn't have RIGHT JOINs

Since SQLite3 doesn't have RIGHT JOINs, we use LEFT OUTER JOIN and put the table whose rows we are interested in to the left of the JOIN

```
sqlite> SELECT color, license
          FROM color LEFT OUTER JOIN car ON car.color_id = color.color_id;
```

left right

All rows in the right table (car) will get NULL values on its columns, since there is no matching car in the result for some color rows!

```
PRAGMA foreign_keys=OFF;
BEGIN TRANSACTION;
CREATE TABLE color(color_id INTEGER PRIMARY KEY, color TEXT UNIQUE NOT NULL);
INSERT INTO "color" VALUES(1,'Red');
INSERT INTO "color" VALUES(2,'Blue');
INSERT INTO "color" VALUES(3,'White');
INSERT INTO "color" VALUES(4,'Green');
INSERT INTO "color" VALUES(5,'Silver');
INSERT INTO "color" VALUES(6,'Gold');
INSERT INTO "color" VALUES(7,'Pink');
INSERT INTO "color" VALUES(8,'Grey');
INSERT INTO "color" VALUES(9,'Black');
INSERT INTO "color" VALUES(10,'Orange');
INSERT INTO "color" VALUES(11,'Purple');

CREATE TABLE car(id INTEGER PRIMARY KEY, license TEXT UNIQUE NOT NULL, color_id INTEGER,
FOREIGN KEY(color_id) REFERENCES color(color_id));
INSERT INTO "car" VALUES(1,'AAA 001',1);
INSERT INTO "car" VALUES(2,'AAA 002',2);
INSERT INTO "car" VALUES(3,'AAA 003',3);
COMMIT;
```