---------- SQLite3 Tutorial 3 ----------

```
-- Show test results for all students for the quiz given on 2018-10-1
-- We need to pull this information from 2 tables this time

SELECT student_id, score, test_type, date
FROM test, test_score
WHERE date = '2018-10-1'
AND test.id = test_score.test_id;

-- Print out the students name with the scores
-- You have to match the student ids for tables test_score and student
-- That way they will only show the test score that corresponds with each
-- individual student

SELECT f_name, l_name, score, test_type, date
FROM test, test_score, student
WHERE date = '2018-10-1'
AND test.id = test_score.test_id
AND test_score.student_id = student.id;

-- List all students along with their number of absences
-- Since we are using an aggregate query here to group data we have to define
-- how we want the information to be grouped when it is displayed on the screen.
-- That is why we define id_number as the way to group information. It is saying
-- that we should calculate the number of absences for each id_number.

SELECT f_name || ' ' || l_name AS NAME,
COUNT(absence.date) AS ABSENCES
FROM student, absence
WHERE absence.student_id = student.id
GROUP BY id;

-- SQLite JOINS

-- Above we defined INNER JOINs by separating tables with a comma. You can also
-- define them with the word INNER JOIN

-- An INNER JOIN is the most common join. An INNER JOIN returns only those
-- records from tables that match. The JOIN CONDITION defines the results.

SELECT f_name || ' ' || l_name AS name, score, test_id
FROM test_score JOIN student
ON student_id = id;

-- To show all students with the number of absences even if they have none we
-- have to use a LEFT JOIN.

-- The LEFT JOIN says that we need a row for each piece of data listed on the
-- left of the join. Don't forget to change WHERE into ON

SELECT f_name || ' ' || l_name AS name,
COUNT(absence.date) AS absences
```

```
FROM student LEFT JOIN absence
ON absence.student_id = student.id
GROUP BY id;

-- A NATURAL INNER JOIN is similar to a LEFT JOIN in that it returns all columns
-- that match in both tables.

SELECT score, test_id
FROM student NATURAL JOIN test_score
WHERE student_id = id;

-- A CROSS INNER JOIN (Cartesian Join) combines all the records from 2 tables.
-- This can sometimes make a mess and should normally be avoided

SELECT score, test_id
FROM student CROSS JOIN test_score;

-- SQLites SELECT can also be used to perform numerous Arithmetic, Boolean,
-- Bitwise, Relational and other Operations

SELECT (1+2) / (6-3) * 10;

SELECT 15 % 10;

-- You can perform boolean operations in which 0 is false and any other number
-- is true

SELECT 1 AND 0, 1 OR 0, NOT 1;

-- Other Operators

SELECT 'Paul' IN ('Mike', 'Phil', 'Paul');

-- BETWEEN can be used to make comparisons as well

SELECT * FROM test_score;

SELECT * FROM test_score
WHERE score
BETWEEN 15 AND 20;

-- Generate minimum and maximum values from a result

SELECT min(id), max(id)
FROM student;

-- Returns the total number of changes made to the
-- database since it was last opened

SELECT total_changes();
```