```
-- Tables with duplicates

-- Important: To set up the context for these examples, we use SQL commands
-- that have not been covered yet in the course, such as CREATE TABLE
-- and INSERT INTO.  They are a preview of material to come in about a week,
-- but you don't need to understand these commands in any detail yet.
-- The point of these examples is something else: how and when duplicates can
-- exist in a table, and can be returned by a query.


csc343h-dianeh=> -- Let's define a table Blah(x, y), and with x as the "primary" key.
csc343h-dianeh=> -- SQL will not allow any duplicate values for x.
csc343h-dianeh=> create table Blah(x int primary key, y int);
NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index "blah_pkey" for table "blah"
CREATE TABLE

csc343h-dianeh=> -- Here we put some harmless values in, and then see what's there.
csc343h-dianeh=> insert into Blah values
csc343h-dianeh-> (1, 3), (2, 4), (7, 9);
INSERT 0 3
csc343h-dianeh=> select * from Blah;
 x | y
---+---
 1 | 3
 2 | 4
 7 | 9
(3 rows)

csc343h-dianeh=> -- Let's confirm that SQL enforces our key constraint.
csc343h-dianeh=> -- (a) We can't repeat an x-value, even if
csc343h-dianeh=> -- the rest of the tuple is different, because x is primary key.
csc343h-dianeh=> insert into Blah values (1, 88);
ERROR:  duplicate key value violates unique constraint "blah_pkey"
csc343h-dianeh=> -- (b) We certainly can't repeat a whole tuple.
csc343h-dianeh=> insert into Blah values (1, 3);
ERROR:  duplicate key value violates unique constraint "blah_pkey"

csc343h-dianeh=> -- What if there is no primary key?  We can define a table with no key.
csc343h-dianeh=> create table Keyless (x int, y int);
CREATE TABLE
csc343h-dianeh=> -- Let's put the same values into it as the first table.
csc343h-dianeh=> insert into Keyless values
csc343h-dianeh-> (1, 3), (2, 4), (7, 9);
INSERT 0 3

-- Now try some duplicates.
csc343h-dianeh=> -- (a) We can repeat an x-value!  No problem.
csc343h-dianeh=> insert into Keyless values (1, 88);
INSERT 0 1
csc343h-dianeh=> select * from keyless;
 x | y
---+----
 1 |  3
 2 |  4
 7 |  9
 1 | 88
(4 rows)
csc343h-dianeh=> -- (b) We can repeat an entire tuple, in fact!
csc343h-dianeh=> insert into Keyless values (1, 3);
INSERT 0 1
csc343h-dianeh=> select * from keyless;
 x | y
---+----
 1 |  3
```

```
 2 |  4
 7 |  9
 1 | 88
 1 |  3
(5 rows)

csc343h-dianeh=> -- Now we have a relation with duplicates!
csc343h-dianeh=> -- This can happen because a relation in SQL is a bag, not a set.

csc343h-dianeh=> -- Even for a relation that has a primary key (and therefore
csc343h-dianeh=> -- can't actually have duplicate tuples), we can use it in queries
csc343h-dianeh=> -- that *create* duplicate tuples.
csc343h-dianeh=>
csc343h-dianeh=> select * from Blah;
 x | y
---+---
 1 | 3
 2 | 4
 7 | 9
(3 rows)

csc343h-dianeh=> insert into Blah values (8, 9);
INSERT 0 1
csc343h-dianeh=> select * from Blah;
 x | y
---+---
 1 | 3
 2 | 4
 7 | 9
 8 | 9
(4 rows)

csc343h-dianeh=> select y from Blah;
 y
---
 3
 4
 9
 9
(4 rows)

csc343h-dianeh=> -- This is just the result of a query, and is not saved
csc343h-dianeh=> -- anywhere.  But we could save it in another relation as long as y is not
csc343h-dianeh=> -- a primary key there.  Remember, relations are bags!

csc343h-dianeh=> create table OneCol (y int);
CREATE TABLE
csc343h-dianeh=> insert into OneCol (select y from Blah);
INSERT 0 4
csc343h-dianeh=> select * from OneCol;
 y
---
 3
 4
 9
 9
(4 rows)
```