Assignment 1
Testing Script

Last updated: **Wednesday 1st March 3:50pm**

Most recent changes are shown in red ... older changes are shown in brown.

In order to test your views and functions, we have written a bunch of SQL and PLpgSQL functions that execute your views and functions and check whether the result the produce matches the expected outputs.

The script is available in the file:

```
/home/cs3311/web/22T3/assignments/ass1/check.sql
```

Loading this file inserts testing functions into *your* database. The testing functions generally have names like `ass1_XXX` or `check_XXX`. If you have any views or functions with names like these, you'll need to change them before loading the tests.

You load the testing functions using the command:

```
$ psql ass1 -f /home/cs3311/web/22T3/assignments/ass1/check.sql
```

This assumes that

- you're on a machine with a running PostgreSQL server
- your database for Assignment 1 is called `ass1`
- you have loaded your `ass1.sql` into the database
- you have access to the `check.sql` file

If you're working on your home machine, you can download the `check.sql` and `check2.sql` files into your Assignment 1 directory and load it into your database from there.

When you run the above commands, you'll see output like

```
SET
CREATE FUNCTION
CREATE FUNCTION
CREATE FUNCTION
CREATE FUNCTION
CREATE FUNCTION
etc. etc. etc.
CREATE TABLE
COPY 1
CREATE FUNCTION
DROP TABLE
CREATE TABLE
COPY 1
CREATE FUNCTION
DROP TABLE
CREATE TABLE
COPY 0
```

There may be `NOTICE`s but these are not errors, no matter how much they look like it. If there are genuine errors, you would see `ERROR` messages. If you see any `ERROR`s, let me (jas@cse) know. The `check.sql` file loads without error into my database on `d2`.

You can load the checking files as many times as you like. They clean up old testing views and functions before re-loading them,

There are individual functions with names like `check_q2()`. You use these as follows:

```
ass1=# select * from check_q2();
 check_q2
----------
 correct
(1 row)
```

If you don't have a view called `q3` currently loaded in the database, you'll see something like:

```
ass1=# select * from check_q3();
             check_q3
--------------------------------------
 No q3 view; did it load correctly?
(1 row)
```

If you view is not working correctly, you'll get a (hopefully) informative message, e.g. if your `q4` view returns the wrong results

```
ass1=# select * from check_q4();
        check_q4
-------------------------
 incorrect result tuples
(1 row)
```

If you want to manually compare the expected output to your output, you can use the following to see what's expected:

```
ass1=# select * from q3_expected;
           brewery            | founded
------------------------------+---------
 Sierra Nevada Brewing Company |    1980
(1 row)
```

If you want more information on why your view might not be correct, you could run a query like:

```
ass1=# (select * from q4) except (select * from q4_expected);
```

or

```
ass1=# (select * from q4_expected) except (select * from q4);
```

which will show you the actual differences.

One difference that might not be obvious is in Q12. If a beer has no ingredients (in the database), the `info` field should have the value `null`. This is different to an empty string (`''`), which looks the same.

The above is useful for testing individual views. If you want to test everything at once, use the following:

```
ass1=# select * from check_all();
 test |               result
------+--------------------------------------
 q1   | correct
 q2   | incorrect result tuples
 q3   | No q3 view; did it load correctly?
 q4   | correct
 q5   | too many result tuples
 q6   | correct
 q7   | missing result tuples
 q8   | No q8 view; did it load correctly?
 q9   | correct
...
(24 rows)
```

The above assumes that your `ass1.sql` is a bit half-baked. Eventually, of course, you'd be hoping to see:

```
ass1=# select * from check_all();
 test |               result
------+--------------------------------------
 q1   | correct
 q2   | correct
 q3   | correct
 q4   | correct
 q5   | correct
 q6   | correct
 q7   | correct
 q8   | correct
 q9   | correct
 q10  | correct
 q11a | correct
 q11b | correct
 q11c | correct
 q11d | correct
 q11e | correct
 q11f | correct
 q12a | correct
 q12b | correct
 q12c | correct
 q12d | correct
 q12e | correct
 q12f | correct
 q12g | correct
(23 rows)
```

Feel free to read `check.sql`. You might learn a bit more PLpgSQL. And give me feedback if you think you can do it better.

Enjoy, *jas*