ENGAGE HEALTH

Utrition Testing Manual

Table of Contents

1. Install Dependencies
2. How to Run the Testing Script
3. Test File Structure

Install Dependencies

The testing on this PostgreSQL database has been wrapped up in a python script. This python script uses a library called “psycopg2” to interact with the PostgreSQL database currently on your computer. If you do not currently have PostgreSQL installed on your machine, please follow the instructions in the file “Utrition Setup Manual” first before returning to this manual.

First, you need to install “pip”, which allows you to install python packages. If you have not installed python, do that first. Use the following command to install pip on your Ubuntu linux system: 'sudo apt-get install python3-pip'

Next, you can install psycopg2 using the “pip” command: 'pip install psycopg2-binary'. Note that this is installing the binary version of this library, which contains some precompiled dependencies. These dependencies for psycopg2 are not installed on your system separately, but rather are contained within the psycopg2 binary. As such, the dependencies can never be updated. If you wish to install the standard psycopg2 package along with installing its dependencies, check the psycopg2 documentation at https://www.psycopg.org/docs/install.html#install-from-source.

How To Run Testing Script

To run the python script from an ubuntu environment, you will use the command 'python3 check_test_cases.py' with a few extra command line arguments. The first argument will be the name of the text file containing all the postgreSQL commands you wish to execute. The next one will be a number specifying how many test cases you expect to pass. The next one will be how many test cases you expect to fail. The final argument is how many prerequisite inserts there are at the beginning of the file. Here is an example of using this script from the command line: 'python3 check_test_cases.py users.txt 18 17 0'

The reason for the final command line argument is because if you are trying to test inserts into a table that has a foreign key, you will get an error if the foreign key is referencing a value that does not currently exist in the database in its respective table. If the program is missing a command line argument, or the arguments are of the wrong type, then the script has undefined behavior.

Test File Structure

The test case files should be structured such that they can be read into psql directly (example on Ubuntu unix: ’psql -f users.txt’), without any python intermediary. As such, the first line of the file should contain the line "\connect <database>", where <database> is the name of the database that you wish to connect to. All test cases should be on a single line. This is done so that the python script can read them more easily.

Here is an example of a test case on one line: ‘INSERT INTO users (id,first_name,last_name,email) VALUES (5,'Jared','Schneider', ‘jared@gmail.com');’

Here is an example of a test case on multiple lines, which will cause the script to have undefined behavior

‘INSERT INTO users (id,first_name,last_name,email)
VALUES (5,'Jared','Schneider','jaredschneider@gmail.com');’

Your file may contain psql comments, which must be on their own line beginning with "--". The python script will ignore these.

The file should be structured such that after the first line containing ‘\connect database’, all of the prerequisites INSERT statements should be listed. Then, if there are 25 test cases that should pass, they should all be written in the file after the prerequisite INSERT statements and before any of the test cases that should fail. Then if there are 30 test cases that should fail, they should all be written in the file after any of the test cases that should pass. This ordering is crucial, as it allows the script to easily determine if there were any test cases that passed when they should have failed, and vice versa. If you had a file named "new_test_cases.txt", organized as described above with 10 prerequisite insert statements, the command to run the program would be: 'python3 check_test_cases.py newtestcases.txt 25 30 10'

You can find the number of test cases that should pass/fail and the number of prerequisites by looking in the file who's test cases you are running. As you add/remove test cases, please update the respective numbers so that it accurately reflects the contents of the file.