The following instructions are the instructions for Assignment 3. Similar scripts will be available for the midterm test.

Testing Your Answers

To help test your answers, download the file **assign3.zip** from LumiNUS Assignment file folder. The unzipped directory contains the following files:

- init.sql - SQL script to initialize the database with tables
- answers.sql - SQL script for drafting your answers to the 10 questions
- status1.sql, . . . , status10.sql - SQL scripts to report a summary of test comparisons for each question
- test1.sql, . . . , test10.sql - SQL scripts to report the details of test comparisons for each question
- set-psql.txt - script to customize psql
- Backup/ - contains a backup copy of the original answers.sql script
- Misc/ - optional SQL scripts for loading data into the database for each test case

answers.sql

The only file that you need to edit for the purpose of testing is the file named **answers.sql**. This file consists of 10 CREATE VIEW statements representing your answers for the 10 questions. For example, the view definition for Question 1 is given as

```
create view v1 (eid)  as
select 1
;
```

You must not modify the schema definition; i.e., the line "create view v1 (eid) as". If your answer for this question is "select eid from Employees", then you should replace the line "select 1" with your answer as follows:

```
create view v1 (eid)  as
select eid
from Employees
;
```

- Your answers in answers.sql are only for the purpose of testing. As this file **will not** be submitted, it is important that you remember to copy each of the tested answers from this file to the appropriate question's answer in Examplify. Furthermore, you must copy the entire view definition as a **valid, single SQL statement**. Taking Question 1 as an example, your answer for Question 1 in Examplify should start with the line "CREATE VIEW v1 (eid) AS" and end with the line containing the semicolon.
- It is important to remember to execute the answers.sql script before testing your answers. Otherwise, if you've modified answers.sql after your last execution of answers.sql, you will be testing your old answers (installed

by the last execution of answers.sql) and not your revised answers in answers.sql.
- As usual, the test cases are meant to help catch certain errors in your answers, and an answer that passes all the provided test cases does not necessarily mean that the answer is indeed correct. The grading of this assignment will be using both the provided test cases as well as additional test cases.

init.sql

For this assignment, instead of using the default database named *postgres*, which might contain other tables previously created that could conflict with the assignment's tables, you will create and connect to a new database named *assign3*. After you've connected to the new database, execute the init.sql script to create the application's tables and additional solution tables that will be used for testing.

If you're currently running psql and connected to the postgres database, execute the following commands to create and connect to the new database named *assign3*, followed by executing the init.sql script.

```
postgres=# create database assign3;
CREATE DATABASE
postgres=# \c assign3
You are now connected to database "assign3" as user "postgres".
assign3=# \i init.sql
```

To execute a shell/commandline command from psql, you can prefix the command with psql's meta-command \!. For example, to show the current working directory, execute the command "\! pwd" (for MacOS) or "\! cd" (for Windows).

answers.sql, statusN.sql & testN.sql

Suppose that you have edited answers.sql with a view definition for v1 (i.e., your answer for Question 1). To test your answer for Question 1, perform the following steps:

- First, execute the script **answers.sql** (using the command "\i answers.sql") to install your answer's view definition for Question 1.
- Next, execute the script **status1.sql** (using the command "\i status1.sql") to obtain a summary of the test outcomes for the provided test cases. The following example output indicates that your answer did not pass the first five test cases. Specifically, for test case 1, your answer's output is different from the correct solution as there are 3 correct tuples that are missing from your output
  and your output contains 2 extra tuples that are absent from the correct output.

```
        Summary of testing query 1
```

```
-----------------------------------------------
 Test case 1 - INCORRECT: MISSING = 3, EXTRA = 2
 Test case 2 - INCORRECT: MISSING = 5, EXTRA = 4
 Test case 3 - INCORRECT: MISSING = 5, EXTRA = 4
 Test case 4 - INCORRECT: MISSING = 5, EXTRA = 4
 Test case 5 - INCORRECT: MISSING = 5, EXTRA = 4
 ....
```

- To see the detailed test comparisons for your answer to Question 1, execute the script **test1.sql** (using the command "\i test1.sql"). The following example output shows the detailed comparison for test case 1 which indicates that there are 4 correct tuples in your output (marked as "OK"), 2 extra tuples in your output (marked as "EXTRA"), and 3 correct tuples that are mising in your output (marked as "MISSING").

eid	Test case 1
108	OK
105	OK
109	OK
115	OK
113	EXTRA
102	EXTRA
101	MISSING
114	MISSING
106	MISSING

As the default mode of psql is quite verbose (it echos each of the executed commands), you can configure psql to be quieter by executing the psql commands in **psql-set.txt**. The execution of set-psql.txt needs only be done once for each psql session.

```
assign3=# \i set-psql.txt
assign3=# \i answers.sql
psql:answers.sql:2: NOTICE:  view "zzanswer" does not exist, skipping
assign3=#
assign3=# \i status1.sql
psql:status1.sql:70: NOTICE:  view "zzanswer" does not exist, skipping
            Summary of testing query 1
......
assign3=#
assign3=# \i test1.sql
......
assign3=#
```

You can ignore psql's warnings about missing views.

Debugging Queries

Suppose that your query for Question 1 is incorrect for test case 5. If you want to issue queries on test case 5's database for debugging, you can load its database by executing the SQL script **Misc/load-db5.sql**.

Re-initializing the Database

Should you need to restore the assign3 database to its initial state, perform the following steps. First, if you are currently connected to assign3 database, you need to disconnect from assign3 by connecting to the default database named postgres. For psql users, execute the following commands:

```
assign3=# \c postgres
postgres=# drop database assign3;
postgres=# create database assign3;
CREATE DATABASE
postgres=# \c assign3
You are now connected to database "assign3" as user "postgres".
assign3=# \i init.sql
```