CS 166: Lab 08 Assignment
Triggers and Stored Procedures

The purpose of the following assignment is to explore how triggers and stored procedures can be used to implement custom domain logic and response to specified changes in the database. You are encouraged to learn it yourself. Here are several helpful links:

1. PostgresSQL docs on triggers
2. Triggers and Stored Procedures
3. PostgresSQL docs on sequences

For this assignment you can reuse the same database from Lab7 or use scripts (*create_tables.sql*) to load your data from scratch. **Remember to use the absolute path of data files to avoid ambiguity.**

Your task is to implement a procedure and trigger to automatically populate part_number with incremented value upon insertion of the new row into part_nyc. After that your INSERT statements should not include value for part_number and will look like this:

```
Insert into part_nyc(supplier,color,on_hand,descr) Values (0,0,20,'Desc');
```

Implement your procedure & trigger in *triggers.sql*. You can test your code using *test.sh*

1. First create a sequence using the following SQL statements:

   ```
   DROP SEQUENCE IF EXISTS part_number_seq;
   CREATE SEQUENCE part_number_seq START WITH 50000;
   ```

2. Create a procedure that will return next value of the aforementioned sequence. Use function nextval('part_number_seq') to get the next value from the sequence.

   Use the following syntax to create your procedure:

   ```
   CREATE OR REPLACE LANGUAGE plpgsql;
   CREATE OR REPLACE FUNCTION func_name
    RETURNS "trigger" AS
    $BODY$
    BEGIN
       ...
    END;
    $BODY$
    LANGUAGE plpgsql VOLATILE;
   ```

3. Use the following syntax to create a trigger calling the procedure upon insertion of the new record:

   ```
   CREATE TRIGGER name { BEFORE | AFTER } { INSERT | UPDATE | DELETE }
   ON table FOR EACH { ROW | STATEMENT }
   EXECUTE PROCEDURE func_name ( arguments );
   ```

   Use the following syntax to drop a trigger before creating a trigger:

   ```
   DROP TRIGGER [IF EXISTS] name ON table;
   ```

Turn-in: Submit the *triggers.sql* file at the end of the lab.

*** DO NOT forget to execute *source ./stopPostgreDB.sh* to stop the server once you are done.