1: Preliminaries

Before starting on this assignment, make sure you can login to the class PostgreSQL server. Instructions are available in the PostgreSQL Slides document on the Resources pages on Piazza.

2: Goal

The second assignment has multiple goals

1: Create Tables on the Postgres Server

2: Load data into the created Tables.

3: Merge data where necessary

4: Write Queries that use the Table data

3: Description

3.1 Creating Table

You will create tables for the source databases (Downtown Videos and City Books)

Important: To create the tables, you must use the following schemas. You must use the attribute names as given, and the attributes must be in the order given.

```
dv_customer (customer_id, first_name, last_name, email, address_id,
    active)
dv_address (address_id, address, address2, district, city_id, postal_code
    , phone)
dv_film (film_id, title, description, length, rating)
```
Listing 1: DowntownVideoDB

```
cb_customers (last_name, first_name)
cb_books (title, author_id)
cb_authors (author_id, first_name, last_name)
```
Listing 2: CityBooksDB

```
mg_customers (customer_id, first_name, last_name, email, address_id,
    active)
```
Listing 3: Merged Tables

3.1.1 Datatypes

For 'id' attributes use integer

For first names, last names, emails, film titles and all address related fields use varying character [50]

For the active attribute which indicates whether the customer has an active account or not use Boolean.

For film description use text.

For film length use small int.

For film rating create the following enumerated type. Make sure you put this command *immediately before* the command to create dv_film table.

```
CREATE TYPE mpaa_rating AS ENUM (
    'G',
    'PG',
    'PG-13',
    'R',
    'NC-17'
);
```

You will write a CREATE TABLE command for each of the tables in the above listings. Write the commands in the order the tables are listed above. Save the commands in the file script1.sql.

3.2 Load Data

Download the data from the Resources page on Piazza. There are six files, one for each table. You can copy data from each file into its corresponding table using this command:

\copy <table> FROM '<filename>'.

Save the commands in a logical order in the file script2.sql.

17th July 2017

3.3 Merge Data

3.3.1 Copy Data

 You need to populate the merged tables with data from both the *City Books* and *Downtown Video* databases.

However, you need to preserve the customer_id of the dv_customerstable. For this reason, you'll first copy data from this table into the merged table.

To copy, use the below syntax. Specify the destination table and attributes, and the source table and attributes.

```
INSERT INTO <destination table> (attribute 1, attribute 2, ...)
SELECT <source table>.<attribute a>, <source table>.<attribute b>, ... FROM
<source table>
```

Ensure that there is the same number of source and destination attributes, and that they are in the correct order. Here's an example: Suppose table `dst` has attributes `X, Y, Z`, and table `src` has attributes `P, Q, R, S`. The following will copy values of `P` and `S` from `src` into `Y` and `Z` in new rows of `dst`:

```
INSERT INTO dst (Y, Z)
SELECT src.P, src.S FROM
src;
```

3.3.2 Create Sequences

Create a [sequence](#) for the merged table. For simplicity, we hard-code the start of the sequence to start just after the highest respective values already in the merged table

```
CREATE SEQUENCE
mg_customers_seq START
600;
```

3.3.3. Associate Sequence with attributes

Then ensure that the id attribute in the merged table uses the sequence to generate values:

```
ALTER TABLE mg_customers ALTER
COLUMN customer_id
SET DEFAULT NEXTVAL('mg_customers_seq');
```

Now when you insert a new record into the merged table, you should omit the id attribute (and corresponding value). The id will be automatically generated by the sequence.

3.3.4. Copy Additional Data

You've copied the Downtown Video data into the merged tables. Now copy the City Books data. There may be overlaps, so ensure that you don't copy the same data twice.

(Assumption: No two customers will have same first name and last name)

That is, select only those City Books customers whose first and last names together do not match any Downtown Video customer's first and last names. (Hint: Select the first and last names of all City Books customers. Separately, select all City Books and Downtown Video customers whose first and last names together do match. Then use the EXCEPT keyword to find the difference of the two sets.)

Insert these first and last names into the merged table. The sequence you created previously will automatically generate new id values.

Save the commands in a logical order in the file
script3.sql

3.4. Write Queries

1. What are the first and last names of all people who are customers of both Downtown Video and City Books ?

2. What are phone numbers of all people who are customers of both Downtown Video and City Books?

3. What are the first and last names of all customers who live in the district having the most customers?

4. What rating is the least common among films in the Downtown Video database, and how many films have that rating? (Return both the rating and the number of films in one result.)

5. What are the first and last names of the top 10 authors when ranked by the number of books each has written? (Return both author name and the number of books of top 10 authors, sorted in descending order)

Save the queries in the above order in the file script4.sql.

4. Testing

Before you submit, login to your database via `psql` and execute all four scripts in order.The command to execute a script is: `\i <filename>`. Verify that every table has been created and that the attributes are in the correct order. Verify that each attribute is assigned its correct data type using the following command: `\d <table>`. This command will also show you the sequence associated with a particular attribute. Verify that all data is loaded into your tables, and that the correct data is copied into the merged tables. Finally, verify the results of the queries.

5. Submission

1. Save your scripts for parts 1, 2, 3, 4 separately as `script1.sql`, `script2.sql`, `script3.sql`, and `script4.sql`. Remember to add comments to the scripts so that the intent of your commands is clear. Put any other information for the grader in a separate README file.

2. Copy the scripts to your home directory on `unix.lt.ucsc.edu` and place them in the cmps182 directory.

3. Login to `unix.lt.ucsc.edu`. At the shell prompt, submit your work:

```
> submit cmps182-dm.u16 lab2 script1.sql script2.sql script3.sql
script4.sql
```

You can submit more than once. Only your latest submission will be graded.