## CS 327E Lab 2: Data Load and Reporting Tables

**Prerequisites:**

1. Completed Lab 1.
2. Continuing to work with same partner or found new partner for Lab 2.
3. Performed updates to Lab 1 based on your graded rubric.
4. Created Postgres RDS instance.
5. Installed psql client locally [1].
6. Connect to Postgres RDS instance from psql.

**Step 1.** Create a new folder in your local git repository called **lab2**. All the work you will do for this lab will go into this folder.

**Step 2.** Create a new **database** named `imdb` on your Postgres RDS instance. To create the database, run the command from your psql client:

```
create database imdb;
```

**Step 3.** Write the **DDL** for the IMDB tables based on the ERD from Lab 1. If you have doubts about the correctness of the ERD you submitted, please ask one of us to review it before proceeding with this step. If you find any mistakes, please update the ERD with any corrections and commit the changes to your **lab1** folder.

Place all the create table statements in a file called `create_base_tables.sql`. Run and verify the script and fix any issues as required.

Here are a few useful psql commands:

- `\c imdb` (to connect to the imdb database)
- `\i create_base_tables.sql` (to run the create table script)
- `\dt` (to see a list of tables in the database)
- `drop database if exists imdb;` (to drop the imdb database)
  `create database imdb;` (re-create the imdb database)

Add the `create_base_tables.sql` script to your git repo. Remember that you can reference the Instacart create table statements [3] which include various data types, primary keys, and foreign keys.

**Step 4.** Download the full IMDB dataset from this link: http://cs327e-fall2017-imdb.s3.amazonaws.com/pg.zip and unzip the file. There should be 10 csv files in the dataset. You do **not** need to add these files to your git repo.

**Step 5.** Write the required `\copy` commands to import the csv files into the appropriate database tables. Each copy command imports the data from exactly one csv file. The copy command is documented in the Postgres manual [4] and examples from the Instacart class demo are also available for reference [5].

Place all the copy commands in a file called `load_data.sql`. Connect to the IMDB database and run the script using the command: `\i load_data.sql;` Fix any syntax errors that occur and re-run the script until it is error-free. Add the file to your git repo.

**Step 6.** Perform some simple verification checks of the imported data. Check the record count of each table to ensure that it matches the output from the copy command. Check some sample values to ensure that all the columns from the csv files were imported into their designated columns in the table. If you notice an issue with the imported data, drop the table, re-create the table, and then re-load the data into the new table.

Place the select statements used for these checks into a file named `check_load.sql`. Do not include the output from each select statement, just the select statement that was run. Add the file to your git repo.

**Step 7.** The IMDB leadership team would like to perform some analytics with your database. They want to analyze how movie ratings have changed over time. They have settled on a simple rating system for this analysis: **appalling** titles, **average** titles, and **outstanding** titles. Titles are considered **appalling** if they have an average rating score of 2.0 or below. Titles are considered **average** if they have an average rating score of 2.1 to 7.9. Titles are considered **outstanding** if they have an average rating score of 8.0 or above.

The leadership team would like to know how many titles fall into the appalling, average, and outstanding categories for each year. But they also want the option to examine the numbers by **type of title** (movie, short, tvseries, etc.) and/or by **genre**(s) associated with the title. For example, how many outstanding TV series were made in 2016 or how many appalling movie dramas are there in the whole database?

Design a dimensional schema that meets these criteria. You should create a new self-contained diagram for this schema that includes the appropriate fact and dimension tables. Create the diagram in LucidChart and download it in png format, choosing the download option "Crop to Content" and naming the file imdb_star.png. Create a data dictionary that explains the entities and important attributes that are in diagram. The data dictionary should be in a text file named data_dictionary.txt. Add both the diagram and data dictionary to your git repo.

**Step 8**. Find the **Stache** entry that you created for Lab 1. Update the entry to include the additional RDS details:

```
{
  "aws-username": "shouldbeAdmin",
  "aws-password": "mypassword",
  "aws-console-link": "myconsolelink",
  "rds-endpoint-link": "myrdsendpoint_without_port_number",
  "rds-username": "shouldbeMaster",
  "rds-password": "myrdspassword"
}
```

Make sure to **not** include a port number with the RDS endpoint. We will assume the Postgres RDS instance is running on the default port of 5432.

**Step 9**. Locate the **commit id** that you will be using for your team's submission. This is a long 40-character that shows up on your main Github repo page next to the heading "Latest commit" (e.g. commit 6ca6f695bca36f7fc2c33485d1080ae30f8b9928). Locate the link to your team's repo. This is the URL to your private repo on Github (e.g. https://github.com/cs327e-fall2017/xyz.git where xyz is your repo name). Go back to the Stache entry and locate the read-only API endpoint and read key. Replace the commit id, repo link, API endpoint, and read key in the json string below with your own:

```
{
  "repository-link": "https://github.com/cs327e-fall2017/xyz.git",
  "commit-id": "6ca6f695bca36f7fc2c33485d1080ae30f8b9928",
  "stache-endpoint": "/api/v1/item/read/62021",
  "stache-read-key": "ec1f815a603234eb8c5e2c02b474839f0b6d3b9e76b103f1ab0463b655e6661b"
}
```

Create a file called **submission.json** that contains your modified json string.

Click on the Lab 2 Assignment in Canvas and upload submission.json. This submission is due by **Friday, 10/06 at 11:59pm**. If it's late, there will be a **10% grade reduction per late day**. This late policy is also documented in the syllabus. **Note: there should be one submission per team.**

**References:**


[1] Snippets wiki: https://github.com/cs327e-fall2017/snippets/wiki
[2] Grading Rubric: http://www.cs.utexas.edu/~scohen/projects/lab2-rubric.pdf
[3] Instacart create table statements: https://tinyurl.com/yd6ztaw3
[4] Copy command documentation: https://www.postgresql.org/docs/9.6/static/sql-copy.html
[5] Instacart copy commands: https://tinyurl.com/ycb7dr8y