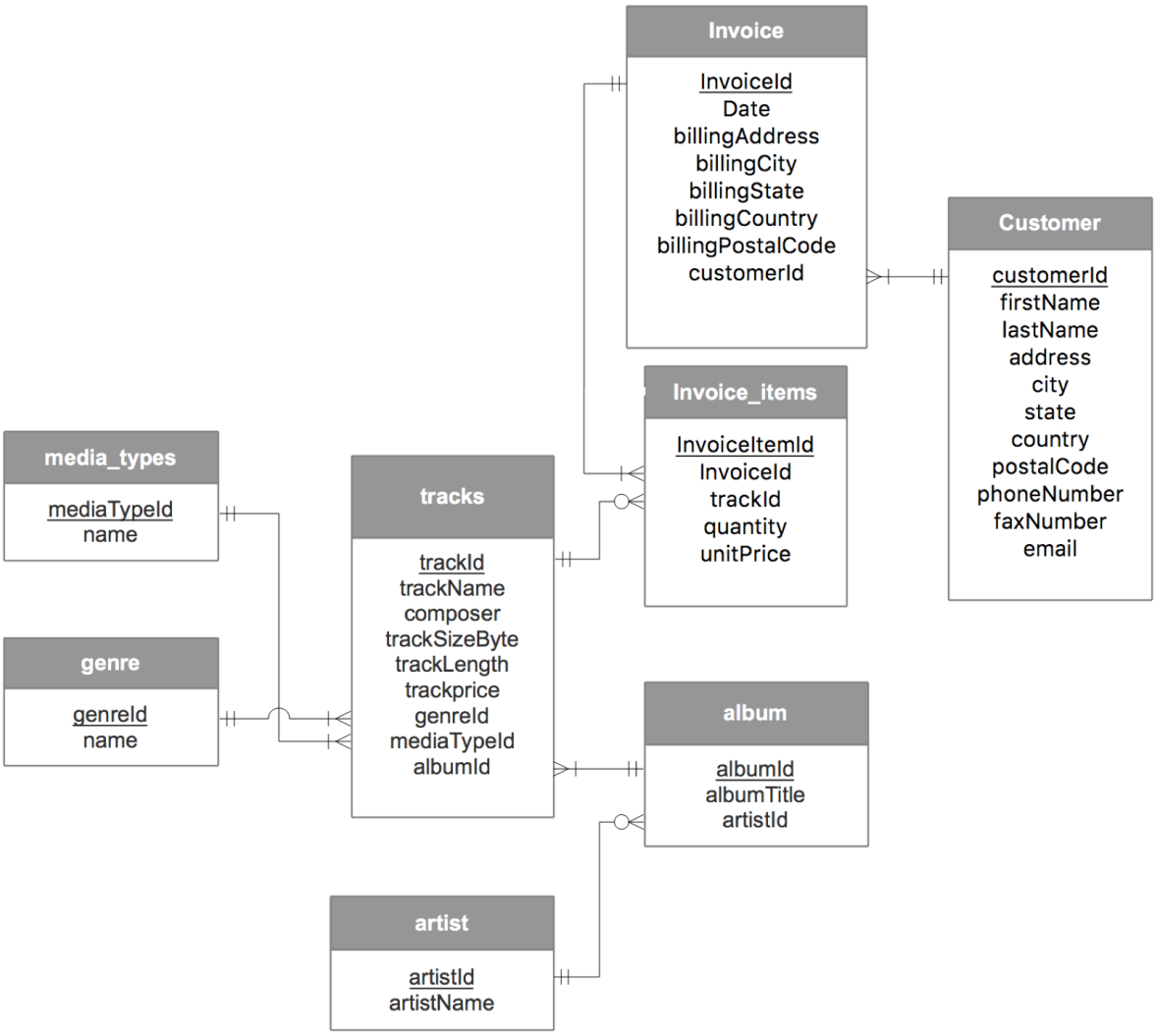
CS217 HW 5 – Database Design and Implementation

The `music.csv` file is an original dataset that we want to transform into an online music database in sqlite. Given the data in this csv file, it is normalized into the following schema.



There are several other rules defined for the database:

1) Artist names and customer last names are unique and not null.

2) The following columns always have a value: album title, customer first and last name, customer email, track name, track price, track length, invoice date, invoice item unit price, and invoice item quantity.

Instructions

Step 1: Create a folder and name it as hw5_yourNetid. For example, if your netid is abc1234, your folder name should be `hw5_abc1234`. Put the attached py file `hw5.py` and the data file `music.csv` in the folder.

Step 2: Execute the python file, it should generate db file `music.db`. Inside this db file, it includes only one table named `source_dataset`. This step is to create an empty database and import all data from the csv file.

Step 3: You can use Sqlite Studio to continue working on this database. Use the `CREATE` and `INSERT` statements to create and populate all the tables.
Step 4: When you are done with implementing the database, collect all your code and put them at the specified location in the python file. Make sure your code is well organized and formatted.
Step 5: Make sure your code is runnable and will generate the music database with all tables. The complete database should include 9 tables: eight tables in the schema diagram and one `source_dataset` table. (Note: It's recommended to delete the current `music.db` file in the directory and then run the python code. It will ensure everything works.)

Helpful tips:

1. If you got decoding errors with the py file, please replace
```
with open('music.csv','r') as f:
```
with
```
with open('music.csv','r',encoding='utf-8') as f:
```

2. When creating and populating the tables, make sure to generate parent tables before generate child table.
3. Make sure the table names and column names are the same as in the schema diagram. If there are typos, it will not pass our testing code.
4. Make sure to use semicolon after each `CREATE` or `INSERT` statements. It's also recommended to use `DROP TABLE IF EXIST` to make the code rerunnable.
5. Make sure your code is well organized and formatted. E.g., you can use double dash (`--`) to make comments and provide explanations to your code.
6. We recommend use IDLE (https://www.python.org/downloads/) for Python since it's the standard Python development environment.
7. In some tables, you might need to create primary key columns. You can use the keyword `AUTOINCREMENT`. Here is a tutorial
https://www.tutorialspoint.com/sqlite/sqlite_using_autoincrement.htm
8. For the `INSERT INTO` statements, here is a detailed tutorial:
https://www.sqlitetutorial.net/sqlite-insert/
9. In the `INSERT INTO...SELECT...` statement, you can use the `JOIN` clause to obtain data from multiple tables. Here is a reference example:
https://stackoverflow.com/questions/44469503/sql-insert-into-with-inner-join

Submission

You should submit a zip file that includes the python file and the csv data file. The python file should be runnable and generate a db file. (Note: do not submit your db file. We will execute your py file on our end and run some tests on the generated db file.) After unzipping your submission, the directory structure should look like the following.
```
hw5_netid/
hw5_netid/hw5.py
hw5_netid/music.csv
```