

# MSiA-413 Data Management and Information Processing

## Homework 5: Advanced SQL Queries and Data Creation

Name 1: \_\_\_\_\_

NetID 1: \_\_\_\_\_

Name 2: \_\_\_\_\_

NetID 2: \_\_\_\_\_

Name 3: \_\_\_\_\_

NetID 3: \_\_\_\_\_

### Instructions

You should submit this homework assignment via Canvas. Acceptable formats are word files, text files, and pdf files. Paper submissions are not allowed and they will receive an automatic zero.

As explained during lecture and in the syllabus, assignments are done in groups. The groups have been created and assigned (randomly) by the instructors. Each group needs to submit only one assignment (i.e., there is no need for both partners to submit individually the same homework assignment).

Each group can submit solutions multiple times (for example, you may discover an error in your earlier submission and choose to submit a new solution set). We will grade only the last submission and ignore earlier ones.

Make sure you submit your solutions before the deadline. The policies governing academic integrity, tardiness and penalties are detailed in the syllabus.

**Due Date: Tuesday November 27, 11:59 pm**

## EntertainmentAgency.sqlite Database (60 points)

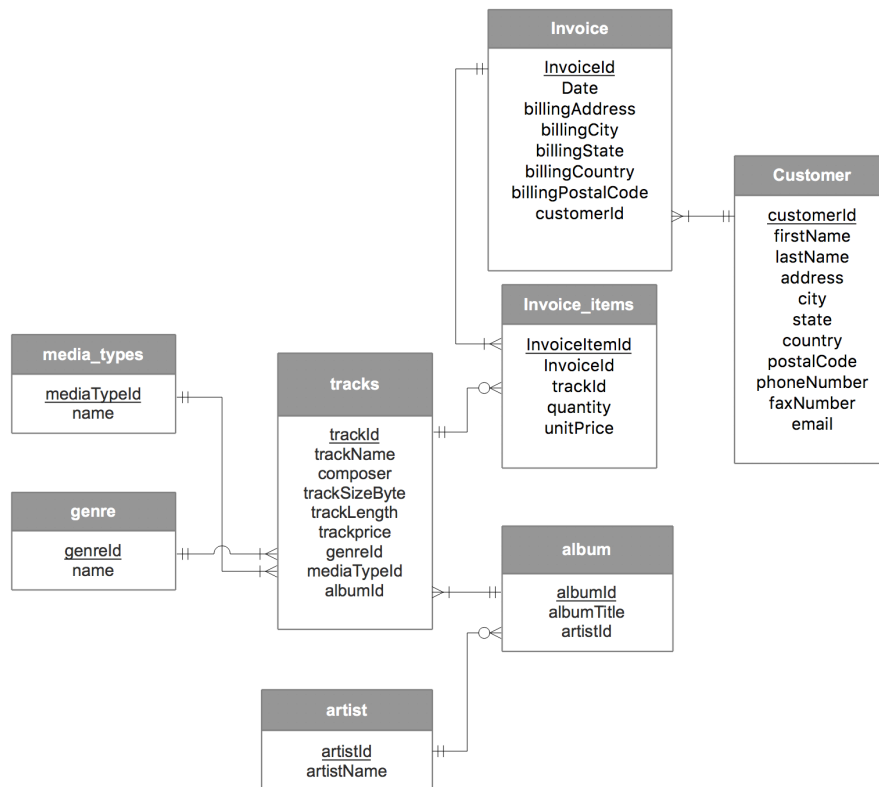
- 1) **(5 points)** Find the EntertainerID of the entertainers that have no engagements. You **must** use EXCEPT for full credit.
- 2) **(10 points)** Find the EntertainerID **and** stage name of the entertainers that have no engagements. Your answer must be a single query with no subqueries. You **must not** directly use the result of question (1) above.
- 3) **(10 points)** For all customers that have less than 10 engagements, list the customer ID, full name (i.e., a string containing the customer's first name followed by the last name with a space in between), and number of engagements, in ascending order of number of engagements. Your answer must be a single query with no subqueries.
- 4) **(5 points)** Using a single query, identify the members that have not been assigned a gender and update their gender to male. The updated table will be used to answer later questions in this homework.
- 5) **(10 points)** Using the updated database/table from question (4), find the number of male and female members (separate counts for each gender) for each entertainer. The output table should have the columns EntertainerID, Gender, and GenderCount. The query **must** use the UNION operator.
- 6) **(10 points)** Write a query to answer question (5) above, but this time the query **must not** use a set operation and it **must** use a natural join. *Hint:* GROUP BY can take multiple columns as arguments.
- 7) **(10 points)** You want to classify each entertainer as follows:
  - Super Band (if it has more than 10 engagements)
  - Regular Band (if it has more than 7 but no more than 10 engagements)
  - Support Band (if it has at least one engagement, but no more than 7), and
  - Amateur Band (if it has no engagements)

Write the query that makes this classification and returns the class of the entertainer, the entertainer's stage name, and the number of engagements, with the entertainers appearing in descending rank (i.e., super bands first, followed by regular bands, then support bands, and amateurs at the bottom). Your answer must be a single query with no subqueries.

## Online Music Store Database (40 points + 50 bonus points)

- 8) **(30 points)**. The hw5\_original.csv file is a database of an on-line music store in a comma-separated file format. In order to remove redundancy and inefficiencies, we normalized it according to the following rules:
- Artist names, customer last names and invoice IDs are unique
  - Each artist can have zero or more albums
  - Each album is made by exactly one artist
  - Each track appears in exactly one album. Note that some tracks with the same name from the same composer may appear in different albums; in that case, however, they have different lengths, so they are considered different tracks (i.e., the tuple <trackName, trackLength> is unique for each track)
  - Each album has at least one track
  - Each track belongs to exactly one genre
  - Each genre is represented by at least one track
  - Each track is stored in exactly one media type
  - Each media type is used by at least one track
  - Each invoice item is for exactly one track and part of exactly one invoice
  - Each invoice has at least one invoice item
  - Some tracks may have never been sold (so there are no invoices for them)
  - Each invoice is issued to exactly one customer
  - 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.

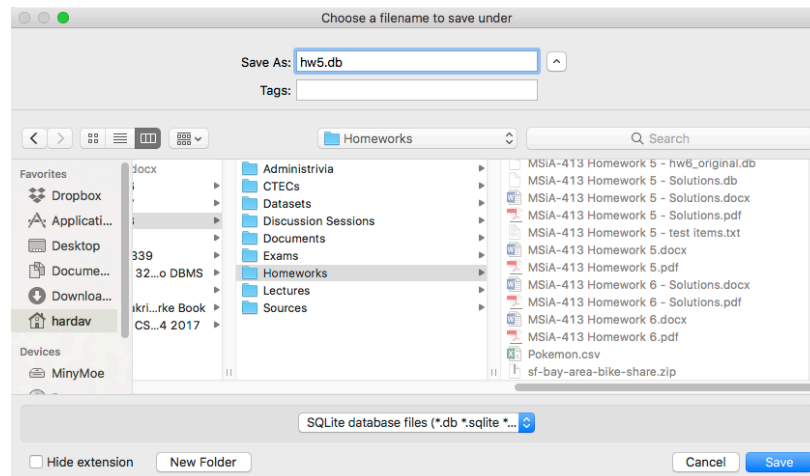
The corresponding normalized database diagram is given below:



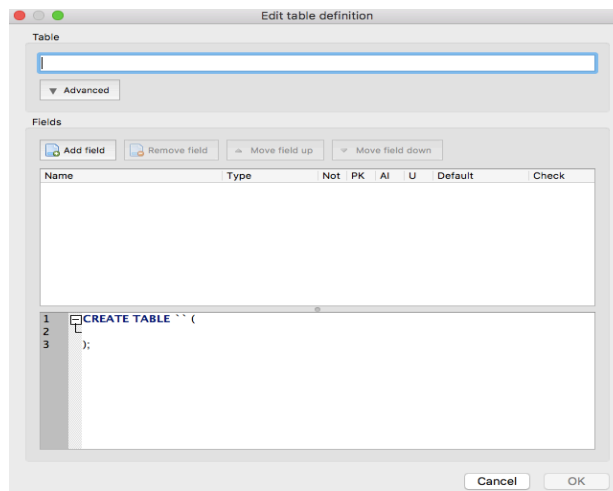
We now want to create a SQLite database that implements the normalized database. The first step is to create a new database with just one table that has the same data as the CSV file. Here is how to do that:

### **(0 points) Part A: Creating a new database and importing a CSV file**

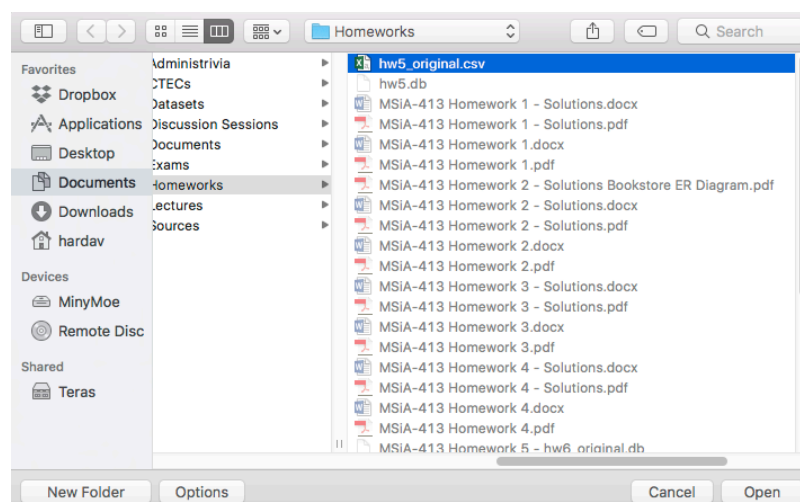
Start DB Browser for SQLite. Create a new database (File → New Database). A window will pop up like the one below. Specify a name and path to your new database file and click “save” to create the file.



Then, another window will pop up like the one below that asks for a table definition. Click cancel.



Then, create and populate one table with the data from the hw5\_original.csv file. The easiest way to do this is to use File → Import → Table from CSV file. Select the hw5\_original.csv file and click open.



Then, a window will pop up in which you specify how the CSV data will be imported. Make sure “Column names in first line” is checked and the UTF-8 encoding is selected, as in the snapshot below.

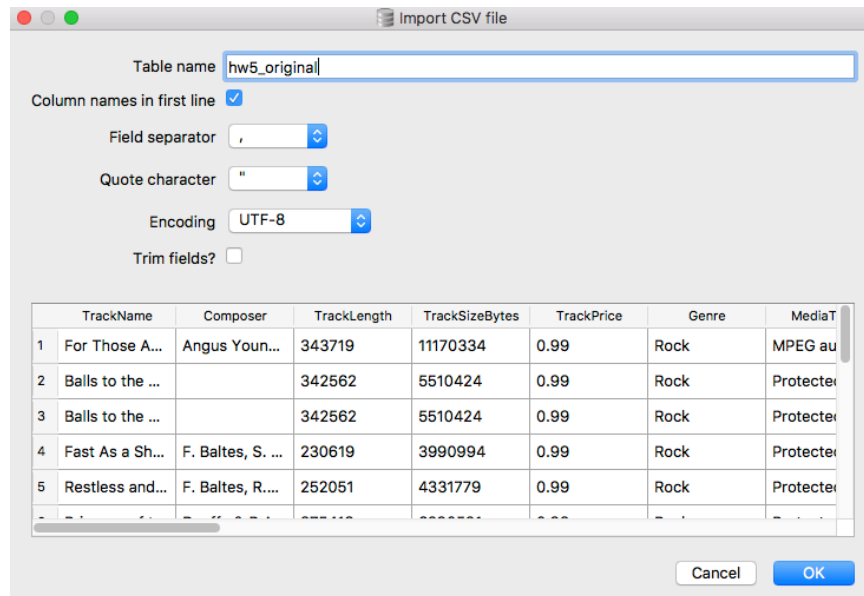


Table name: hw5\_original

Column names in first line: ☒

Field separator: ,

Quote character: "

Encoding: UTF-8

Trim fields?: ☐

	TrackName	Composer	TrackLength	TrackSizeBytes	TrackPrice	Genre	MediaT
1	For Those A...	Angus Youn...	343719	11170334	0.99	Rock	MPEG au
2	Balls to the ...		342562	5510424	0.99	Rock	Protecter
3	Balls to the ...		342562	5510424	0.99	Rock	Protecter
4	Fast As a Sh...	F. Baltes, S. ...	230619	3990994	0.99	Rock	Protecter
5	Restless and...	F. Baltes, R....	252051	4331779	0.99	Rock	Protecter

Cancel OK

This process automatically creates a single table (with the default name `hw5_or iginal`) with as many columns as the CSV file, and with the column names extracted from the first line of the CSV file. Now, it is time to create the tables of the normalized database and populate them with data.

### **(30 points) Part B: Creating the normalized database**

With the single-table database above as the starting point, create the database tables that follow the normalized database diagram shown earlier. Then, issue SQL queries against the original table you created (`hw5_or iginal`) to populate these tables with data. Include in your answer:

- All the SQL commands that you used to create the tables and populate the data.
- The resulting SQLite database (submit it as a separate file through canvas). Remember to click “Write Changes” to save your data and tables. Please do not drop the `hw5_or iginal` table (the initial table you created when uploading the data from the CSV file). Please leave it in your database for debugging and grading purposes.

Please note that:

- In the SQL command that creates a table you must explicitly specify the primary and foreign keys and any `UNIQUE` and `NOT NULL` constraints.
- When you import a row from a CSV file into a table, an empty column may have as a value the empty string. An empty string is different than a `NULL` string. This means that if you want to check whether a field has a value, you may need to compare against the empty string and not against a `NULL` value (i.e., instead of writing `WHERE customerLastName IS NOT NULL` you may need to write `WHERE customerLastName != ""`).

As a starting point, below are the SQL commands to create the table for media types. Use these SQL statements as the first statements in your answer, and proceed with creating and populating the remaining tables. We assume that the imported data from the CSV file were stored in a table with name `hw5_or iginal`.

```
drop table if exists media_types;
create table media_types (
    mediaTypeId integer not null primary key autoincrement,
    mediaName nvarchar(20)
);
insert into media_types (mediaName)
select distinct mediaType from hw5_or iginal;
```

- 9) **(10 points)** Find the best-selling artist and how much customers spent in buying this artist's songs, based on the normalized database that you created and populated in the previous question (8).
- 10) **(50 bonus points)** Instead of creating and loading the database through the GUI as explained in Problem (8) Part A, write your own program to do it. Your program can be written in any language (e.g., C, C++, Java, Python, Pearl). Your program should open the CSV file, interface with the SQLite database, and issue SQL commands to the database to create and populate the `hw5_original` table with data from the CSV file.

There are several tutorials online on the programming interfaces supported by SQLite. For example, at the bottom of <http://www.sqlitetutorial.net/> you can find tutorials for SQLite interfaces with Python, Java, PHP and node.js. Similarly, at the bottom of the left pane at <https://www.tutorialspoint.com/sqlite/> you can find tutorials of SQLite interfacing with C, C++, Java, Python, Pearl, and PHP. There are many other tutorials online that expand the choices to other programming languages (e.g., R, Matlab, ML, etc). Feel free to find a tutorial that best fits your needs and utilize it.

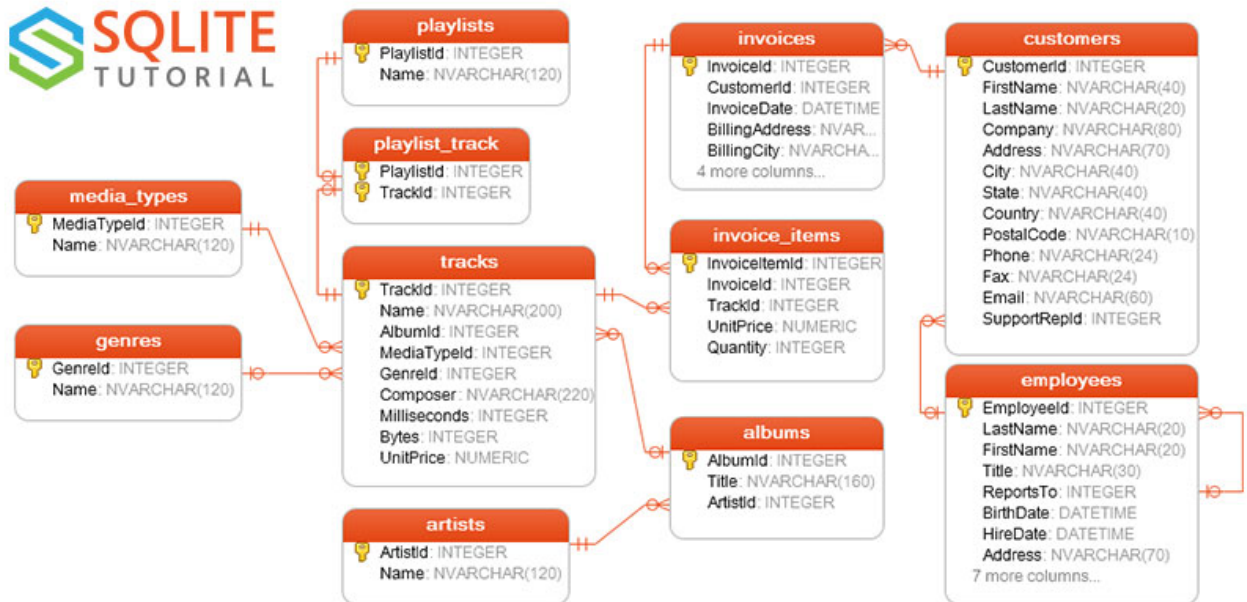
To receive full credit in this bonus problem you must submit with your answer a tarball with your program's source code modules, and instructions on how to install the necessary libraries, how to compile it and how to run it. You must also provide pointers to the tutorial(s) or any other online sources you used.

Because of the complexity of the bonus problem and the free choice of programming languages we give you, **we will not be able to provide TA/instructor support for the bonus problem.** Remember this is a bonus problem, not a required one. You can get full credit without answering Problem (10).

### Reference: creating a CSV dataset through sqlite3

This part is written as a reference of how I created the dataset and stored it in a CSV file for problems 8-10. It is not needed to solve this homework, but it is given here for your perusal and future reference.

- I took the database from <http://www.sqlitetutorial.net/sqlite-sample-database/>



- Then, I joined these tables together into a single result table using the following commands on SQLite's command line interface (sqlite3):

```
sqlite> .open hw5_original.db
sqlite> .headers on
sqlite> .mode csv
sqlite> .output hw5_original.csv
sqlite> select tracks.Name as TrackName,
        tracks.composer,
        tracks.milliseconds as TrackLength,
        tracks.bytes as TrackSizeBytes,
        tracks.unitPrice as TrackPrice,
        genres.name as Genre,
        media_types.name as MediaType,
        albums.Title as AlbumTitle,
        artists.Name as ArtistName,
        invoice_items.Quantity as InvoiceItemQuantity,
        invoice_items.UnitPrice as InvoiceItemUnitPrice,
        invoices.invoiceId,
        invoices.InvoiceDate,
        invoices.BillingAddress as InvoiceBillingAddress,
        invoices.BillingCity as InvoiceBillingCity,
        invoices.BillingState as InvoiceBillingState,
        invoices.BillingCountry as InvoiceBillingCountry,
        invoices.BillingPostalCode as InvoiceBillingPostalCode,
        customers.FirstName as CustomerFirstName,
        customers.LastName as CustomerLastName,
        customers.Address as CustomerAddress,
        customers.City as CustomerCity,
        customers.State as CustomerState,
        customers.Country as CustomerCountry,
        customers.PostalCode as CustomerPostalCode,
        customers.Phone as CustomerPhone,
        customers.Fax as CustomerFax,
        customers.Email as CustomerEmail
from tracks
left join invoice_items on tracks.TrackId = invoice_items.TrackId
left join media_types on media_types.MediaTypeId = tracks.MediaTypeId
```

```
left join genres on genres.GenreId = tracks.GenreId
left join albums on albums.AlbumId = tracks.AlbumId
left join artists on artists.ArtistId = albums.ArtistId
left join invoices on invoice_items.InvoiceId = invoices.InvoiceId
left join customers on invoices.CustomerId = customers.CustomerId;
```

- Then, I exported the resulting table as a CSV file with name `hw5_original.csv`.