PSTAT 10 Worksheet 7

Setup

```
library(RSQLite)
library(sqldf)
```

```
## Loading required package: gsubfn
```

```
## Loading required package: proto
```

```
library(DBI)
chinook_db <- dbConnect(SQLite(), "Chinook_Sqlite.sqlite")
dbExecute(chinook_db, "pragma foreign_keys = on")
```

```
## [1] 0
```

```
set.seed(1)
```

Problem 1

1. Primary and foreign keys are used to model relations between data sets. Specifically, they are used to link related observations in different data sets together to show that they have a real world connection.
2. Relational databases use relations between primary and foreign keys to represent relations between different sets of data. For example, a data set containing data on student grades would be related to a data set on personal information through a common key such as a student ID.
3. SQL is a standardized language used to work with relational databases. It can be implemented in different relational database systems, or RDBSs. SQLite is a lightweight, serverless RDMS that implements SQL-compatible syntax. It is an *implementation* of SQL in a RDBS. RSQLite is an R package that implements an interface between R and a SQLite implementation. It provides functions for R to run SQLite queries on SQLite databases.

Problem 2

```
dbListTables(chinook_db)
```

```
##  [1] "Album"         "Artist"        "Customer"      "Employee"
##  [5] "Genre"         "Invoice"       "InvoiceLine"   "MediaType"
##  [9] "Playlist"      "PlaylistTrack" "Track"
```

```
dbGetQuery(chinook_db, "pragma foreign_key_list(Album)")
```

```
##   id seq  table     from       to on_update on_delete match
## 1  0   0 Artist ArtistId ArtistId NO ACTION NO ACTION  NONE
```

```
dbGetQuery(chinook_db, "pragma foreign_key_list(Artist)")
```

```
## [1] id        seq       table     from      to        on_update on_delete
## [8] match
## <0 rows> (or 0-length row.names)
```

```r
dbGetQuery(chinook_db, "pragma foreign_key_list(Customer)")
```

```
##   id seq    table         from         to on_update on_delete match
## 1  0   0 Employee SupportRepId EmployeeId NO ACTION NO ACTION  NONE
```

```r
dbGetQuery(chinook_db, "pragma foreign_key_list(Employee)")
```

```
##   id seq    table      from         to on_update on_delete match
## 1  0   0 Employee ReportsTo EmployeeId NO ACTION NO ACTION  NONE
```

```r
dbGetQuery(chinook_db, "pragma foreign_key_list(Genre)")
```

```
## [1] id        seq       table     from      to        on_update on_delete
## [8] match
## <0 rows> (or 0-length row.names)
```

```r
dbGetQuery(chinook_db, "pragma foreign_key_list(Invoice)")
```

```
##   id seq    table       from         to on_update on_delete match
## 1  0   0 Customer CustomerId CustomerId NO ACTION NO ACTION  NONE
```

```r
dbGetQuery(chinook_db, "pragma foreign_key_list(InvoiceLine)")
```

```
##   id seq   table      from        to on_update on_delete match
## 1  0   0   Track   TrackId   TrackId NO ACTION NO ACTION  NONE
## 2  1   0 Invoice InvoiceId InvoiceId NO ACTION NO ACTION  NONE
```

```r
dbGetQuery(chinook_db, "pragma foreign_key_list(MediaType)")
```

```
## [1] id        seq       table     from      to        on_update on_delete
## [8] match
## <0 rows> (or 0-length row.names)
```

```r
dbGetQuery(chinook_db, "pragma foreign_key_list(Playlist)")
```

```
## [1] id        seq       table     from      to        on_update on_delete
## [8] match
## <0 rows> (or 0-length row.names)
```

```r
dbGetQuery(chinook_db, "pragma foreign_key_list(PlaylistTrack)")
```

```
##   id seq    table       from         to on_update on_delete match
## 1  0   0    Track    TrackId    TrackId NO ACTION NO ACTION  NONE
## 2  1   0 Playlist PlaylistId PlaylistId NO ACTION NO ACTION  NONE
```

```r
dbGetQuery(chinook_db, "pragma foreign_key_list(Track)")
```

```
##   id seq     table        from          to on_update on_delete match
## 1  0   0 MediaType MediaTypeId MediaTypeId NO ACTION NO ACTION  NONE
## 2  1   0     Genre     GenreId     GenreId NO ACTION NO ACTION  NONE
## 3  2   0     Album     AlbumId     AlbumId NO ACTION NO ACTION  NONE
```

The foreign key relations are:
Album.ArtistId -> Artist.ArtistId
Customer.EmployeeId -> Employee.SupportRepId
Employee.ReportsTo -> Employee.EmployeeId
Invoice.CustomerId -> Customer.CustomerId
InvoiceLine.TrackId -> Track.TrackId
InvoiceLine.InvoiceId -> Invoice.InvoiceId
PlaylistTrack.TrackId -> Track.TrackId

PlaylistTrack.PlaylistId -> Playlist.PlaylistId
Track.MediaTypeId -> MediaType.MediaTypeId
Track.GenreId -> Genre.GenreId
Track.AlbumId -> Album.AlbumId

Problem 3

1.

```r
dbGetQuery(chinook_db, "SELECT CustomerId, FirstName, LastName, State, Country
                        FROM Customer
                        WHERE State='CA'")
```

```
##   CustomerId FirstName LastName State Country
## 1         16     Frank   Harris    CA     USA
## 2         19       Tim    Goyer    CA     USA
## 3         20       Dan   Miller    CA     USA
```

2.

```r
dbGetQuery(chinook_db, "SELECT count(*)
                        FROM Customer
                        WHERE Country='Brazil'")
```

```
##   count(*)
## 1        5
```

There are 5 customers from Brazil.