HW #7

Complete By: Wednesday March 19th @ 9pm
Policy: Individual work only, late work not accepted
Submission: electronic via Blackboard (see below)

Background

You're going to write a database application to access the Netflix database we've been discussing in class. This application must be a Windows Forms Application, written using C#. You have reasonable freedom to design the app as you see fit, including the user interface and the database access code. The one restriction is that you must use the .NET ADO database objects (as shown in class on 03-12) to access the database; you may not use the built-in controls that automate database access and display, nor may you use LINQ or other technologies that simplify database access. You can use these tools to prototype and help you build the SQL queries, but the final app you submit for grading must access the database, and display the results, yourself.

Getting Started

To get started, let's create the Visual Studio project, install the database, and make sure you can open and close it… Startup Visual Studio, and create a new C#, Windows Forms Application project. Name it "NetflixApp". The next step is to add a reference to the appropriate database engine so you can open the Netflix database: *Project*, *Add References*, and select *Assemblies*. Expand *Extensions*, scroll, and find "*System.Data.SqlServerCe*" — select version 4.0.0.0, but also CHECK the box. If you do *not* have this library listed, then you need to install *Microsoft SQL Server Compact Edition 4.0*. You can do an internet search to find the download (it's a small 2MB file), or follow this link:

http://www.microsoft.com/en-us/download/details.aspx?id=30709

At this point you should have a reference set to the library component needed to access our Netflix database.

Minimize Visual Studio, browse to the course web page, and download the provided Netflix databases. The files can be found under HW7 at http://www.joehummel.net/cs341.html. Note that the files are in .zip format, so you'll need to extract and install into the bin\Debug sub-folder of your Visual Studio project. The "Netflix-65K.sdf" file contains roughly 65,000 reviews, while the "1MB" file version contains over 1 million reviews. Your program should work correctly with either file, in fact any Netflix database with same schema.

Back in Visual Studio, add a button to your form. Double-click the button to stub out an event handler, and then code the button to open and close the database, as shown in the lecture notes from Wednesday, 03-12. You'll need to add the following using statement to the top of your code-behind file:

```
using System.Data.SqlServerCe;
```

Once you can open and close the database successfully, you're ready to go!

The Assignment

Your assignment is to provide the following functionality. You are free to design whatever user interface you want for collecting the necessary input, and displaying the results. However, since you are probably new to both C# and GUI programming, “keep it simple” is a good rule of thumb :-) For some background and coding examples of using the common .NET user interface controls, see the lecture notes for Friday, 03-14. Here is the functionality you need to provide:

1. **Add Movie**: allow the user to add a new movie to the database. Report the success or failure of this operation, along with the newly-generated MovieID for the new movie. Note that to add a new movie, you need to compute a unique movie id before you can insert the movie; the typical approach is to use the SQL “MAX()” function to find the max existing MovieID, then add 1 to that value. The MAX function is used in a manner similar to the AVG function shown in the lecture notes.

2. **Add Review**: allow the user to add a new review based on the movie name; ratings are integers 1, 2, 3, 4, or 5, do not allow the user to insert any other value. Use whatever UserID you want, be creative :-) Report on the success or failure of this operation. Note: in this case the database is designed to auto-generate a unique ReviewID when you insert the review, so only insert 3 values as part of your insert statement: UserID, MovieID, and Rating.

3. **Average Rating**: provide the average review for a movie name entered by the user. Report on the success or failure of this operation; display the average rating if the computation is successful. Display the average rating with 2 decimal places as follows… Instead of using AVG(Rating) in your SQL query, use the following to cast the ratings to real numbers, compute the average, and then round the result to 2 decimal places: `ROUND(AVG( CAST(Rating AS Float) ), 2).`

4. **Each Rating**: allow the user to enter a movie name, and then display (if successful) the number of reviews found for each category, along with the total. In order words, a display along the lines of

   5: # of reviews
   4: # of reviews
   3: # of reviews
   2: # of reviews
   1: # of reviews
   Total: # of reviews

   Note: the best approach for obtaining this information is to use SQL’s “Group By” clause. An example of using “Group By” is shown in the next section.

5. **Top-N Movies by Average Rating**: display the top N movies by average rating, where N is a positive integer entered by the user (top 1, top 10, etc.). Display in descending order, movie name and average rating. You’ll want to use SQL’s “Group By” clause as discussed in the next section.

6. **Top-N Prolific Users**: display the top N users, based on the # of reviews submitted, in descending

order. Display both user id and the total # of reviews submitted by that user.

7. **Top-N Reviewed Movies**: display the top N movies, based on the # of reviews; in descending order. Display both the movie name and the total # of reviews of that movie.

SQL "Group By"

Most of the SQL you need has been presented in class, in particular lecture 03-10. The C# and ADO database code you need to execute this SQL is given in lecture 03-12. However, one feature of SQL that you will find very help is "Group By". Here's an example (this example works programmatically, but fails for some reason when typed into Visual Studio's Server Explorer query window):

```
SELECT TOP 10 MovieID, ROUND(AVG(CAST(Rating AS Float)), 2) as AvgRating
FROM Reviews
GROUP BY MovieID
ORDER BY AvgRating DESC;
```

This is the answer to functionality #5 :-) The 1st line computes the average rating to 2 decimal places, giving this column the name "AvgRating" in this query (and the resulting temporary table). However, is this the average of all ratings, or what exactly is this the average of? The 3rd line groups the data by MovieID, so that AVG will be applied to the ratings in each group — yielding the average of each movie. Finally, we sort these averages into descending order, and the "TOP 10" clause in line 1 limits the result set to the first 10. The result? The top-10 movies by average rating.

Electronic Submission

The first step is to create a .zip file / compressed folder of your *entire* Visual Studio project folder: this should be the folder that you created, probably called "NetflixApp". Then, using Blackboard, submit this .zip file / compressed under the assignment "HW7". We expect your C# code to be commented and readable, including a header comment at the top of your GUI code-behind file along the lines of

```
//
// Netflix database application.
//
// <<YOUR NAME HERE>>
// U. of Illinois, Chicago
// CS341, Spring 2014
// Homework 7
//
```

You may submit as many times as you want before the due date, but we grade the last version submitted.

Policy

Late work is not accepted. All work is to be done individually — group work is not allowed. Academic dishonesty is unacceptable, and all parties involved will be immediately subject to the official academic integrity review process. The University's policy is quite clear, and can be read here: http://www.uic.edu/depts/dos/studentconduct.html. In particular, note that you are guilty of academic dishonesty if you extend or receive any kind of unauthorized assistance. Examples of academic dishonesty include emailing your program to another student, copying-pasting code from the internet, working in a group on a homework assignment, and allowing a tutor, TA, or another individual to write an answer for you.