CSc 460 — Database Design
Spring 2017 (McCann)

http://www.cs.arizona.edu/classes/cs460/spring17

Program #3: JDBC

Due Date: April 5th, 2017, at the beginning of class

Overview: Embedding SQL within another programming language is nice for applications that require more complex calculations or manipulations than plain SQL can handle. Many DBMSes have add-ons that can be used for developing nice applications, but even they may not be flexible enough to create the application you have in mind.

As many of you know from being educated in Arizona, the state has used the "Arizona's Instrument to Measure Standards" (AIMS) test to assess educational progress of grade-school students in recent years. Starting with the 2010 year and continuing through 2014, the Arizona Department of Education posted school-by-school results on its web site as Microsoft Excel (`.xls`) files. It's easy to use a spreadsheet program to save those files to CSV (comma-separated value) text format, which is a good format for reading into a program, and from there inserting into a database.

Assignment: For this assignment, you will need to do the following, probably in this order:

1. Get the "AIMS and AIMS A" 2010, 2011, 2012, 2013, and 2014 Excel files from the Arizona State Dept. of Education web site (see **Data**, below, for the URL).
2. Convert the content of the "School" tabs to CSV files. (Each `.xls` file contains several spreadsheets; we're only interested in the first one, labeled with the concatenation of the year and "School".)
3. "Scrub" the files to make them consistent and suitable for importing into an Oracle database. (Again, see **Data**, below.)
4. Within your Oracle database, create five tables, one for each AIMS file, with appropriately named and typed fields.
5. Import the files' content into the relations.
6. Write a Java program that offers the user a menu of questions that can be asked of the database of AIMS data, uses JDBC to interact with the database to fetch the query results, and displays the results to the screen in a format of your choice. The questions are:
   (a) For a year provided by the user, how many High Schools are listed in the results? (You'll have to determine this based on the school names. Note that "Junior High" is not the same as "High".)
   (b) For each of the five years, display the number of charter schools and how many of them had a sum of the Math percentages "Falls Far Below" and "Approaches" that was less than the percent "Passing".
   (c) For each county in 2014, which 10 schools had the greatest differences between the "Passing" percentages in Reading and Writing? Display one table for each county that includes position number ('1' for the school with the biggest difference, and duplicate position numbers for ties), the school name, the reading and the writing percentages, and the absolute value of the difference. Order the tables alphabetically (ascending order) by county name; thus, Apache's table will be first. Each county's table should display the 10 schools in descending order by the absolute value of the difference. If ties mean that the table has to have more than 10 entries, let it have enough entries to include all of the ties. Here's some made-up example output that covers all of these situations:

(Continued...)

```
Aardvark County
===============
                                       Reading   Writing
  Pos  School Name                     Passing%  Passing%  |Difference|
  ---  ------------------------------  --------  --------  ------------
    1  ASU Graduate College               45         7          38
    2  Lincoln Elementary                 12        47          35
    2  Washington High                    45        80          35
    4  All-Day Recess                     39        11          28
    5  Rotting Floors Middle School       78       100          22
    5  Bryson Elementary                  50        28          22
    5  James Street Elementary            98        76          22
    8  Ceramic Magnet School              61        80          19
    9  John H. Watson Elementary          99        84          15
    9  Mean Median Mode Middle School      3        18          15
    9  Hippity Hooper High School         43        28          15
```

(d) A query of your choice, subject to these restrictions: The question must use data from at least three of the five relations, and must be constructed using at least one piece of information gathered from the user. If you have trouble thinking of a suitable question, envision yourself as a parent of a grade–school student and think about what you, as a parent, might like to be able to ask about school performance.

Data: The Excel files can be downloaded from this AZ DoE web page:

`http://www.azed.gov/research-evaluation/aims-assessment-results/`

The spreadsheets' format is consistent across the years, but there are some issues you will need to address before you can create relational tables from the data. (This is what we mean by "scrubbing" the data.) Some examples:

- The rows with column labels aren't actual data, and as such are not to be included in the tables.
- Many numeric cells in the spreadsheets hold asterisks instead of values. This means that the data isn't available, or exists but is generated from so small a group of students that it can't be reported. In your relations, represent such cells with null.
- Even allowing for asterisks, the spreadsheets have some rows that aren't complete; those rows shouldn't be inserted into the database.

These sorts of inconsistencies can be eliminated with some combination of creative (or exhaustive!) text editor skills, small data cleaning scripts of your own creation, and basic SQL statements. It's always tedious, but rarely difficult. As data goes, the AIMS results are quite clean.

The menu format, structure of the requests for user info needed by the queries, etc., is up to you.

Output: Your application is to display the output of a query in a clear, easy to read format of your choice (remember, you're writing your own program; you are not restricted to SQL's output format).

Hand In: You are required to submit a `.tar` file of your well-documented application program file(s) — including any code written to automate the data cleansing process, although that code need not be well-documented — via turnin to the folder `cs460p3`. Name your main application program's `main()` class `Prog3`, so that we don't have to guess which file to compile, but feel free to split up your code over additional files as appropriate for good code modularity.

(Continued...)

Want to Learn More?

- Some info about the AIMS results can be found in the PDFs on this page:
  `http://www.azed.gov/research-evaluation/aims-assessment-results/`
- Wikipedia has some basic info about AIMS and its history (because, of course it does!), which is good, because the AZ DoE site doesn't have much at all:
  `https://en.wikipedia.org/wiki/Arizona's_Instrument_to_Measure_Standards`

Other Requirements and Hints:

- Because we will be grading your program on lectura using Oracle, it needs to run on lectura and use Oracle.
- LibreOffice (installed on lectura) can save spreadsheets as CSV files.
- If you wish to share the data conversion and "scrubbing" chores with your classmates, that's fine. Stop collaborating when you start coding your application. In your documentation, be sure to credit those who helped you with the data organization. Please DO NOT post scripts, etc., on Piazza; we don't want one generous person doing all of the dirty work for the entire class!
- It is OK to share query results on Piazza. Doing so can help you discover that your query isn't finding everything it should be finding.
- Make certain that your database tables are accessible to us (by `GRANT`ing us `SELECT` privileges) and that your relations are prefixed with "`yourNetID.`" in your queries so that we can execute your program against your database, just as my tables for Homework #3 were accessible with the "`mccann.`" prefix.
- Avoid the temptation to wait to start writing the JDBC code and your application program until you have all of the data loaded into tables. You can (and should!) create and populate small tables for testing purposes early in the development process.