CSc 460 — Database Design
Spring 2026 (McCann)

https://cs.arizona.edu/classes/cs460/spring26/

Program #3: JDBC

Due Date: April 2nd, 2026, 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 windowed applications, but even they may not be flexible enough to create the application you have in mind.

When accidents between trains and highway traffic occur, Form 57 is filed with the U.S. Department of Transportation. The data, back to 1975, is then made available on their web site and can be downloaded as a CSV file. (See the Data section, below, for the URL.) It's straight–forward to write a program that reads CSV files and inserts the data into tables of a DBMS.

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

1. Get the four highway–rail incident CSV files for 1980, 1995, 2010, and 2025. I've done the data partitioning for you, and have provided the files on lectura. The Data section has the location.
2. As necessary, "scrub" the files to make them consistent and suitable for importing into an Oracle database. (Again, see the **Data** section, below.)
3. Within your personal Oracle database, create four tables, one for each year, with suitable table and attribute names.
4. Import the CSV files' content into the corresponding relations. This can be done with by writing a `.sql` script of `INSERT INTO` commands, or by writing a separate Java program to insert data using JDBC.
5. Write a Java program named `Prog3.java` that offers the user a simple text menu of queries that can be asked of the database of incidents, and uses JDBC and SQL to interact with the database to produce the correct answers (with the help of some additional Java processing, if needed/desired). The queries are:
   (a) How many incident reports were filed in each of the four years?
   (b) Given a year entered by the user, what are the top 10 states for most reported incidents in that year, sorted in descending order by quantity of incidents? For each state, display its name and its quantity of incidents.
   (c) Given two years entered by the user, which five states had largest percentage decrease in quantity of incidents in the later year as compared to the earlier year? For each state, display its name, the quantities of incidents in the two years (earlier year first), and the percentage decrease, with the five lines of data in descending order by percentage decrease. For example, if the State of Confusion reported 60 incidents in 1980 and 45 in 2025, the percentage decrease is 25.0%. If there aren't five such states, display all that showed a decrease.
   (d) A query of your choice, subject to these restrictions: The question must use data from multiple years, and must be constructed using at least one piece of information gathered from the user. If you have trouble thinking of a suitable question, take a look at the fields that the three previous queries didn't use, and use your imagination!

   For each of these queries, you may answer it using one or more SQL SELECT statements, with as much or as little additional Java processing as you wish or need to use.

(Continued...)

Data: You can find the 1980 CSV file on lectura here:

```
/home/cs460/spring26/highwayrail1980.csv
```

Just change the year to get the 1995, 2010, and 2025 files.

The CSV format should be consistent across the years, but there are some issues you will need to address before you can create relational tables from the data. For example, you'll have to delete/skip the metadata on the early rows. For another, some values may be missing (no data) for some fields. Use NULL in your tables to show such missing info. You may well find additional issues. If so, ask us about them; we'll let you know what to do.

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 answer to 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 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.

Want to Learn More?

- The Highway–Rail Grade Crossing Incident Data page's URL is:
  https://data.transportation.gov/Railroads/Highway-Rail-Grade-Crossing-Incident-Data-Form-57-/7wn6-i5b9/data_preview
  I've linked to it from the class web page.
- You can find an example JDBC program on the class web page:
  https://www2.cs.arizona.edu/classes/cs460/spring26/JDBC.java

Other Requirements and Hints:

- Because we will grade your program on lectura using Oracle, it must run on lectura and use Oracle.
- If you wish to share any necessary data pre–processing, "scrubbing," and/or script–writing 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 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 (or is finding more than) it should be finding.
- (IMPORTANT!) 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 application so that we can execute your program against your database, just as my tables for Homework #3 were accessible with the "`mccann.`" prefix. The form of the `GRANT` command is:

  ```
  GRANT SELECT ON tablename TO PUBLIC;
  ```

  If you need to delete and re–create your tables, you'll need to re–issue the `GRANT` commands on the new tables.
- 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.