Project #5: Design a College Database

1 Design a Database (10 points)

Create an initial database design for Cascade Community College (CC). They want to track many important entities related to their learning environment. Use the details here to help you with your design decisions:

- Courses. Courses have a school-assigned designation that is at maximum seven characters long, including both letters and numbers, e.g., MATH102. Courses also have a descriptive name and a number representing how many credit hours the course represents, e.g., 5 (5 hours per week).
- Departments. The school has many different departments, each having a specific name, e.g., "Math and Science." Departments manage and offer many courses, but never share courses, i.e., MATH102 is managed solely by the Math and Science department.
- Sections. In each quarter (Fall, Winter, Spring, or Summer of a specific year), a course may be offered as a section. Each section is offered as either Day, Night, or Online.
- Textbooks. Each course may require one or more textbooks. All sections of the course use the same textbook. Each book has an ISBN (10 digits), a title, and an associated author. Some textbooks serve more than one course, e.g., two courses in programming sequences rely on the same book.
- Students. The College maintains a student ID (eight digits), plus a last name, first name, and email address. Students may take one or more sections each quarter.
- Instructors. Like Students, Instructors have an ID (eight digits), a last name, first name, and email address. Instructors may teach one or more sections each quarter. Sections are taught by a single instructor only.
- Parking Passes. For a given quarter, students may hold a quarterly parking pass.

Create a database diagram (MySQL EER) showing all tables necessary to support this effort. Choose data types wisely. Show the relationships between the tables, identify primary and foreign keys, etc. Organize the diagram such that the layout helps spell out the relationships and make clear what tables are central to the data model. Save the model and be prepared to submit it.

2 Submitting Your Work

Submit your model file (.mwb) in Canvas.

3 Check Your Work

Testing a design is hard; it's easy to see what's there but hard to see what's *missing*. Take some time to walk through the diagram, thinking about different scenarios and queries. Here are some hints about how to verify what you've done:

- **Model vs. Handout**—looking at your model, read over the handout in detail, perhaps highlighting or marking off sentences as you go. Many sentences will imply relationships or give details about what attributes should be stored for each entity. Some sentences will have multiple hints within them. Make sure that everything you read in the handout has properly "landed" in your model.
- **Sample Data**—on paper, or in some generic tool like Excel or Word, create a table for each of the tables in your model, then populate them with a tiny bit of data, just enough so you get the flavor of

what will live there. If the data doesn't seem right when you look at it, and when you compare it across tables, then follow your intuition and see if there's a problem.

- **Sample Queries**—think about the kinds of questions that a user of the database would want to ask, and how they'd formulate queries to get it. You don't have to write full SQL queries, but use what you know about relationships to see if they are possible and plausible. If your model makes sense, users should be able to answer nearly any question they have about the scenario.
- **Normalization**—using the basics of what we learned, look at the obvious signs that normalization has not been fully realized. Look for repeated columns that contain similar data, repeated information sitting within a single column, data that doesn't depend on the "PK and only the PK," etc.
- **Relationships**—look again at the relationships between pairs of independent tables, and consider very carefully whether one-to-many or many-to-many relationships are needed. And don't make assumptions here or impose your own view of the world on your model; make sure that you're following the user's requirements as laid out in the handout. Some students will go crazy in one direction or another, thinking that everything must be 1:m or n:m, but it's not always one way or another.

4 Hints

- Be careful to model the relationships that are discussed in the handout, not relationships you *imagine* exist. If you don't find support for the relationship in the document, be wary of including it.
- Save your work. Save it often, and save it in versions. This is especially true if you decide on a major redesign; you want to be able to go back to what you had before, should you change your mind!
- The bulleted list in the first section isn't necessarily a full list of the tables you need. Others will be needed to support the obvious tables. You may also discover the need for more tables, as you work through your design.

5 Grading

For designs, you'll be graded on whether the design incorporates all the elements discussed in the specification and implements them in a reasonable way. I'm looking for completeness and correctness. There may be variations in approaches, however; I'm not expecting your work to be a clone of mine.