Assignment 1

Problem 1. Normalization (50 pts).

This problem will use the file techcrunch.csv. This dataset consists of company funding records reported by TechCrunch. Each row represents one funding event for a company.

1) (10 pts) What is a good choice for a primary key here? In contrast, give an example of an attribute (or composite) that would *not* be a valid primary key.
   Be careful. You can check that your proposal really satisfies the definition of a primary key by sorting your data by the relevant column(s) in Excel.
2) (10 pts) For your choice of primary key, do the data satisfy 1NF? Why or why not?
3) (10 pts) For your choice of primary key, do the data satisfy 2NF? Why or why not?
4) (10 pts) For your choice of primary key, do the data satisfy 3NF? Why or why not?

To get full credit, if the dataset fails to be in any of the normal forms above, you should document *all* the ways in which it so fails. For example if you think the data are not in 1NF, it does not suffice to say "the data aren't in 2NF because you can't be in 2NF without being in 1NF".

5) (10 pts) Sketch a proposed Entity-Relationship diagram that would bring this dataset into 3NF. If you answered "yes" to (4), for example, your ERD would just be the raw data table with no changes.

   If however your ERD requires multiple tables to be in 3NF, you should draw all relationships between them and indicate their type (one-to-one, one-to-many, etc.) Explain why your proposal satisfies all the requirements of 3NF.

Problem 2. Case study (adapted from Comeau, Chapter 9) (50 pts)

This question is designed to exercise your ability to use JOIN statements to "undo" the often-complex relationships that result from normalization. We will use the Recipe Database case study from the textbook. The associated .sql file contains code that creates a new schema and populates it with associated tables and a small amount of example data.

1) (10 points) The database is mostly empty. Use the INSERT INTO statement to insert information about two (2) complete recipes of your choosing into the database. You can make up your own recipes or copy them from a website.
2) (25 pts) Write a set of queries to return *all* information on a specific recipe including main details, ingredients, recipe tags, nutrition, comments, food warnings and any available substitutions. Use as few queries as possible. Your set of queries should be designed to create something similar to Figure 9.14 in the textbook—a complete recipe page (in that case, Chicken Marsala) that might appear on a website. Remember to use aliases on the field names that are returned so that the raw query results will be more readable.

   Show the results of your set of queries for one (1) recipe of your choosing.

3) (15 pts) Write a SELECT query that would supply the information that might be used in the back of a cookbook (the book's index). Specifically, your results should be the recipe name, category, and all tag values. Your output should be sorted first by tag value and then by category. Show the results of your query on **all** data in your db.

To achieve full credit, you should not merely show your code, but explain it as well. In particular, if you use JOIN statements, explain why you are using your particular type of JOIN (LEFT, RIGHT, INNER, OUTER, etc.).