University of Toronto
CSC343, Summer 2019

Assignment 3

Due: Monday, August 5, 2019, before 10:00pm

You may work with a partner. Note that your submission must be **typed** and not handwritten. More details are in the Submission Instructions section at the end of the handout.

Database Design and SQL DDL

1. Consider the relation $R(A, B, C, D, E, F)$.
   Let the set of FD's for R be $\{A \to B, CD \to A, CB \to D, CE \to D, AE \to F\}$.

   (a) What are all of the keys for R?

   (b) Do the given FD's form a minimal basis? Prove or disprove.

   (c) Provide a decomposition of R into 3NF-satisfying relations.

   (d) Are any of the relations that you made in part (c) not in BCNF?

2. Answer the following questions.

   (a) Prove or disprove the following:
   Suppose a relation R is decomposed into R1 and R2 with one common attribute between the two new relations.
   If the common attribute between R1 and R2 forms a key for at least one of R1 or R2, then the decomposition is lossless.

   (b) Can a relation and a set of FDs be in both BCNF and 3NF at the same time? If so, explain what conditions must be met. If not, explain what is preventing this from being possible.

3. Prove or disprove that:

   (a) If $A \to B$ then $B \to C$

   (b) If $AB \to C$ then $A \to C$ and $B \to C$

4. **Design and DDL**
   Consider the following domain:
   You are running a Fresh Juice business with multiple stores around the country, and you want to keep the information for these stores in a relational database. The following is a list of that information:

   - Stores: Each store has a city, phone number, and manager. There is only one store per city.
   - Beverages: Each juice beverage has a name (e.g. 'Kiwi Lime'), and a number of calories for a regular and large size. A large size always has 200 more calories than the regular size. Every store should keep track of the number of inventory of each beverage (how much it has left in stock).
   - Transactions: When a customer makes an order, that order should have a date, price, and an indication of which loyalty card was used, if applicable. You can assume one beverage is ordered per transaction, and we should know what that beverage was.
   - Loyalty card: Customers can have a loyalty card if they like to go to your stores a lot. There needs to be information on how many transactions a customer made with the card, and their 'home store' (the one they go to most frequently).

Continued on next page...

(a) Define a **single** relation for this domain that manages to store all of the required information (just write it out $R(...)$, no need for SQL definitions yet). There is not necessarily one correct answer for this relation, but the information should be stored in a practically useful way. It is ok to add attributes that aren't explicitly listed in the domain as long as they are useful.

(b) Write all of the functional dependencies for your relation that would be inferred by the description of this domain. Do not include trivial or redundant FDs (find a minimal basis).

(c) Provide a useful instance of your relation that shows all three types of anomalies. Describe the anomalies you have presented as they appear in your particular instance (give an example for each of the three anomalies in your relation).

(d) Your relation will likely (read certainly) have some redundancy. Decompose your relation into a set of relations without any BCNF violations. Write all of your steps in full and clearly show why your relations do not violate BCNF.

(e) Explain how your new relations prevent the anomalies you pointed out in part (c).

(f) **SQL DDL:**

Using your new relations from part (d), create a schema using the SQL DDL language. They should have proper relation names, attribute names and types, and constraints (including keys, foreign key, unique, not null, etc.).
You should add **comments** above each table and attribute describing what it represents. You can add comments using a double dash `--`. You should also insert some useful data into each of the relations (directly in the ddl file, not through csv).
Use the DDL files from the lectures and A2 as examples to help you make them. Unlike A2, you do not need to write any SQL queries for this part.

What to hand in for this part:
In your A3.pdf file, describe the decisions for any constraints you put in your DDL file. Hand in a file called `fruits.ddl` containing your schema, as well as a plain text file called `fruits-demo.txt` that shows you starting postgreSQL, successfully importing fruits.ddl, and exiting posgreSQL. This is similar to what you did in the preps. You must hand in this demo and the file must be a plain text file or you get **zero** for this part of the assignment.

Submission instructions

Your assignment must be typed; handwritten assignments will not be marked. You may use any word-processing software you like.

For this assignment, hand in a file A3.pdf that contains your answers to the questions above.
Also hand in `fruits.ddl` and `fruits-demo.txt`.

You must declare your team and hand in your work electronically using the MarkUs online system. Well before the due date, you should declare your team and try submitting with MarkUs.