NYU School of Engineering February 3, 2020
Computer Science and Engineering
CS 6083, Spring 2020

# Problem Set #1 (due 2/17)

Note: In this homework, you do not need to create tables and execute queries using an actual DBMS. A written solution is sufficient.

**1**. Suppose you have a database for modeling a video streaming service (similar to Netflix), where customers can watch movies. It is given by the following schema:

Customer (<u>cid</u>, cname, ccn, ccity)
Movie (<u>mid</u>, mtitle, mlength, genre, year)
Watch (<u>cid, mid, starttime</u>, endtime)
  cid references Customer.cid, and mid references Movie.mid
Rating (<u>cid, mid</u>, score, comment)
  cid references Customer.cid, and mid references Movie.mid

In this schema, the Customer table stores the customer ID, name, credit card number, and city of a customer, while the Movie table stores the movie ID, title, length, genre (e.g., Science Fiction or Drama), and year of release of a movie. The Watch table has an entry for every time a customer watches a movie, with the cid of the customer, the mid of the movie, and the time and date when they started and ended watching the movie. Rating stores movie ratings by customers, where score is a value between 0 (hated it) to 5 (great movie), and comment is some short review or comment that the customer provided.

(a) Write SQL queries for the following

- Output the cid and name of the customer(s) who has spent the most time watching movies.
- Output the cid of any customer who has watched every movie (if such a person exists).
- Output the mids of all movies that were watched last week.
- Output the cids of all customers who stopped watching ``The Piano'' after less than 10 painful minutes.
- Output the mid and title of any movie that has not been watched by even a single customer.
- Output the mid and name of the movie(s) that had the highest average rating among all movies.
- Output the mid of the other movie(s) that received the highest average rating from those people who gave a score of 5 to the movie ``The Piano''.

(b) Write statements in Relational Algebra for all seven queries. Use basic RA whenever possible, and extended RA otherwise.

(c) Write statements in (Domain or Tuple) Relational Calculus for the first five queries, or explain if it is not possible to do so.

**2**. In this problem, you have to design a relational schema for a web-based service, similar to Amazon Subscribe and Save, where people can subscribe to certain products and receive them on a regular schedule, say every month or every three months. People can also make changes to this schedule as they choose. Moreover, whenever they purchase at least a certain number of items in one delivery, they get a discount on the price.

Thus, there are customers identified by a customer ID, with a name, shipping address, credit card info, etc., and there are products with a product ID, a product type, and product name, description, and a current price. Customers can subscribe to a product, and specify how often they want to receive the product, say every month, or every second or every fifth month, etc. Every month, the company will then check which products should be sent to the customer during this month, and store this information as one *delivery order*, say for March 2020. (For simplicity, we do not specify a delivery date, as in Amazon, but just a month, so that the items could arrive any time during the month.) There is also a set of rules about discounts that the customer receives when they get a certain number of items in one month. For example, a customer getting at least 3 items might get 5% off, for at least 5 items they might get 8% off, and for at least 10 items they might get 10% off. These rules about discounts might be changed, and they should be stored in the database, not hardcoded into queries. Note that once the delivery order is created, the prices of the items (before discount) are fixed for this delivery; thus, subsequent changes to the price stored in the product table should not be propagated into a pending delivery order even if it has not be dispatched yet.

The customer will receive an email a few days before the delivery is actually dispatched, so that they can make changes to the next delivery, such as delaying an item by one month, or adding an item that was not due yet to the delivery. You do not have to worry about how the customer can add and remove items from a delivery, or how the email is sent – those are user interface issues separate from the relational design. However, when such a change happens, you need to make sure this is represented in your database. Note also that such changes impact future deliveries – if an item was supposed to be delivered in April, and is on a 3-month schedule, then moving it up to March also moves the next delivery up from July to June, and so on. In general, when an item is on a 3-month schedule, then its next delivery should be 3 months after the previous delivery was done.

(a) Design a relational database schema for this application that supports the above functionality. Specify all primary and foreign key constraints, and state any assumptions you are making. You can decide which exact attributes make sense for this schema.

(b) Write SQL statements for the following queries. If your schema does not support these, you need to modify it appropriately. (For this first homework, you may use informal expressions such as year(ts) = ``2018'', where ts is a timestamp, to check if the year is 2018.)

I. For a particular customer, say the customer with ID 7184995, output for each subscription by that customer the month and year when the item should be delivered the next time.

II. Output the user with the largest number of subscriptions.

III. Output the ID and names of all users who have a subscription to "Acme Rocket Powered Roller Skates", and who moved their delivery of this item up from March to February 2020.

IV. For each product, output the total number of times it was delivered in 2019, and the total amount of money that was paid for it.