**University of Toronto**
**CSC343, Summer 2019**

# Assignment 1

**Due**: Saturday, June 8, 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.

## Introduction

For this assignment, you will be writing queries in relational algebra on a database. We are providing the schema for you. Later in the course, you will learn about how to develop your own schema based on knowledge of the domain. Even though developing a schema is really the first step in building a database, it is a more advanced task than querying an existing database; this is why you will be learning about it later.

## Domain

The schema below represents data for restaurants in the city, where visitors, or 'patrons', of the restaurant can make reservations, order dishes, and leave ratings for the restaurant. It attempts to represent the domain well, but in order to keep the assignment manageable, some things are simplified and other details are not represented. Remember that all your queries will be written with respect to the schema below. Your goal is to produce answers that are correct with respect to this schema, even if they arent quite correct in the real world.

## Schema

### Relations

Restaurant(name, owner, capacity, country)

Patron(PID, fname, lname, birthday)

Dish(DID, name, dietary)

Reservation(RID, PID, rname, date)

Order(RID, DID, number)

Rating(PID, rname, rating)

- Restaurants: This relation stores the name of the restaurant, its owner, and its capacity (maximum number of people it can hold at one time).
- Patrons: Patrons are the people who visit the restaurant. *PID* is the Patron's ID. This relation also stores their full name and birthday. Dates are stored without the year (i.e. '0518' for May 18).
- Dish: A certain menu item that a restaurant can offer. Different restaurants can offer the same dish. *DID* is the Dish ID. This relation also stores the name of the dish, and any dietary restrictions. The *dietary* field is *'none'* if there are no restrictions for this menu item, but can be something like *'vegetarian'* or *'vegan'*.

- Reservations: This relation holds the reservation information for the restaurants. *RID* is the reservation ID. Each reservation has a Patron who made the reservation, and the date of the reservation (same date format as birthdays for Patrons - you can assume the database only holds data for the current year's reservations).
- Order: Tuples in this relation represent an order of a specific dish for a particular reservation. *number* is the amount of that dish ordered (for example, you can order 4 salads if there are 4 people at your table).
- Rating: Tuples in this relation represent the rating that Patrons give to a restaurant with a certain name. Patrons can only give one rating to a restaurant. Also, Patrons could give a rating to a restaurant despite not going to it (think of how online review sites allow anyone to rate a restaurant).

**Integrity constraints**

- Reservation[PID] $\subseteq$ Patron[PID]
- Reservation[rname] $\subseteq$ Restaurant[name]
- Order[RID] $\subseteq$ Reservation[RID]
- Order[DID] $\subseteq$ Dish[DID]
- Rating[PID] $\subseteq$ Patron[PID]
- Rating[rname] $\subseteq$ Restaurant[name]

## Part 1: Queries

Write the queries below in relational algebra. There are a number of variations on relational algebra, and different notations for the operations. You must use the same notation as we have used in class and on the slides. You may only use assignment, and the operators we have used in class: $\Pi, \sigma, \bowtie, \bowtie_{condition}, \times, \cap, \cup, -, \rho$. Assume that all relations are sets (not bags), as we have done in class, and do not use any of the extended relational algebra operations from Chapter 5 of the textbook (for example, do not use the division operator or aggregation).

Some additional points to keep in mind:

- Do not make any assumptions about the data that are not enforced by the original constraints given above. Your queries should work for any database that satisfies those constraints.
- Assume that every tuple has a value for every attribute, in other words, there are no null values.
- The condition on a select operation can use comparison operators (such as $\leq$ and $\neq$) and boolean operators ($\vee$, $\wedge$ and $\neg$). Simple arithmetic is also okay, `e.g.`, attribute1 $\leq$ attribute2 + 5000.
- You may assume that all attribute values are ordered so they can be compared with $=, <, >, \leq, \geq, etc.$
- You are encouraged to use the **assignment operator** (:=) to define intermediate results.
- To get **full marks**, you should **add commentary** explaining what you're doing. This will also help you get partial marks even if your final answer is not completely correct.
- The order of the columns in the result doesn't matter.

- When asked for a maximum or minimum, if there are ties, report all of them.
- If you believe that a query **cannot be expressed** in the Relational Algebra language that you are using, then, simply write "cannot be expressed". Later in the course, when we learn SQL, it will be interesting to consider whether they can be expressed in SQL.

Note: The queries are not ordered by difficulty.

1. Report the name of the Patron that has given the highest rating to a restaurant. If there are ties, report all of them.
2. Report the name of the restaurant for which the highest number of reservations were made. If there are ties report all of them.
3. Report the PID(s) of the Patrons(s) who reserved a spot at a restaurant, but did not order anything.
4. Report the name(s) of the Patrons(s) who have made a reservation to the restaurant named 'Boston Pizza' and ordered 3 of a dish called 'Margherita Pizza'.
5. Report the owner of the restaurant with the highest average rating. If there are ties, report all of them.
6. Report the capacities of the restaurants from which patrons have so far only ordered foods with a 'gluten-free' dietary restriction.
7. Report the restaurant owner for which the very earliest reservation out of all the reservations in the database was made. Report any ties.
8. Report the PID(s) of the Patrons who have made reservations to the restaurant named 'Red Lobster' on their birthday.
9. Consider all patrons that have made reservations to at least two different restaurants. For each of those patrons, report their name, and the names and ratings of all of the restaurants they went to (not ones they rated without actually going to).
10. Report the name of all Restaurants that had reservations made on every day that someone made a reservation at the restaurant named 'Pickle Barrel'.

Make sure you remember to add commentary explaining what you did and your intermediate results.

## Part 2: Integrity constraints

Express the following integrity constraints with the notation $R = \emptyset$, where $R$ is an expression of relational algebra. You are welcome to define intermediate results with assignment and then use them in an integrity constraint.

1. A restaurant owner can only own one restaurant.
2. Patrons who did not make a reservation for a restaurant cannot review it.
3. A Patron cannot make multiple reservations in one day for a restaurant that has a capacity less than 100.

## Submission instructions

Your assignment must be typed; handwritten assignments will not be marked. You may use any word-processing software you like. Many academics use LaTeX. It produces beautifully typeset text and handles mathematical notation well. If you would like to learn LaTeX, there are helpful resources online. Whatever you choose to use, you need to produce a final document in pdf format.

You must declare your team and hand in your work electronically using the MarkUs online system. Instructions for doing so will be posted on the Assignments page of the course website. Well before the due date, you should declare your team and try submitting with MarkUs.

For this assignment, hand in just one file: A1.pdf. If you are working in a team, only one of you should hand it in. Check that you have submitted the correct version of your file by downloading it from MarkUs;