Introduction

The goal of this assignment is to improve your skills of writing declarative queries on a relational database, in general, and also to improve your fluency in SQL (and SQLite). You have been provided with the following relational schema.

- *members(email, name, phone)*
- *cars(cno, make, model, year, seats, owner)*
- *locations(lcode, city, prov, address)*
- *rides(rno, price, rdate, seats, lugDesc, src, dst, driver, cno)*
- *bookings(bno, email, rno, cost, seats, pickup, dropoff)*
- *enroute(rno, lcode)*
- *requests(rid, email, rdate, pickup, dropoff, amount)*

The tables are derived from the specification of Assignment 1 and the names of the tables and columns should give the semantics, except minor differences which are explicit in table definitions, insert statements or queries.

Creating the database

Using the SQL statements provided, create the above tables in SQLite3 on Lab machines with some data. Here is a small initial data to get you started.

Queries

Write down the following queries in SQL and run them in SQLite3 over the database created. You will be writing ONE SQL statement for every query (here One SQL statement starts with a SELECT and ends with a semicolon but may include multiple select statements combined in the form of subqueries and/or using set operations). Your SQL queries for questions 1-3 cannot use any of aggregation, grouping, or nesting (set operations are ok).

1. Find (name and email of) members who have at least two cars and at least one car is associated with a ride.
2. Find (name and email of) members who have cars and bookings but have not offered any ride.
3. Find (email of) members who are booked on a ride from *Edmonton* to *Calgary* in *November 2018*. *Hint*: Check out the date and time functions in SQLite.
4. Find requests that are served by the rides offered. A request is served by a ride if the dates are the same, the ride price is not greater than the requested amount, and the pickup and the drop off cities and provinces are also the same (the street addresses may not be the same). For each qualifying request, list the rid of the request, the requester's email, the location codes for the requested pickup and drop-off locations, and the rno of the rides that serve it.
5. Find top 3 destination cities with the largest number of rides. For the qualifying cities, list both the province and the city. *Hint*: Check out the *limit* clause for SQLite.
6. For every city, list the city, the province it is in, the number of rides from, the number of rides to, and the number of rides enroute that city. Include every city where at least one of the counts is non-zero. *Hint*: you may find outer join useful.
7. Find (the rno of) the cheapest ride(s) from *Edmonton* to *Calgary* in *October 2018* with available seats (meaning not all seats are booked).
8. Find (email of) members who have offered rides to more than half of the locations in *Alberta* and all their rides take place in *2016* or after.
9. Create a view called *ride_info* with columns *rno, booked, available, rdate, price, src*, and *dst*. The view includes for each ride in future, respectively the rno of the ride, the number of seats booked, the number of seats available, the date, the price, the source city and the destination city.
10. Using the view created in Q9, find all ride(s) from *Edmonton* to *Calgary* in *December 2018* with available seats (meaning not all seats are booked). For

each qualifying ride, list all information in the view, the email of the member offering the ride, and the number of days from the date of the ride to *Jan 1, 2019*. Sort the result on price from the cheapest to more expensive ones.