CS 775/875 HW 2 2/28/2017 Due date: 03/24/2017

Instructions:

1. Electronic submission: You assignment is **due by 11:00 PM, 3/24/2017**.

   The submit instructions are posted on MyCourses with the assignment file and sample database source file.

2. Download $buscompanies.sql$ from MyCourses.

   This source file provides the detailed schema for each table in the database (CREATE TABLE statements). I recommend creating your own test data for each question.

   Check out lab1 for instructions on how to log into your mysql account.

   Please note that the TA will use a different instance of the buscompanies database while grading. Your grade is based on how well your queries perform against this testing database.

3. Late policy: 1 day late: 2 points off, 2 days late: 5 points off, 3 days late 10 points off.

4. The relevant reading material is from Chapters 6, 7, and class notes.

Description:
Write SQL queries for a buscompany database. The database consists of the tables listed below. Primary key attributes are underlined.
NOTE: All table names, attribute names and data values are entirely in lowercase.

Tables (Entity Sets):
tripinfo (code, company, tripnbr, segnbr, origin, destination, duration, tripmiles)
departure (code, date, busnbr)
passenger (ssn, name, phone, earnedmiles)
crew (ssn, name, phone, job, salary)

Tables (Relationship Sets):
(passenger) reservations (ssn, code, date, seat)
(crew) schedule (ssn, code, date)

Notes about the database:

- Buses have one or more segments. A non-stop bus trip is considered a trip with only 1 segment.
- A candidate key for tripinfo is (company + tripnbr + segnbr).
- Each segment is numbered consecutively (from 1) and it has a city of origin, a destination city, duration of travel, and the number of trip miles.
- Constraints include that each segment originates from the destination of the previous segment.

- The database includes crew member duty scheduling in table schedule.
- Any crew member can also be a passenger (on departures other than the ones s/he works).
- All queries regarding tripinfo refer to just codes (do not check dates); all queries regarding departure refer to code + date.

Queries

For each of the following queries, write the SQL SELECT statements required to produce the desired output. At the end of each query, you will find a "result schema," which details the attributes that should be displayed by the query and their order.

1. (3) Retrieve the names of bus companies (without duplicates) that offer service in cities whose names start start with 'b' (example boston). The city could be an origin or destination.

   Result has schema (company).

2. (3) Retrieve names of all crew members and tripinfo they register for as passenger, NULL otherwise. Order results in ascending order by name.

   Result has schema (ssn, name, code, date).

3. (3) For each departure, retrieve names of assigned crew, NULL if no crew assigned.

   Result has schema (code, date, ssn, name).

4. (3) Retrieve the names of passengers that are traveling on 'peterpan' on May 19, 2017.

   Result has schema (name)

5. (3) Retrieve the average, total, minimum, and maximum salaries for each job in the database.

   Result has schema (job, avgSalary, totalSalary, minSalary, maxSalary).

6. (5) Retrieve tripinfo with no reservations.

   Result has schema (company, tripnbr, segnbr, date).

7. (5) Retrieve tripinfo with no non-crew reservations.

   Result has schema (company, tripnbr, segnbr, date).

8. (5) Retrieve the names of all bus companies and the total number of miles all of their buses will travel in the month of May 2017. If a company has not traveled in May, write NULL for total miles.

   Result has schema (company, miles)

9. (5) Retrieve the total number of miles each individual bus will travel for all upcoming trips.

   Result has schema (busnbr, miles)

10. (5) Retrieve bus trips whose last segment ends in 'miami'.

    Result has schema (company, tripnbr, segnbr, origin, destination, duration, tripmiles).

11. (5) Retrieve crew members who never arrive or depart from 'losangeles' either as passengers or on duty.

    Result has schema (ssn, name).

12. (5) Retrieve crew members who drive the same trip segments on which they are also passengers. (On different dates, of course, hence different departure dates.)

    Result has schema (ssn, name).

13. (7) Retrieve passengers who book entire bus trips (that is, they take every segment of the trip on the same date).

    Result has schema (ssn, name, company, tripnbr, date).

14. (7) Retrieve non-crew passenger pairs where the first passenger takes at least every departure the second takes.

    Result schema (ssn1, name1, ssn2, name2).

15. (7) Retrieve non-crew passenger pairs where the first passenger takes none of the departures that the second takes.

    Result has schema (ssn1, name1, ssn2, name2).

16. (5) Retrieve non-crew passengers and the tripmiles that they will earn based on current reservations.

    Result has schema (ssn, name, tripmiles).

17. (7) Retrieve total non-crew passenger tripmiles adding the new miles from current reservations to their existing (already earned) tripmiles. Print each non-crew passenger with total tripmiles.

    Result has schema (ssn, name, total tripmiles).

18. (5) For each departure, retrieve all passengers (NULL if no passengers). Order by code, date, name, ascending.

    Result has schema (code, date, ssn, name).

19. (7) Retrieve the driver who will drive the maximum number of miles in the current schedule.

    Result has schema (name, maxmiles).

20. (5) Retrieve passengers who take all departures in busnbr 'ABC123'

    Result has schema (name, ssn).

21. (5) Retrieve passenger who do not take departures in bus 'ABC123' nor 'XYZ123.'

    Result has schema (name, ssn).

22. (5) Retrieve passenger and their total earned miles if it is greater than 0. If a passenger has NULL miles or a 0 earned miles, then print NULL for totalmiles.

    Result has schema (ssn, name, totalmiles).

23. (5) Retrieve crew who never travel on bus 'ABC123' either as passengers or as crew.

    Result has schema (ssn, name)