CS 122A: Introduction to Data Management – Spring 2016, UC Irvine

Prof. Chen Li

Homework 4: More SQL (Hands-On) (100 points)

Due Date: **Thursday, May 12, 2016 11:45 PM, on EEE**

Submission

For this assignment, you need to create **a TXT file** to include **your SQL queries and results** and submit your file to the EEE dropbox. Points may be deducted if you don't follow the instructions. **Refer to the separate instruction**. All homework assignments should have the student IDs and names of your team members. Remember that all homework assignments should be done in a group. This homework assignment should be submitted on EEE before 11:45 pm on the due date. Only one student in a group should submit the file. Everybody on the team is required to have the finally submitted version. Refer to the following table for the submission guidelines. After the 24-hour grace period, no more submission is allowed on EEE. That is, we will **not** accept assignments after that time. We will publish the solutions at that time for the next assignment. Please get all your work in on time!

Date / Time	Place	Remark
Thursday, May 12, 2016 11:45 PM	EEE dropbox	Due date
Friday, May 13, 2016 11:45 PM	EEE dropbox	24-hour grace period - **10 points will be deducted**

More SQL [100 pts]

It's time to use your SQL skills to analyze the UCA database. Since we are going to use a real MySQL database, we will avoid certain limitations of the RelaX web service, e.g., it does not support "datetime" type .

Step 1: Install MySQL by following the instructions (for Windows, for OS X) on the class Web site.

Step 2: **Create and populate the database using the provided script with** the following schema:

1. **Airplane** = {registration_number:VARCHAR(10), model_number:VARCHAR(10), purchased_year:INTEGER, manufactured_year:INTEGER,capacity:INTEGER}
2. **Airport** = {IATA_code:CHAR(3), name:VARCHAR(40), airport_city:VARCHAR(20), airport_state:VARCHAR(20)}
3. **Customer** = {cid:INTEGER, ssn:CHAR(9) , gender:VARCHAR(6), email:VARCHAR(30), address_street:VARCHAR(50), address_city:VARCHAR(20), address_state:VARCHAR(20), address_zipcode:CHAR(5)}
4. **Credit_Card** = {cid:INTEGER, card_number:VARCHAR(20), expr_date:CHAR(6)}
5. **Flight** = {flight_number:VARCHAR(8), projected_departure_datetime:datetime, projected_arrival_datetime:datetime, aiplane_registration_number:VARCHAR(10), departure_airport_IATA_code:CHAR(3), actual_departure_datetime:datetime, arrival_airport_IATA_code:CHAR(3), actual_arrival_datetime:datetime}
6. **FlightAttendant** = {faid:INTEGER, phone_number:VARCHAR(20), birthdate:date, ssn:CHAR(9), job_title:VARCHAR(50), address_street:VARCHAR(50), address_city:VARCHAR(20), address_state:VARCHAR(20), address_zipcode:CHAR(5), service_year:INTEGER}
7. **Lounge** = {lid:INTEGER, location:VARCHAR(50), airport_IATA_code:CHAR(3)}
8. **MaintenanceEngineer** = {meid:INTEGER, phone_number:VARCHAR(20), birthdate:date, ssn:CHAR(9), job_title:VARCHAR(50), address_street:VARCHAR(50), address_city:VARCHAR(20), address_state:VARCHAR(20), address_zipcode:CHAR(5), skill:VARCHAR(50)}
9. **OperationStaff** = {osid:INTEGER, phone_number:VARCHAR(20), birthdate:date, ssn:CHAR(9), job_title:VARCHAR(50), address_street:VARCHAR(50), address_city:VARCHAR(20), address_state:VARCHAR(20), address_zipcode:CHAR(5), department:VARCHAR(50)}
10. **Pilot** = {pid:INTEGER, phone_number:VARCHAR(20), birthdate:date, ssn:CHAR(9), job_title:VARCHAR(50), address_street:VARCHAR(50), address_city:VARCHAR(20), address_state:VARCHAR(20), address_zipcode:CHAR(5), since:VARCHAR(50)}
11. **Customer_Reserves_Flight** = {cid:INTEGER, flight_number:VARCHAR(8), projected_departure_datetime:DATETIME, purchased_datetime:DATETIME, purchased_price:DECIMAL(7,2), quantity:INTEGER}
12. **Dish** = {lid:INTEGER, name:VARCHAR(40), price:DECIMAL(6,2)}
13. **DishOrder** = {oid:INTEGER, cid:INTEGER, lid:INTEGER, order_datetime:VARCHAR(10), total_amount:INTEGER}
14. **DishOrder_Contains_Dish** = {oid:INTEGER, lid:INTEGER, name:VARCHAR(40), quantity:INTEGER}
15. **Customer_Reserves_Flight** = {cid:INTEGER, flight_number:VARCHAR(8), projected_departure_datetime:datetime, purchased_datetime:datetime, purchased_price:decimal(7,2), quantity:INTGER}
16. **FlightAttendant_Participates_Flight** = {faid:INTEGER, flight_number:VARCHAR(8), projected_departure_datetime:datetime}

17. **MaintenanceEngineer_Maintains_Airplane** = {meid:INTEGER, Aiplane_registration_number:VARCHAR(10)}
18. **Pilot_Operates_Flight** = {pid:INTEGER, flight_number:VARCHAR(8), projected_departure_datetime:datetime}

Step 3: Write the following queries using SQL and run them on your MySQL instance to collect results.

1. [10 pts] For each airport, return its IATA code and number of lounges. **You don't need to return an airport if it doesn't have any lounge.**

a) [7pts] SQL

b) [3pts] Results

For Questions 2 through 4, we use two types of flight duration (as a number in seconds):

- projected_flight_duration = projected_arrival_datetime - projected_departure_datetime
- actual_flight_duration = actual_arrival_datetime - actual_departure_datetime

The "-" operation can be implemented by using the timestampdiff() function.

2. [10pts] Among all the flights, find the flight number and actual duration of the latest flight based on projected departure datetimes.

a) [7 pts] SQL

b) [3 pts] Results

3. [10 pts] Among all the flights, return the maximum absolute difference between a flight's projected duration and its actual duration.

a) [7 pts] SQL

b) [3 pts] Results

4. [10 pts] For each flight number, return its flight number, minimum and maximal absolute difference between its projected duration and actual duration. For example, if there are two instances of a flight number 'UC9049' and their duration differences are 100 and 200 seconds, respectively, then its maximum duration difference for 'UC9049' is 200 seconds.

a) [7 pts] SQL

b) [3 pts] Results

5. [10 pts] Return the employee id and number of flights for the pilots with the maximum number of flight instances that they operated.

a) [7 pts] SQL

b) [3 pts] Results

6. [10 pts] Find the ids and average menu price of lounges whose average menu price is greater than the overall average menu price.

a) [7 pts] SQL

b) [3 pts] Results

7. [10 pts] For each dish order with at least two different dishes, return its order id, name, and quantity of each dish in the order.

a) [7 pts] SQL

b) [3 pts] Results

8. [10 pts] Return the customer ids along with their total prices for all their flight reservations. Make sure to include customers without any flight reservation. Sort the final result by customer id.

a) [7 pts] SQL

b) [3 pts] Results

9. [10 pts] For each airport, return its IATA_code and count of orders received by all its lounges. Include an airport on the list only if it has at least one lounge. Sort the results by IATA_code.

a) [7 pts] SQL

b) [3 pts] Results

10. [10pts] For each lounge, return its id, IATA_code, number of dishes, lowest dish price, highest dish price, and average dish price. The id and IATA_code of a lounge should be returned even if there are no dishes served at the lounge. Sort the results by the lounge ~~name~~**id**.

a) [7 pts] SQL

b) [3 pts] Results