CISS430: Database Systems
Test t01

Name: ncochran1@cougars.ccis.edu Score:

INSTRUCTIONS

- This is a open-book, no-discussion, no-calculator, no-browsing-on-the-web test. Open-book mean only my notes. You may use your MySQL shell.
- Cheating is a serious academic offense. If caught you will receive an immediate score of -100%.
- If any of the question cannot be answered, write ERROR. For instance if an SQL statement cannot be written to answer a query, write ERROR. If the problem cannot be solved using SQL, briefly explain why.
- Do not use recursion in this test.

All the questions involve writing SQL statements.

HONOR STATEMENT

I, [Nathan Cochran], attest to the fact that the submitted work is my own and is not the result of plagiarism. Furthermore, I have not aided another student in the act of plagiarism.

You are part of the rebel group. But lucky you – you work in the IT department. No dangerous missions. But your immediate boss is Princess Leia – and she's bossy. Well at least you don't have to worry about meeting Darth Vader and having an arm or leg sliced off.

At the rebel base, they have a RDBMS system that keeps track of the flights for transport of personnel, food supplies, etc. Here's the schema (i.e., structure of tables):

```
Flights(fno: integer,
        from: string,
        to: string,
        distance: integer,
        departs: time,
        arrives: time)

Aircraft(aid: integer,
         aname: string,
         cruisingrange: integer)

Certified(eid: integer,
          aid: integer)

Employee(eid: integer,
         ename: string,
         salary: integer)
```

For each table, the field(s) underlined form(s) the key.

The employee table contains pilot and non-pilot employees. Every pilot is certified to fly some aircraft (but possibly not all).

Write down the SQL query that answers the queries below. If you believe a query cannot be expressed in SQL, write ERROR and explain (briefly please) why it cannot be done.

Q1. Find all `eid`s of pilots certified for some "Flight Crew Shuttle" aircraft.

ANSWER

```
SELECT eid FROM Certified WHERE
aid = (SELECT aid FROM Aircraft WHERE aname = 'Flight Crew Shuttle');
```

Q2. Find the names of pilots certified for some "Flight Crew Shuttle" aircraft.

ANSWER

```
SELECT ename FROM Employee WHERE
eid = (SELECT eid FROM Certified WHERE
    aid = (SELECT aid FROM Aircraft WHERE
           aname = 'Flight Crew Shuttle')
       AND Certified.eid = Employee.eid);
```

Q3. Find the aids of all aircraft that can be used on non-stop flights from "Beggar's Canyon" to "Dune Sea Exchange".

ANSWER

```
Note: Beggar's Canyon is written as Beggars Canyon

SELECT aid FROM Aircraft WHERE
cruisingrange >= (SELECT distance FROM Flights WHERE
from_ = 'Beggars Canyon' AND to_ = 'Dune Sea Exchange');
```

Q4. Find aids of aircrafts that can be piloted by every pilot whose salary is more than $100. (You don't expect a rebel's pay to be very high.)

ANSWER

```
SELECT aid FROM Aircraft WHERE
(SELECT COUNT(Certified.aid) FROM Certified WHERE
Certified.aid = Aircraft.aid AND eid =
(SELECT eid FROM Employee WHERE Certified.eid = Employee.eid
AND salary > 100)) = (SELECT COUNT(eid) FROM Employee
WHERE salary > 100);
```

Q5. Find the names of pilots who can operate planes with a range greater than 2000 miles. but are not certified on any "Flight Crew Shuttle" aircraft.

ANSWER

```
SELECT ename FROM Employee WHERE (SELECT COUNT(eid) FROM Certified WHERE
aid = (SELECT aid FROM Aircraft WHERE Aircraft.aid = Certified.aid
AND cruisingrange > 2000) > 0 AND Certified.eid = Employee.eid)
AND
(SELECT COUNT(aid) FROM Certified WHERE
Certified.eid = Employee.eid AND
Certified.aid = (SELECT aid FROM Aircraft WHERE
aname = 'Flight Crew Shuttle')) < 1;
```

Q6. Find the `eid`s of employees who makes the highest salary.

ANSWER

```
SELECT eid FROM Employee WHERE salary = (SELECT Max(salary)
FROM Employee);
```

Q7. Find the eids of employees who makes the second highest salary.

ANSWER

```
SELECT eid FROM Employee WHERE salary = (SELECT Max(salary)
FROM Employee WHERE salary < (SELECT Max(salary) from Employee));
```

Q8. Find the eids of employees who are certified for the largest number of aircraft.

ANSWER

```
ERROR
I believe this question is impossible to answer because answering this
would require generating a temporary table of the # of certifications
each employee has, which requires a loop. Unfortunately, there is no
concept of a loop in SQL, so which would require programmatic access
to loop over Employee, generate a temporary table of maxes, and then
perform the following SQL query, where maxtable is a temporary table.

SELECT eid FROM Employee WHERE (SELECT COUNT(aid) FROM Certified
WHERE Employee.eid = Certified.eid) = (SELECT MAX(max) FROM maxtable);
```

Q9. Find the `eid`s of employees who are certified for exactly three aircrafts.

ANSWER

```
SELECT eid FROM Employee WHERE (SELECT COUNT(aid) FROM
Certified WHERE Employee.eid = Certified.eid) = 3;
```

Q10. Find the total amount paid to employees as salaries.

ANSWER

```
SELECT SUM(salary) FROM Employee;
```

Q11. Is there a sequence of flights from Raider Camp to Rebel Depot? Each flight is the sequence is required to depart from the city that is the destination of thje previous flight; the first flight must leave Raider Camp and the last flight must reach Rebel Depot, and there is no restriction on the number of intermediate flights. Your query must determine whether a sequence of flights from Raider Camp to Rebel Depot exists for any `Flights` data.

ANSWER

```
Note: This problem is likely impossible, depending on the complexity
of the map layout. However, if we assume that every location can
have one and only one connection from it, then the following
recursive query will determine if any path exists. If not, it return
an empty query. If it does, it will return the # of steps required
to get to the destination.

Note: I use to_ and from_ instead of to and from for the Flights
table schema.

WITH RECURSIVE T(f, t, step, path_found)
AS (
         SELECT ('Raider Camp'), (SELECT to_ FROM Flights WHERE
          from_ = 'Raider Camp'), 1, 'FALSE'
         UNION ALL
         SELECT t,
   (SELECT to_ FROM Flights WHERE from_ = t),
   step+1,
   (CASE WHEN ((SELECT to_ FROM Flights WHERE
   from_ = t) = 'Rebel Depot') THEN 'TRUE' ELSE 'FALSE' END)
   FROM T WHERE step < 10 AND (SELECT COUNT(to_)
   FROM Flights WHERE from_ = t) > 0
) SELECT step AS steps_to_take from T WHERE path_found = 'TRUE';
```

The tech support at the rebel camp has recently been automated with an RDBMS. This includes a support ticket subsystem. Staff with technical problems can go to the support website and create a support ticket. When John Doe first creates a support ticket, his information (i.e., email address) is stored in the `SupportTickets` table. A support ticket id is automatically generated. He also has to enter a subject and a short message. His message is stored in the `Notes` table. For a support ticket, there are several extra messages entered into the system. For instance if the support personnel replies to John Doe, that message is stored in the `Notes` table. If the support personal cannot resolve the issue immediately, he will enter notes into the system for future reference. Each note has a state tag. A note from John Doe has state 0. A note replied back to John Doe has state 9.

In the example below, support ticket with id 3 has a total of 9 notes. Princess Leia is bossy and wants to check on the productivity of the support personnel. Therefore she needs to know the total wait time for users of the support system. For instance in the case of support ticket 3, the first message arrived at time stamp 5 (state 0). A reply was sent out at time 10 (state 9). The waiting time is $10-5=5$. But the issue was not resolved – a follow up question was posted at time 24. A reply was then sent at time 30. At this point, the total wait time is $(10-5)+(30-24)=11$. However, the issue is still not resolved so that a second follow up question was entered at time 45. A few personal notes were entered while the support personnel research on the problem (state 1, 4, 7). Finally a reply was sent at time 68. At this point, the total wait time for the user is $(10-5)+(30-24)+(68-45)=34$. Note that support ticket with id 3 is handled by employee with employee id 97. (More generally, a support question can be handled by multiple employees.)

Consider the following schema:

```
SupportTickets(sid:integer,
               email:string,
               subject:string)

Employees(eid:integer,
          name:string)

Notes(sid:integer,
      eid:integer,
      state:integer,
      note:string,
      time:integer)
```

Example data:

(Notes for support ticket with id 6 and 8 not shown.)

Using MySQL, answer the next few questions.

[OPTIONAL – THIS IS ONLY FOR YOUR OWN BENEFIT]

Create the above three tables. For `string`, use `VARCHAR(200)`. Obviously you must use appropriate constraints whenever possible.

ANSWER

```
CREATE TABLE SupportTickets (
                        sid INT AUTO_INCREMENT PRIMARY KEY,
                        email VARCHAR(200),
                        subject VARCHAR(200)
);

CREATE TABLE Employees (
                        eid INT PRIMARY KEY,
                        name VARCHAR(200)
);

CREATE TABLE Notes (
                        nid INT AUTO_INCREMENT PRIMARY KEY,
                        sid INT,
                        eid INT,
                        state INT,
                        note VARCHAR(200),
                        time INT
);
```

[OPTIONAL – THIS IS ONLY FOR YOUR OWN BENEFIT]

Add some data into the tables.

ANSWER

```
INSERT INTO SupportTickets (email, subject) VALUES
('test_email@gmail.com', 'Question #1'),
('test2_email@gmail.com', 'Question #2'),
('test3_email@gmail.com', 'Question #3');

INSERT INTO Employees (eid, name) VALUES
(3, 'John Doe'),
(97, 'Blake Deere');


INSERT INTO Notes (sid, eid, state, note, time) VALUES
(
  (SELECT sid FROM SupportTickets WHERE subject = 'Question #1'),
  (SELECT eid FROM Employees WHERE name = 'John Doe'),
  0,
  'Hey I have a problem X!',
  5
),
(
  (SELECT sid FROM SupportTickets WHERE subject = 'Question #1'),
  (SELECT eid FROM Employees WHERE name = 'John Doe'),
  0,
  'Hurry Up!!!',
  6
),
(
  (SELECT sid FROM SupportTickets WHERE subject = 'Question #1'),
  (SELECT eid FROM Employees WHERE name = 'Blake Deere'),
  9,
  'Here is the solution for X!',
  10
),
(
  (SELECT sid FROM SupportTickets WHERE subject = 'Question #1'),
  (SELECT eid FROM Employees WHERE name = 'John Doe'),
  0,
  'That did not completely fix problem X!',
  24
),
(
  (SELECT sid FROM SupportTickets WHERE subject = 'Question #1'),
  (SELECT eid FROM Employees WHERE name = 'Blake Deere'),
  9,
  'Try this other thing to fix X!',
```

```
  30
),
(
  (SELECT sid FROM SupportTickets WHERE subject = 'Question #1'),
  (SELECT eid FROM Employees WHERE name = 'John Doe'),
  0,
  'That still does not fix it!',
  45
),
(
  (SELECT sid FROM SupportTickets WHERE subject = 'Question #1'),
  (SELECT eid FROM Employees WHERE name = 'Blake Deere'),
  9,
  'Our apologies, try this last final solution!',
  68
);
```

Q12. Write an SQL statement to answer this query: What is the total wait time for each support ticket? The resulting relation has the support ticket id, email of person creating the ticket, and the total wait time. You must not assume that for every sid, for each note with state 0, there's a followup note with state 9, i.e., do not assume the state 0 notes and state 9 notes pair up. It's possible to have a state 0 that is not paired up with a state 9 note.

In the case when the question-reply sequence (for a fixed support ticket id) is $Q_1R_1Q_2R_2Q_3R_3Q_4$ (where Q_i is a question and R_j is a reply), the total wait time should ignore the last question Q_4 since there's no reply yet. For the case when the sequence is $Q_1Q_2R_1$, i.e., there are two questions and one reply, you assume R_1 replies both Q_1 and Q_2 and therefore the total time is the sum of two difference between R_1, Q_1 and R_1, Q_2.

ANSWER

```
CREATE TEMPORARY TABLE T1 AS
SELECT N1.sid, N1.note AS N1_note, N1.time AS N1_time, N2.note AS
N2_note, N2.time AS N2_time, N2.time - N1.time AS wait
FROM Notes AS N1
JOIN
Notes AS N2
ON N1.sid = N2.sid
WHERE N1.state = 0
AND N2.state = 9
AND N2.time > N1.time
AND N2.time = (SELECT MIN(N3.time) FROM Notes AS
N3 WHERE N3.time > N1.time AND N3.state = 9);

SELECT * FROM T1;
SELECT SUM(wait) FROM T1;
```