Midterm Exam #2 (Version B)
CS 122A
Spring 2018

Max. Points: 100
(Please read the instructions carefully)

Instructions:
- The total time for the exam is 50 minutes; be sure to budget your time accordingly.
- The exam is closed book and closed notes but “open cheat sheet”.
- Read each question first, in its entirety, and then carefully answer each part of the question.
- If you don’t understand something, ask one of the exam proctors for clarification.
- If you still find ambiguities in a question, note the interpretation you are taking.

<table>
<thead>
<tr>
<th>QUESTION</th>
<th>TOPIC</th>
<th>POINTS</th>
<th>SCORE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>How Queryous</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>2</td>
<td>Never Ending SQL</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>3</td>
<td>Name That Query</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>4</td>
<td>And Briefly</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>5</td>
<td>True, False, or Null?</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>TOTAL</td>
<td>All</td>
<td>100</td>
<td>100</td>
</tr>
</tbody>
</table>
Question 1: How Queryous (20 points)

(20 pts) Consider again the Hoofers Piloting Club database, and suppose (if needed or helpful) that the tables contain the data shown below.

Pilot(pid, pname, rating, age)
Reserves(pid, aid, date)
Airplanes(aid, aname, color)

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>Abby</td>
<td>3</td>
<td>null</td>
</tr>
<tr>
<td>5</td>
<td>Joe</td>
<td>null</td>
<td>35</td>
</tr>
<tr>
<td>1</td>
<td>Bob</td>
<td>5</td>
<td>35</td>
</tr>
<tr>
<td>3</td>
<td>Zack</td>
<td>10</td>
<td>19</td>
</tr>
<tr>
<td>2</td>
<td>Sally</td>
<td>9</td>
<td>22</td>
</tr>
</tbody>
</table>

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>2017-03-15</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>2017-04-15</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>2017-04-15</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>2018-01-01</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>2017-12-25</td>
<td></td>
</tr>
</tbody>
</table>

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Air Force</td>
<td>red</td>
</tr>
<tr>
<td>2</td>
<td>Boeing</td>
<td>yellow</td>
</tr>
<tr>
<td>3</td>
<td>Hawker</td>
<td>green</td>
</tr>
<tr>
<td>5</td>
<td>Falcon</td>
<td>yellow</td>
</tr>
<tr>
<td>6</td>
<td>Eagle</td>
<td>red</td>
</tr>
</tbody>
</table>

Write the following queries in the indicated language:

(10 pts) Relational Algebra:
Print the aids of airplanes that are reserved by a pilot with a rating of at least 5 as well as a pilot named Abby.

(10 pts) SQL:
For each pilot, print their pid, name, rating, and – if they have any red airplanes reserved – the reservation details (date, aid, and aname).

SCORE: _________
Question 2: Never Ending SQL (20 points)

And again! Just when you thought it was safe. For this problem, you are to assume that the Reserves table has been created with pid and aid each being defined as FOREIGN KEYs with the ON DELETE CASCADE option for their respective tables (Pilots and Airplanes). Note that a new table has just been added to the mix – PlanePrefs – which has no primary or foreign keys. PlanePrefs is initially empty and will soon be used by the club to assist them in advertising airplanes and other services to their club members.

Pilots(pid, pname, rating, age) Reserves(pid, aid, date) Airplanes(aid, aname, color) PlanePrefs(pid, date, aname, color)

<table>
<thead>
<tr>
<th>Pilots</th>
<th>Reserves</th>
<th>Airplanes</th>
</tr>
</thead>
<tbody>
<tr>
<td>pid</td>
<td>aid</td>
<td>date</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>2017-03-15</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>2017-04-15</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>2017-04-15</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>2018-01-01</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2017-12-25</td>
</tr>
</tbody>
</table>

In addition, assume that the following additional SQL goodies have already been defined:

```
DELIMITER $$
CREATE PROCEDURE DropPlane(oldplane INT(11))
BEGIN
  DELETE FROM Airplanes WHERE aid = oldplane;
END $$
```

```
CREATE TRIGGER TrackActivity
AFTER INSERT ON Reserves FOR EACH ROW
BEGIN
  INSERT INTO PlanePrefs(pid, date, aname, color)
  VALUES( NEW.pid, NEW.date,
    (SELECT aname
     FROM Airplanes WHERE aid = NEW.aid),
    (SELECT color
     FROM Airplanes WHERE aid = NEW.aid));
END; $$
DELIMITER ;
```

(a) (10 pts) Show below the result of running the following statement against the initial four tables above. (I.e., show the contents of the tables after execution. You may skip any tables that are unaffected.)

```
CALL DropPlane(1);
```

SCORE: _________
**Question 2: SQL Never Ending SQL (continued)**

Here’s all of the database information once again. You are to assume that Reserves table was created with pid and aid each being defined as FOREIGN KEYs with the ON DELETE CASCADE option for their respective tables (Pilots and Airplanes). Note the new table – PlanePrefs – which has no primary or foreign keys. This new table, which they just added, is currently **empty** and will be used by the club to assist them in advertising airplanes and other services to their club members.

- **Pilots** (pid, pname, rating, age)
- **Reserves** (pid, aid, date)
- **Airplanes** (aid, aname, color)
- **PlanePrefs** (pid, date, aname, color)

In addition, assume that the following additional SQL goodies have already been defined:

```sql
DELIMITER $$
CREATE TRIGGER TrackActivity
AFTER INSERT ON Reserves FOR EACH ROW
BEGIN
DELETE FROM Airplanes WHERE aid = NEW.aid;
 INSERT INTO PlanePrefs(pid, date, aname, color)
 VALUES( NEW.pid, NEW.date,
 (SELECT aname FROM Airplanes WHERE aid = NEW.aid),
 (SELECT color FROM Airplanes WHERE aid = NEW.aid));
END; $$
DELIMITER ;
```

(b) (5 pts) Show below the result of running the following statement starting from the **initial four** tables above. (i.e., show the contents of the tables after execution. You may skip any tables that are unaffected.)

```
INSERT INTO Reserves(pid, aid, date) VALUES (3,2,'2018-05-30'), (5,4,'2018-05-31');
```

(c) (5 pts) Show below the results of executing the following statement against the initial tables above.

```
CREATE VIEW AgeSummary(age, numpilots) AS
SELECT age, COUNT(*) as numpilots FROM Pilots GROUP BY age;
SELECT * FROM AgeSummary A WHERE A.age >= 21;
```
Question 3: Name That Query (30 points)
(30 pts) Consider the Hoofers Piloting Club database:

Pilots(pid, pname, rating, age)  Reserves(pid, aid, rdate)  Airplanes(aid, aname, color)

What follows is a list of queries in various languages. For the first five queries, Q1-Q5, list all of the equivalent queries from the remainder of the list (Q6-Q16). Note that there may be several, or perhaps none, and that two queries are equivalent if their results are identical under all database states.

Q1 (6 pts) – Select the ids and names of pilots to whom alcohol can’t be legally served at a UCI party;(Proof of age 21 or over required)

*Equivalent queries:*

Q2 (6 pts) – SELECT MAX(age) FROM Pilots;

*Equivalent queries:*

Q3 (6 pts) – SELECT aid FROM Airplanes WHERE color = 'red' AND color = 'purple';

*Equivalent queries:*

Q4 (6 pts) – \{ t(age) | \exists p \in Pilots (t.age = p.age \land (p.rating < 3)) \}

*Equivalent queries:*

Q5 (6 pts) – \pi_{aid,aname,color}(\pi_{age=22}Pilots) \bowtie (Reserves \bowtie Airplanes)

*Equivalent queries:*

Q6 – \pi_{age}(\sigma_{rating < 3} Pilots)
Q7 – SELECT * FROM Airplanes A WHERE A.aid IN (SELECT R.aid FROM Reserves R WHERE R.pid IN (SELECT P.pid FROM Pilots P WHERE P.age = 22));
Q8 – SELECT P.age FROM Pilots P ORDER BY P.age ASC LIMIT 1;
Q9 – SELECT pid, pname FROM Pilots WHERE age < 21;
Q10 – SELECT R.aid FROM Reserves R WHERE FALSE;
Q12 – \{ t(age) | \exists p \in Pilots (t.age = p.age \land p2 \in Pilots (p2.age > p1.age)) \}
Q13 – SELECT pid, pname FROM Pilots WHERE age < 21 OR age IS NULL;
Q15 – SELECT age FROM Pilots WHERE rating < 3;
Q16 – \{ t(age) | \exists p \in Pilots (t.age = p.age \land (p.rating >= 3)) \}

SCORE: _________
Question 4: Short Answers (10 points)

(a) (1 pt each) Given two relations $T$ and $S$, where $T$ has $N_T$ tuples, $S$ has $N_S$ tuples, and $N_T > N_S > 0$, what are the minimum and maximum possible result cardinalities for the following relational algebra queries expressed in terms of $N_T$ and $N_S$?

\[
\pi_A T: \quad \text{min size:} \quad \text{max size:}
\]

\[
T \bowtie S: \quad \text{min size:} \quad \text{max size:}
\]

\[
S - T: \quad \text{min size:} \quad \text{max size:}
\]

(b) (4 points) Suppose that the table $\text{Items}(\text{i_id, iname, price, shipping, tax})$ contains three tuples:

- $(352, 'Whatzit', 70.00, 4.00, \text{NULL})$
- $(76, 'Gadget', 45.00, 2.00, 1.20)$
- $(457, 'Widget', \text{NULL}, \text{NULL}, 1.50)$

For each of the following SQL queries, show the output that it would produce if run against $\text{Products}$:

(i) (2 pts) SELECT SUM(price + shipping) AS taxfreecost FROM Items;

(ii) (2 pts) SELECT i_id, (price * tax) + shipping AS totalcost FROM Items;

Question 5: True, False, or Null? (20 points)

SCORE: _________
(2 pts each) For each of the following statements, indicate whether the statement is true (circle TRUE) or false (circle FALSE):

- Most real SQL implementations disallow CHECK constraints that involve correlated subqueries.
  
  TRUE  ❌ FALSE

- Enforcing the uniqueness of a table’s column (e.g., User.email) would be a good use of a trigger.
  
  TRUE  ❌ FALSE

- Consider the query “SELECT * FROM Orders o INNER JOIN Lineitems l ON (o.oid = l.oid)”. The result tuple count for this query could be larger if “INNER” were “LEFT OUTER” instead.
  
  TRUE  ❌ FALSE

- Two tables (or queries) are considered by SQL to be UNION-compatible if they have the same number of columns and the same column names in the same order.
  
  TRUE  ❌ FALSE

- The query “(SELECT * FROM Boat B WHERE B.color = ‘yellow’) UNION ALL (SELECT * FROM Boat B WHERE B.color != ‘yellow’)” is a way to (inefficiently ☹️) return all Boat tuples.
  
  TRUE  ❌ FALSE

- The SQL set operation EXCEPT is not actually necessary because it is always possible to express an equivalent query by appropriately using a subquery instead.
  
  TRUE  ❌ FALSE

- It’s possible to write a relational algebra query involving small Sailor, Boat, and Reservation relations whose result tuple count is infinite.
  
  TRUE  ❌ FALSE

- The use of triggers and stored procedures is advisable when writing database applications that need to be easily run on different relational DBMS choices (e.g., MySQL, Oracle, DB2, …).
  
  TRUE  ❌ FALSE

- If “CREATE TABLE Shirts (name VARCHAR(40), size ENUM('small', 'medium', 'large'))” is the DDL for the table Shirts, “SELECT size FROM Shirts” can have at most 3 result rows.
  
  TRUE  ❌ FALSE

- It’s possible to write a tuple relational calculus query involving Sailor, Boat, and Reservation relations containing less than 20 tuples each whose result tuple count is infinite.
  
  TRUE  ❌ FALSE

SCORE: _________