Midterm Exam #2 (Version B)
CS 122A
Spring 2019
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.
- The last page of the exam contains a set of tables that will be used throughout the exam; feel free to detach that page for easier/faster access if you want.

<table>
<thead>
<tr>
<th>QUESTION</th>
<th>TOPIC</th>
<th>POINTS</th>
<th>SCORE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>I’m Queryous</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Try To SQLch That</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Never Ending SQL</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Query This</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>TOTAL</td>
<td>All</td>
<td>100</td>
<td></td>
</tr>
</tbody>
</table>
Question 1: I’m Queryous (30 points)

(30 pts) Consider the AnteaterMusic.com database on the last page of the exam. What follows is a list of queries against it 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 remember that two queries are only equivalent if their results are identical under all possible legal database states.

Q1 (6 pts) – SELECT (i1.price + i2.price) as totalp FROM Inst i1, Inst i2 WHERE i1.sku = 202 AND i2.sku = 203;

   Equivalent queries: ________________________

Q2 (6 pts) – π sku (σ color = 'sunburst' Inst) - π sku (σ cust = 'Mike' Order)

   Equivalent queries: ________________________

Q3 (6 pts) – { t(model) | ∃n ∈ Inst (t.model = n.model ∧ ∃s ∈ Inst (s.model = n.model ∧ s.color != n.color)) }

   Equivalent queries: ________________________

Q4(6 pts) – π mname, model ((σ city = 'Memphis' (Manuf) ⨝ Inst))

   Equivalent queries: ________________________

Q5 (6 pts) – SELECT sku FROM Inst WHERE price < (SELECT MAX(price) FROM Inst);

   Equivalent queries: ________________________

Q6 – SELECT model FROM Inst GROUP BY model HAVING COUNT(color) = 1;

Q7 – SELECT i.sku FROM Inst iWHERE i.color = 'sunburst'
   AND i.sku NOT IN (SELECT o.sku FROM Order o WHERE o.cust = 'Mike');

Q8 – SELECT SUM(price) as totalp FROM Inst WHERE sku = 202 OR sku = 203;

Q9 – SELECT (SELECT i1.price FROM Inst i1 WHERE i1.sku = 202) +
     (SELECT i2.price FROM Inst i2 WHERE i2.sku = 203);

Q10– SELECT model FROM Inst ORDER BY color ASC LIMIT 1;

Q11 – SELECT m.mname, i.model FROM Manuf m LEFT OUTER JOIN Inst I ON m.mid = i.mid
     WHERE m.city = 'Memphis';

Q12 – SELECT i.model FROM Inst iWHERE 0 =
     (SELECT COUNT(*) FROM Inst c WHERE c.model = i.model AND c.color != i.color);

Q13 – { t(sku) | ∃i ∈ Inst (t.model = i.model ∧ ∃s ∈ Inst (s.price > i.price)) }

Q14 – SELECT sku FROM Inst WHERE price < ANY (SELECT price FROM Inst);

Q15 – SELECT i.sku FROM Inst i WHERE i.color = 'sunburst'
     AND NOT EXISTS (SELECT o.sku FROM Order o WHERE o.cust = 'Mike');

Q16 – { t(mname, model) | ∃i ∈ Inst (t.model = i.model ∧
     ∃m ∈ Manuf (t.mname = m.mname ∧ i.mid = m.mid ∧ m.city = 'Memphis')) }

SCORE: _________
Question 2: Try To SQLch That (30 points)

(3 pts each) For each of the following statements, circle the appropriate answer (or answers!).

- The tuple relational calculus is important to understand because it is very similar to the way that relational database systems communicate SQL query plans to their users.
  
  \[
  \begin{array}{ll}
    \text{FALSE} & \text{TRUE} \\
  \end{array}
  \]

- For any non-empty state of a relational table \( \text{lnst} \) (\( \text{sku} \), \( \text{mid} \), \( \text{model} \), \( \text{color} \), \( \text{price} \)), the relational algebra query \( \pi_{\text{model, color}}(\text{lnst}) \) will return more tuples than the query \( \pi_{\text{model}}(\text{lnst}) \).
  
  \[
  \begin{array}{ll}
    \text{FALSE} & \text{TRUE} \\
  \end{array}
  \]

- An \( \text{AnteaterMusic.com} \) customer wants to know the \( \text{mid} \) and \( \text{model} \) of the instruments (if any) in \( \text{lnst} \) that come in all of the known colors. Which of the following relational algebra operator(s) would be useful in answering their query?
  
  \[
  \sigma \quad \pi \quad \div \quad \cup 
  \]

- For a MySQL table \( \text{Stock} \) (\( \text{sku} \), \( \text{mid} \), \( \text{qty} \)), which among the following mechanisms should be used to make sure that all newly inserted or updated rows have \( \text{qty} \geq 0 \)?
  
  \[
  \text{TRIGGER} \quad \text{VIEW} \quad \text{FOREIGN KEY constraint} \quad \text{CHECK constraint} 
  \]

- Which of the following standard SQL set operations does MySQL’s SQL support?
  
  \[
  \text{UNION} \quad \text{EXCEPT} \quad \text{INTERSECT} 
  \]

- The tuple relational calculus has the same expressive power as the relational algebra.
  
  \[
  \begin{array}{ll}
    \text{FALSE} & \text{TRUE} \\
  \end{array}
  \]

- System DBMS1 has LEFT OUTER JOIN but not RIGHT OUTER JOIN, while DBMS2 has both. This means that DBMS1 has less expressive power than DBMS2 (i.e., that there are English queries that only S2 could answer.)
  
  \[
  \begin{array}{ll}
    \text{FALSE} & \text{TRUE} \\
  \end{array}
  \]

- In a SQL GROUP BY query, tuples with null values for the grouping attribute are ignored.
  
  \[
  \begin{array}{ll}
    \text{FALSE} & \text{TRUE} \\
  \end{array}
  \]

- Views can be a useful mechanism when attempting to ensure that end users of a database can only see data that they should be permitted to see based on their role in an enterprise.
  
  \[
  \begin{array}{ll}
    \text{FALSE} & \text{TRUE} \\
  \end{array}
  \]

- Tom, Dick, and Harry work for \( \text{AnteaterMusic.com} \). Harry is the DBA and he GRANTs the SELECT and UPDATE privileges on table \( \text{lnst} \) to Dick, both with GRANT OPTION, but only GRANTs the SELECT privilege to Tom. If Tom really wants to, there is still a way that he could gain UPDATE access to \( \text{lnst} \) without Harry GRANTing him that permission.
  
  \[
  \begin{array}{ll}
    \text{FALSE} & \text{TRUE} \\
  \end{array}
  \]
Question 3: Never Ending SQL (20 points)

Consider the AnteaterMusic.com database on the last page of the exam. One of the employees in the warehouse has noticed some shortcomings of the current way that the company’s inventory and orders are being managed. Have a look at the data related to sku 202 and you will see what one of her concerns is: The store has somehow managed to take more ES-175 orders than it has instruments in stock! (Oops.) It’s time to explore what could be done about this and to make sure there aren’t other issues with the way the database has been set up so far.

Assume that the following trigger has been added to the MySQL database:

```
DELIMITER $$

CREATE TRIGGER checkSupply
AFTER INSERT ON Order
FOR EACH ROW
BEGIN
    DECLARE num INT;
    SET num = (SELECT qty FROM Stock WHERE sku = NEW.sku);
    IF num = 0 THEN
        SIGNAL SQLSTATE '02000'
        SET MESSAGE_TEXT =
        'Warning: This item will need to be placed on back order.';
    END IF;
END;
$$
```

(a) (8 pts) Show and very briefly explain below the result of running the following statement starting from the initial four tables shown on the last page of the exam. (I.e., show the contents of the tables after execution, and explain briefly what transpired. Skip redrawing tables that are not affected at all, but do show the complete content of any/all affected tables – or at least show enough so that we cannot possibly misunderstand your answer.)

```
INSERT INTO Order VALUES (5678, 'Mimi', 401);
```
Question 3: SQL Never Ending SQL (continued)

Now assume that the following stored procedure has been added to the MySQL database as well:

```
DELIMITER $$
CREATE PROCEDURE TakeOrder (cname VARCHAR(30), isku INT)
BEGIN
    DECLARE nextid INT;
    SET nextid = (SELECT MAX(oid)+1 FROM Order);
    INSERT INTO Order (oid, cust, sku)
    VALUES (nextid, cname, isku);
    UPDATE Stock SET qty = qty - 1
    WHERE sku = isku;
END;
$$
```

(b) (8 pts) Show below the result of running the following statement against the initial four tables shown on the last page of the exam. (I.e., show the contents of the tables after execution. Skip redrawing tables that are not affected at all, but do show the complete content of any/all affected tables – or at least show enough so that we cannot possibly misunderstand your answer.)

```
CALL TakeOrder('Jerry', 103);
```

(c) (4 pts) Ponder (in your head ☺) what part (a)’s SQL insert statement’s result would have been if the sku being ordered had been 101 instead of 401 – and then briefly list the two most important things that requiring applications and users to use the TakeOrder procedure, instead of allowing them to perform such direct inserts into Order, would ensure.

1: 

2: 

SCORE: ________
Question 4: Query This (20 points)

(20 pts) Consider once again the AnteaterMusic.com database on the last page of the exam. Answer the following questions about queries in the indicated language:

(10 pts) SQL:
Show what the result of the following query would be given the current state of the database.

```
SELECT i.model, m.mname, COUNT(DISTINCT i.color) AS colors, AVG(i.price) AS avprice, SUM(s.qty) AS stock
FROM Inst i JOIN Manuf m ON i.mid = m.mid LEFT JOIN Stock s ON i.sku = s.sku
WHERE m.city != 'San Luis Obispo'
GROUP BY i.model, m.mname;
```

(10 pts) Relational Algebra:
Print the names of customers who have ordered an instrument for which there is only one instance in stock (i.e., its quantity is 1) at the moment as well as a sunburst-colored instrument costing over $1100.

Note: These may (or may not) refer to the same instrument, and you can assume that customer names uniquely identify customers (i.e., two different customers will not have the same name).
Appendix: Schemas and Data

The following tables have been set up and populated by the MySQL DBA for AnteaterMusic.com for use in supporting both their in-store and web-based musical instrument sales. (The application code for both types of sales will be running against this shared MySQL database.)

<table>
<thead>
<tr>
<th>Manuf</th>
<th>mid</th>
<th>mname</th>
<th>city</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Fender</td>
<td>Corona</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Gibson</td>
<td>Memphis</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>MusicMan</td>
<td>San Luis Obispo</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Ibanez</td>
<td>Nagoya</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Inst</th>
<th>sku</th>
<th>mid</th>
<th>model</th>
<th>color</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>1</td>
<td>1</td>
<td>Strat</td>
<td>sunburst</td>
<td>600</td>
</tr>
<tr>
<td>102</td>
<td>1</td>
<td>2</td>
<td>Tele</td>
<td>blonde</td>
<td>500</td>
</tr>
<tr>
<td>103</td>
<td>1</td>
<td>3</td>
<td>Strat</td>
<td>black</td>
<td>600</td>
</tr>
<tr>
<td>201</td>
<td>2</td>
<td>4</td>
<td>SG</td>
<td>cherry</td>
<td>1400</td>
</tr>
<tr>
<td>202</td>
<td>2</td>
<td>5</td>
<td>ES-175</td>
<td>sunburst</td>
<td>2500</td>
</tr>
<tr>
<td>203</td>
<td>2</td>
<td>6</td>
<td>ES-335</td>
<td>walnut</td>
<td>null</td>
</tr>
<tr>
<td>301</td>
<td>3</td>
<td>7</td>
<td>Sterling</td>
<td>sunburst</td>
<td>1500</td>
</tr>
<tr>
<td>401</td>
<td>4</td>
<td>8</td>
<td>PM-100</td>
<td>natural</td>
<td>2000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Stock</th>
<th>sku</th>
<th>mid</th>
<th>qty</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>1</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>102</td>
<td>1</td>
<td>13</td>
<td></td>
</tr>
<tr>
<td>103</td>
<td>1</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>201</td>
<td>2</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>202</td>
<td>2</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>301</td>
<td>3</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>401</td>
<td>4</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Order</th>
<th>oid</th>
<th>cust</th>
<th>sku</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>Kenny</td>
<td>202</td>
<td></td>
</tr>
<tr>
<td>1001</td>
<td>Mike</td>
<td>202</td>
<td></td>
</tr>
<tr>
<td>1002</td>
<td>Joyce</td>
<td>102</td>
<td></td>
</tr>
<tr>
<td>1003</td>
<td>Jeff</td>
<td>101</td>
<td></td>
</tr>
<tr>
<td>1004</td>
<td>Ritchie</td>
<td>101</td>
<td></td>
</tr>
<tr>
<td>1005</td>
<td>Pat</td>
<td>401</td>
<td></td>
</tr>
<tr>
<td>1006</td>
<td>Mike</td>
<td>301</td>
<td></td>
</tr>
</tbody>
</table>

-- Manufacturer info (the store’s available brands)
CREATE TABLE Manuf (  
  mid INT,
  mname VARCHAR(40) NOT NULL,
  city VARCHAR(50) NOT NULL,
  PRIMARY KEY (mid)
);

-- Instrument info (products handled by the store)
CREATE TABLE Inst (  
  sku INT,
  mid INT NOT NULL,
  model VARCHAR(40) NOT NULL,
  color VARCHAR(20),
  price DECIMAL(8,2),
  PRIMARY KEY (sku),
  FOREIGN KEY (mid) REFERENCES Manuf (mid)
  ON DELETE NO ACTION
);

-- Order info (who has which instruments on order)
CREATE TABLE Order (  
  oid INT,
  cust VARCHAR(40) NOT NULL,
  sku INT NOT NULL,
  PRIMARY KEY (oid),
  FOREIGN KEY (sku) REFERENCES Inst (sku)
  ON DELETE NO ACTION
);

-- Stock info (the store’s current inventory)
CREATE TABLE Stock (  
  sku INT,
  mid INT NOT NULL,
  qty INT NOT NULL,
  PRIMARY KEY (sku),
  FOREIGN KEY (sku) REFERENCES Inst (sku)
  ON DELETE CASCADE,
  FOREIGN KEY (mid) REFERENCES Manuf (mid)
  ON DELETE CASCADE
);