Question 1.  [6 marks]

Part (a)  [2 marks]

Suppose that we have a relation called One with attributes a, b, c, and d, and that the set of attributes \( \{b, c\} \) is a key for One. Is the set of attributes \( \{b, c, d\} \) a key for One? Circle one.

- Yes
- No
- It depends

Explain:

Even if it satisfies the requirement that no two tuples may agree on the set of attributes, it is not the minimal.

Part (b)  [4 marks]

Consider this schema:

- Brillig(tove, gyre, gimble)
- Slithy(mimsy, rath)
- TumTum(vorpal, jubjub)

Suppose relation TumTum has 100 tuples. How many tuples could Brillig have? Circle all answers that do not violate the schema.

- 0
- 1
- 2
- 61
- 100
- 106

Suppose relation Brillig has 100 tuples. How many tuples could Slithy have? Circle all answers that do not violate the schema.

- 0
- 1
- 2
- 61
- 100
- 106
Question 2.  [8 marks]

Recall this schema from Assignment 1. (Attributes card and expiry have been removed from relation User.)

Relations

User(username, lastName, firstName, email)
A tuple in this relation represents a user.

Venue(vID, name, city, address, owner)
A venue, such as Massey Hall.

Seat(sID, name, venue, accessible)
A seat for which a ticket can be sold.

Event(eID, name, type, when, venue)
An event for which tickets can be sold, such as a concert, or play.

Ticket(eID, sID, price)
A single ticket for an event.

Purchase(eID, sID, username, when)
A ticket was purchased for a seat at an event.

Integrity constraints

Seat[venue] ⊆ Venue[vID]
Event[venue] ⊆ Venue[vID]
Ticket[eID] ⊆ Event[eID]
Ticket[sID] ⊆ Seat[sID]
Purchase[eID, sID] ⊆ Ticket[eID, sID]
Purchase[username] ⊆ User[username]

\[ \Pi_{\text{type}=\text{Event}} \subseteq \{ \text{“concert”, “play”, “musical”, “talk”} \} \]

\[ \sigma_{\text{Ticket}.sID=\text{Seat}.sID \land \text{Event}.venue \neq \text{Seat}.venue}
\]
\[ (\exists \text{Event} \bowtie \text{Ticket}) \times \text{Seat} = \emptyset \]

Write a query in relational algebra to find, for every user, the highest ticket price that user has ever paid.
Report their username, ticket price, event ID, and venue name. A user will appear more than once if there is a tie for their top ticket price. Users who have bought no tickets are not included.

Use only the basic operators \( \Pi, \sigma, \bowtie, \times, \cap, \cup, -, \rho, \) and assignment.

Solution:

\[ \text{Paid} \ (\text{user, price, eID}) := \Pi_{\text{username}, \text{price}} \ (\text{Ticket} \bowtie \text{Purchase}) \]

\[ \text{Not Max} \ (\text{user, price}) :=
\]
\[ \Pi_{\text{price}, \text{user}} \ 
\sigma_{\text{price} \leq \text{price} \bowtie \text{user}} \ 
\sigma_{\text{price} \bowt \text{price} \bowt \text{price}} \ 
\]

\[ \text{Max} \ (\text{user, price}) :=
\]
\[ \Pi_{\text{price}, \text{user}} \ (\text{Paid}) - \text{Not Max} \]

\[ \text{Answer} \ (\text{user, price, eID, venue}) :=
\]
\[ \Pi_{\text{user}, \text{price}, \text{eID}, \text{name}, \text{type}, \text{when}, \text{venue}} \ (\text{Max} \bowtie \text{Paid} \bowtie \text{Event}) \times \text{Venue} \]

\[ \sigma_{\text{venue} = \text{vID}} \]

Student #:  
Page 2 of 7  
CONT'D...
Question 3. [6 marks]

Consider the following definitions, and the integrity constraint that they lead up to.

\[
\begin{align*}
One(eID, sID) & := \Pi_{\text{Event}.eID, sID, \sigma_{\text{name} = \text{'AdelE'}}}(\text{Event} \bowtie \text{Ticket}) \\
Two(\text{username}, eID, sID) & := (\Pi_{\text{username}, \text{User}}) \times One \\
Three(\text{username}, eID, sID) & := \Pi_{\text{username}, eID, sID}(\text{Purchase}) \\
Four(\text{username}, eID, sID) & := Two - Three \\
(\Pi_{\text{username}, eID}(Two)) - (\Pi_{\text{username}, eID}(Four)) & = \emptyset
\end{align*}
\]

Part (a) [4 marks]

Below, make an instance of the database that violates this constraint. Draw only the relevant relations: Event, Ticket and Purchase. Draw each as a table with exactly 5 rows. You can leave columns empty if their values are irrelevant.

<table>
<thead>
<tr>
<th>Event</th>
<th>Ticket</th>
</tr>
</thead>
<tbody>
<tr>
<td>eID</td>
<td>name</td>
</tr>
<tr>
<td>1</td>
<td>AdelE</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Purchase</th>
<th>Ticket</th>
</tr>
</thead>
<tbody>
<tr>
<td>eID</td>
<td>sid</td>
</tr>
<tr>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>1</td>
<td>B</td>
</tr>
<tr>
<td>1</td>
<td>C</td>
</tr>
<tr>
<td>2</td>
<td>A</td>
</tr>
<tr>
<td>2</td>
<td>B</td>
</tr>
</tbody>
</table>

Part (b) [2 marks]

Describe this integrity constraint in plain English.

No one user can purchase tickets for every seat at an Adele event.
Question 4. [6 marks]

Recall the schema from Question 1.

Relations

User(username, lastName, firstName, email)
A tuple in this relation represents a user.

Venue(vID, name, city, address, owner)
A venue, such as Massey Hall.

Seat(sID, name, venue, accessible)
A seat for which a ticket can be sold.

Event(eID, name, type, when, venue)
An event for which tickets can be sold, such as a concert, or play.

Ticket(eID, sID, price)
A single ticket for an event.

Purchase(eID, sID, username, when)
A ticket was purchased for a seat at an event.

Integrity constraints

Seat[vID] ⊆ Venue[vID]
Event[vID] ⊆ Venue[vID]
Ticket[eID] ⊆ Event[eID]
Ticket[sID] ⊆ Seat[sID]
Purchase[eID, sID] ⊆ Ticket[eID, sID]
Purchase[username] ⊆ User[username]
Π_{type} Event ⊆ {“concert”, “play”, “musical”, “talk”}

\[ \sigma_{\text{Ticket.sID} = \text{Seat.sID} \land \text{Event.venue} = \text{Seat.venue}} ( (\text{Event} \bowtie \text{Ticket}) \times \text{Seat}) = \emptyset \]

Write a query in SQL to find users who have bought a ticket to an event at the venue named “Massey Hall”. Report the username, and the number of times that the user has bought a ticket to an event at the venue named “Massey Hall”. Include only users who have spent over $250 in total on tickets to Massey Hall events.

Solution:

```sql
SELECT username, count(*)
FROM Venue v
JOIN Event e ON v.vID = e.venue
JOIN Purchase p ON e.eID = p.eID
JOIN Ticket t ON t.eID = p.eID and t.sID = p.sID
WHERE v.name = 'Massey Hall'
GROUP BY username
HAVING sum(price) > 250;
```
Question 5.  [4 marks]

Suppose we have the following tables:

One:

\[
\begin{array}{c|c|c}
\hline
x & y & z \\
\hline
4 & 7 & 8 \\
4 & 11 & 5 \\
4 & 11 & 2 \\
6 & 11 & 5 \\
0 & 3 & 6 \\
1 & 3 & 6 \\
1 & 3 & 9 \\
\hline
\end{array}
\]

(8 rows)

Two:

\[
\begin{array}{c|c}
\hline
x & y \\
\hline
1 & 3 \\
4 & 11 \\
4 & 11 \\
\hline
\end{array}
\]

(4 rows)

Three:

\[
\begin{array}{c|c}
\hline
x & y \\
\hline
hello & its \\
me & I \\
wondering & hello \\
you & hear \\
the & world \\
\hline
\end{array}
\]

(6 rows)

Four:

\[
\begin{array}{c|c}
\hline
x & z \\
\hline
hello & from \\
the & other \\
think & from \\
outside & tell \\
you & you \\
\hline
\end{array}
\]

(5 rows)

Show the output of each of the following queries:

Solutions

csc343h-dianeh=> (SELECT x, y FROM One)
csc343h-dianeh=> INTERSECT ALL
csc343h-dianeh=> (SELECT * FROM Two);

\[
\begin{array}{c|c}
\hline
x & y \\
\hline
1 & 3 \\
4 & 11 \\
4 & 11 \\
\hline
\end{array}
\]

(3 rows)

csc343h-dianeh=> (SELECT x FROM One)
csc343h-dianeh=> EXCEPT
csc343h-dianeh=> (SELECT x FROM Two);

\[
\begin{array}{c}
x \\
\hline
0 \\
6 \\
\hline
\end{array}
\]

(2 rows)

csc343h-dianeh=> SELECT count(*)
csc343h-dianeh=> FROM Three NATURAL JOIN Four;

count

\[
\begin{array}{c}
\hline
6 \\
\hline
\end{array}
\]

(1 row)

csc343h-dianeh=>
csc343h-dianeh=> SELECT count(*)
csc343h-dianeh-> FROM Three NATURAL FULL OUTER JOIN Four;
count
-----
10
(1 row)