**Question 1.**  [11 marks]

Recall this schema, which we have used many times in class. Here we are adding one more relation called *Program*. It records the subject POSIs that students are enrolled in. (“POSt” is short for “program of study”, by the way.)

**Relations**

- Student(sID, surName, firstName, campus, email, cgpa)
- Course(dept, cNum, name, breadth)
- Offering(oID, dept, cNum, term, instructor)
- Took(sID, oID, grade)
- Program(sID, POSt)

**Integrity constraints**

- Offering[dept, cNum] ⊆ Course[dept, cNum]
- Took[sID] ⊆ Student[sID]
- Took[oID] ⊆ Offering[oID]
- Program[sID] ⊆ Student[sID]

**Part (a)  [7 marks]**

Write a query to find the sIDs of students on campus ‘StG’ who have exactly one subject POSI. Use only the basic operators Π, σ, ⊥, ×, ∩, ∪, −, ρ, and assignment.

**Solution:**

\[
\begin{align*}
StG(sID) & := (\Pi_{sID} \sigma_{\text{campus}=\text{StG}} \text{Student}) \\
OnePOSt(sID) & := \Pi_{P1.sID} \sigma_{P1.sID=P2.sID \land P1.POSt \neq P2.POSt} (\rho_{P1 \text{Program}} \times \rho_{P2 \text{Program}}) \\
Answer(sID) & := StG \cap OnePOSt
\end{align*}
\]
Part (b) [4 marks]

Consider the following query:

\[
Croom(instructor, cNum) := (\Pi_{instructor} Offering) \times (\Pi_{cNum} \sigma_{dept='CSC'\land cNum \geq 200\land cNum < 300} Course)
\]

\[
Flep(instructor, cNum) := Croom - (\Pi_{instructor,cNum} Offering)
\]

\[
Answer(instructor) := [(\Pi_{instructor} Offering) - (\Pi_{instructor} Flep)] - (\Pi_{instructor,\sigma_{dept='MAT'} Offering})
\]

1. Below are instances of the relations that are relevant to this query. Add exactly 5 rows to Offering so that professors Able and Bland will not appear in the result of the query, but professors Cranky and Devlish will. Use only these instructor names, and be sure the instance you create is valid.

<table>
<thead>
<tr>
<th>Course</th>
<th>dept</th>
<th>cNum</th>
<th>name</th>
<th>breadth</th>
<th>Offering</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSC</td>
<td>108</td>
<td>Intro Prog</td>
<td>false</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CSC</td>
<td>207</td>
<td>Intro Design</td>
<td>true</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CSC</td>
<td>209</td>
<td>Sys Prog</td>
<td>false</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MAT</td>
<td>137</td>
<td>Calculus</td>
<td>false</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CSC</td>
<td>369</td>
<td>Op Sys</td>
<td>false</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ENG</td>
<td>244</td>
<td>Shakespeare</td>
<td>true</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CSC</td>
<td>108</td>
<td></td>
<td></td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>CSC</td>
<td>108</td>
<td></td>
<td></td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>CSC</td>
<td>209</td>
<td></td>
<td></td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>CSC</td>
<td>209</td>
<td></td>
<td></td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>CSC</td>
<td>207</td>
<td></td>
<td></td>
<td>9</td>
<td></td>
</tr>
</tbody>
</table>

Solution:

- Professor Able is already not in the result, so there is nothing to do.
- Professor Bland is currently in the result, but we can get rid of him by having him teach some math course.
- Professor Cranky is not in the result, and we need to have him teach csc207 and csc209 (all the 200-level CSC courses), and no math courses, in order to get him in the result.
- Professor Devlish needs CSC207, and no math courses, in order to be in the result.
- To add a fifth new tuple and not mess up the results, we can add some course that is not a math course or a 2nd-year CSC course.

Here is a complete solution:
Offering:

<table>
<thead>
<tr>
<th>oID</th>
<th>dept</th>
<th>cNum</th>
<th>term</th>
<th>instructor</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>CSC</td>
<td>108</td>
<td>t1</td>
<td>Cranky</td>
</tr>
<tr>
<td>1</td>
<td>CSC</td>
<td>108</td>
<td>t2</td>
<td>Able</td>
</tr>
<tr>
<td>7</td>
<td>CSC</td>
<td>209</td>
<td>t1</td>
<td>Devlish</td>
</tr>
<tr>
<td>2</td>
<td>CSC</td>
<td>209</td>
<td>t1</td>
<td>Bland</td>
</tr>
<tr>
<td>9</td>
<td>CSC</td>
<td>207</td>
<td>t3</td>
<td>Bland</td>
</tr>
<tr>
<td>10</td>
<td>MAT</td>
<td>137</td>
<td>t4</td>
<td>Bland</td>
</tr>
<tr>
<td>11</td>
<td>CSC</td>
<td>207</td>
<td>t4</td>
<td>Cranky</td>
</tr>
<tr>
<td>12</td>
<td>CSC</td>
<td>209</td>
<td>t4</td>
<td>Cranky</td>
</tr>
<tr>
<td>13</td>
<td>CSC</td>
<td>108</td>
<td>t4</td>
<td>Able</td>
</tr>
</tbody>
</table>

2. What does this query compute? Do not describe the steps it takes, only what is in the result, and make your answer general to any instance of the schema.

**Solution:**
All instructors who have taught every 2nd-year CSC course, but no MAT course.

**Question 2. [6 marks]**

**Part (a) [2 marks]**
At UofT, a student may have no POSSt, one POSSt, or several POSSts. In the previous question, we introduced a new relation called *Program* to record information about students’ POSSts. Instead of making a separate *Program* relation, we could add a column for *POSSt* in the *Student* relation. Would that be a good design? Circle one answer:

**Solution:**

**Yes**  **No**

**Explain:**

With this design, we wouldn’t be able to record two POSSts for one student. We also could not handle a student who has no POSSt, since we don’t have null values in relational algebra.

**Part (b) [4 marks]**

Consider this schema:

\[
\begin{align*}
R(\text{one}, \text{two}, \text{three}) & \\
S(\text{four}, \text{five}, \text{six}) & \\
T(\text{seven}, \text{eight}) & \\
\end{align*}
\]

\[S[\text{five}, \text{six}] \subseteq R[\text{one}, \text{two}]\]

\[S[\text{four}] \subseteq T[\text{eight}]\]

Suppose relation \( S \) has 100 tuples. How many tuples could \( R \) have? Circle all answers that do not violate the schema.

**Solution:**

\[0\quad 1\quad 82\quad 100\quad 101\]
Suppose relation $S$ has 100 tuples. How many tuples could $T$ have? Circle all answers that do not violate the schema.

**Solution:**

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>82</th>
<th>100</th>
<th>101</th>
</tr>
</thead>
</table>

Question 3. [5 marks]

The question refers to the schema from Question 1. Write a query in SQL to find the total number of courses in the ‘CSC’ department each student has taked. Report the student id and the total number of distinct ‘CSC’ courses taken.

Solution:

```
SELECT sid, count(distinct cNum)
FROM Took, Offering
WHERE dept = 'CSC' AND Took.oID = Offering.oID
GROUP BY sid
```
Question 4.  [8 marks]

Part (a)  [3 marks]
Consider the same schema from the Question 1. Suppose we wrote the query

```sql
SELECT ________________________
FROM Offering, Took
WHERE Offering.oID = Took.oID
GROUP BY Offering.oID;
```

Which of the following could go in the SELECT clause? Circle all that apply.

- sID
- count(sID)
- Offering.oID
- grade
- avg(grade)
- count(instructor)
- oID

Solution:

<table>
<thead>
<tr>
<th>sID</th>
<th>count(sID)</th>
<th>Offering.oID</th>
<th>grade</th>
<th>avg(grade)</th>
<th>count(instructor)</th>
<th>oID</th>
</tr>
</thead>
</table>

Here are the error messages for the problematic ones:

```
csc343h-dianeh=> select sID
csc343h-dianeh-> from offering, took
csc343h-dianeh-> where Offering.oID = Took.oID
csc343h-dianeh-> GROUP BY Offering.oID;
ERROR: column "took.sid" must appear in the GROUP BY clause or be used in an aggregate function
LINE 1: select sID
```

```
csc343h-dianeh=> select grade
csc343h-dianeh-> from offering, took
csc343h-dianeh-> where Offering.oID = Took.oID
csc343h-dianeh-> GROUP BY Offering.oID;
ERROR: column "took.grade" must appear in the GROUP BY clause or be used in an aggregate function
LINE 1: select grade
```

```
csc343h-dianeh=> select oid
sc343h-dianeh-> from offering, took
csc343h-dianeh-> where Offering.oID = Took.oID
csc343h-dianeh-> GROUP BY Offering.oID;
ERROR: column reference "oid" is ambiguous
LINE 1: select oid
```

Part (b)  [3 marks]
We discussed in lecture how SQL subquery operators could possibly be implemented using other SQL operations. Suppose we have two tables R(a,b) and S(b,c).

Consider the following two queries:
-- Query 1
SELECT a AS answer
FROM R
WHERE EXISTS
(SELECT * FROM S
    WHERE c > a AND R.b = S.b);

-- Query 2
SELECT R.a AS answer
FROM R, S
WHERE c > a AND R.b = S.b;

On the next page, give a database instance where these two queries produce different results, and the results of the two queries.

Solution:

insert into R values
(1, 2),
(1, 2),
(3, 4),
(5, 6);

insert into S values
(2, 101),
(4, 103),
(4, 203);

-- Query 1 gives:
answer
-------
  1
  1
  3
(3 rows)

-- Query 2 gives:
answer
-------
  1
  1
  3
  3
(4 rows)
Part (c)  [2 marks]

Here is an attempt to fix Query 2, to make it identical to Query 1. Explain, in English, why this new query is not the same as Query 1. Don’t just give a database instance – write a sentence or two describing the problem precisely.

-- Query 3
SELECT DISTINCT R.a AS answer
FROM R, S
WHERE c > a AND R.b = S.b;

Solution:

If R has duplicate a values they won’t appear in this query, even though they both would in Query 1, if the conditions are satisfied for both tuples.

-- Query 3 gives:
answer
--------
  1
  3
(2 rows)