Question 1. [7 marks]

Recall this schema, which we have used many times in class. Here we are adding one more relation called Club. It records the university clubs that students have joined (if any). For example, if a student joins the ‘Chess’ club, a tuple would appear in table Club indicating that this student is part of the Chess club.

**Relations**

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

**Integrity constraints**

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

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

Write a query to find the sIDs of students who have a cgpa greater than 3 and who are part of exactly one club. Use only the basic operators Π, σ, ◯, ×, ∩, ∪, −, ρ, and assignment.

**Solution:**

- HighGPA(sID) := Π_sIDσ_{cgpa>3}Student
- TwoOrMore(sID) := Π_sIDσ_{C_1.sID=C_2.sID\land C_1.clubName\neq C_2.clubName}(ρ_{C_1Club \times C_2Club})
- OneClub(sID) := (Π_sIDClub) − TwoOrMore
- Answer(sID) := HighGPA ∩ OneClub
Part (b)  [3 marks]

Consider the following query:

$$\text{Apple}(sID, oID) := \Pi_{sID} \text{Took} \times \Pi_{oID}(\sigma_{cNum>300 \land cNum<400 \land dept='CSC' \ Offering})$$

$$\text{Orange}(sID, oID) := \text{Apple} - \Pi_{sID,oID} \text{Took}$$

$$\text{Answer}(sID) := \Pi_{sID} \text{Took} - \Pi_{sID} \text{Orange}$$

You are given below, instances of the relations that are relevant to this query. Add or remove the minimum number of rows to/from these relations so that student s3 and s1 do appear in the results and student s2 does not.

<table>
<thead>
<tr>
<th>Took:</th>
<th>Offering:</th>
</tr>
</thead>
<tbody>
<tr>
<td>sID</td>
<td>oID</td>
</tr>
<tr>
<td>s1</td>
<td>o1</td>
</tr>
<tr>
<td>s1</td>
<td>o2</td>
</tr>
<tr>
<td>s1</td>
<td>o4</td>
</tr>
<tr>
<td>s2</td>
<td>o1</td>
</tr>
<tr>
<td>s2</td>
<td>o2</td>
</tr>
<tr>
<td>s2</td>
<td>o4</td>
</tr>
<tr>
<td>s3</td>
<td>o1</td>
</tr>
<tr>
<td>s3</td>
<td>o3</td>
</tr>
<tr>
<td>s2</td>
<td>o3</td>
</tr>
<tr>
<td>oID</td>
<td>dept</td>
</tr>
<tr>
<td>o1</td>
<td>CSC</td>
</tr>
<tr>
<td>o2</td>
<td>MATH</td>
</tr>
<tr>
<td>o3</td>
<td>CSC</td>
</tr>
<tr>
<td>o4</td>
<td>CSC</td>
</tr>
</tbody>
</table>

Solution: s1 is already in the result. Add (s3,o4,xx), xx is any mark, to Took so that s3 appears in the result. Remove one of CSC3xx courses of s2 from Took relation so that s2 does not appear in the result.
Question 2.  [4 marks]

Part (a)  [2 marks]
In the schema of question 1, relation \( \text{Club}(sID, \text{clubName}) \) records the university clubs that students have joined (if any). Assuming that the key of this relation is \( \text{clubName} \) (instead of \((sID,\text{clubName})\)), how many members can a club have? Circle one answer:

only one  as many as the number of students  as many as possible

Solution: only one

Part (b)  [2 marks]
Consider this schema:

\[
\text{One}(a, b, c) \\
\text{Two}(d, e, f) \\
\text{Two}[d,f] \subseteq \text{One}[a,b]
\]

Assuming that relation One has 100 tuples, how many tuples could relation Two have? Circle all possible answers.

1 100 180 50

Solution: 1, 50, 100
Question 3.  [6 marks]

Recall this schema, which we have used many times in class. Here we are adding a few more relations:

- **Residence**: all the student residences from all 3 campuses.
- **LivesInRes**: which students live in these residences.
- **AppliedToRes**: the current new applications to live in residence, either from students who do not yet have a spot in a residence, or from those who wish to switch residences. For example, if a student applies for the ‘New College’ residence, then a new application entry will be recorded in ‘AppliedInRes’, mapping the student’s sID to the rID for the residence with rname = ‘New College’.

**Relations**

- Student(sID, surName, firstName, campus, email, cgpa)
- Course(dept, cNum, name, breadth)
- Offering(oID, dept, cNum, term, instructor)
- Took(sID, oID, grade)
- Residence(rID, rname, address, campus)
- LivesInRes(sID, rID)
- AppliedToRes(aID, rID, sID)

**Integrity constraints**

- Offering[dept, cNum] ⊆ Course[dept, cNum]
- Took[sID] ⊆ Student[sID]
- Took[oID] ⊆ Offering[oID]
- LivesInRes[sID] ⊆ Student[sID]
- LivesInRes[rID] ⊆ Residence[rID]
- AppliedToRes[sID] ⊆ Student[sID]
- AppliedToRes[rID] ⊆ Residence[rID]

Please note that:

- It is possible that some students do not live in any residence at all, in any campus (e.g., if they rent their own place off-campus, or live with parents).
- In the Student relation, the ‘campus’ field represents the campus where the student is registered. A student can be registered in one campus, but live in a residence located on a different campus (see the ‘campus’ field from the Residence relation).

**What do you need to do?** Write a query to find the sIDs of students who have a cgpa greater than anyone else who lives in the same residence.

**Solution:**

SELECT S.sID FROM Student S, LivingInRes L
WHERE L.sID = S.sID AND S.cgpa >
(SELECT max(S1.cgpa) FROM Student S1, LivingInRes L1
WHERE S1.sID = L1.sID
AND L1.rID = L.rID);

Or anything else they may come up with correctly, possibly using GROUP BY and HAVING, etc.
Question 4. [15 marks]

For this question, we will use the same schema from Question 3.

Part (a) [3 marks]

Is the Select query below valid? Indicate on the query below if there are any invalid items in the Select list or in the Having clause.

Circle all the invalid ones, if any.

SELECT count(*), avg(cgpa), firstName, rID, sID, email
FROM Student S, Took T, LivesInRes L
WHERE S.sID = L.sID AND S.sID = T.sID
GROUP BY rID
HAVING avg(cgpa) > 75 AND count(*) <> 0
ORDER BY count(*);

Solution:

Only aggregates or rID are valid.

Part (b) [4 marks]

Create a view called OffCampus(sID), to find all those students who do not live in any residence, in any of the campuses. The view only has one attribute: sID.

Solution:

CREATE VIEW OffCampus(sID) AS
SELECT sID FROM Student
WHERE sID NOT IN
(SELECT sID FROM LiveInRes);
Part (c)  [4 marks]

Using the view in a), create a view called LiveCloser(sID, campus) that retrieves those off-campus students who applied to get a spot only in a residence that is located on the same campus as the one they are registered in. The attributes in the View are sID and the campus that they are registered in (or the one they applied for, which is the same).

Solution:

CREATE VIEW LiveCloser(sID, campus) AS
SELECT S.sID, S.campus from OffCampus O, Student S, AppliedToRes A
WHERE O.sID = A.sID AND S.sID = O.sID
AND NOT EXISTS
( SELECT sID, campus FROM AppliedToRes A1, Residence R1
WHERE A1.rID = R1.rID
AND A1.sID = A.sID
AND S.campus <> R.campus);

Part (d)  [4 marks]

Find all sIDs for off-campus students, that either prefer to continue living off-campus and did not apply for any residence, or who have applied for some residences that are not in the same campus where they are currently registered. Hint: You may use the views in b) and/or c), if you wish.

Solution:

(SELECT * FROM OffCampus) - (SELECT sID FROM LiveCloser);

^ Instead of “-“ this should be  EXCEPT