Warmup #1

Consider the following set of functional dependencies, $F$, for the schema $R(A, B, C, D, E)$

\[
AB \rightarrow C \\
C \rightarrow D \\
D \rightarrow EA \\
E \rightarrow B
\]

1. Find the candidate keys for the schema $R$.
   - $AB, C, D, \text{and} EA$

2. Compute the closure, $F^+$, for the set $F$.
   - **Hint**: you can use $\ast$ to refer to any subset of attributes in $R$, and $\alpha$ to indicate the full set \{A, B, C, D, E\}. So $A \ast \rightarrow \alpha$ means that $A$, along with any and all subsets of \{B, C, D, E\}, functionally determines all attributes.
   - $EB \rightarrow E, EB \rightarrow B, A \rightarrow A, B \rightarrow B, E \rightarrow EB, E \rightarrow B, E \rightarrow E, E \rightarrow EB$, and all the dependencies of the form: $AB \ast \rightarrow \alpha, C \ast \rightarrow \alpha, D \ast \rightarrow \alpha, EA \ast \rightarrow \alpha$

Warmup #2

Using the functional dependency set from the first warmup, compute the canonical cover $F_c$.

- $D \rightarrow EA, AB \rightarrow C, E \rightarrow B, C \rightarrow D$

Warmup #3

Consider the following schema:

- student($ID, name, departmentName, totalCredits$)
- professor($ID, name, departmentName, salary$)
- classroom($building, roomNumber, capacity$)
- course($courseID, semester, year, building, roomNumber, professorID$)
- takes($studentID, courseID, grade$)

Note that:

- the $building, roomNumber$ attributes of $course$ relation are references to the $classroom$ relation
- $professorID$ of $course$ is a reference to $professor$
- $studentID$ of $takes$ is a reference to $student$
courseID of takes is a reference to course

1. Considering the given database schema, write SQL queries to construct this database, where the attributes name, departmentName, building are strings of various lengths (maximum length of 50), semester is only one character, and all remaining attributes are integers.

```
CREATE TABLE student ( ID int PRIMARY KEY, name varchar(50), departmentName
                         varchar(50), totalCredits int)
CREATE TABLE professor ( ID int PRIMARY KEY, name varchar(50), departmentName
                        varchar(50), salary int)
CREATE TABLE classroom ( building varchar(50), roomNumber int, capacity int,
                        PRIMARY KEY(building, roomNumber))
CREATE TABLE course ( courseID int PRIMARY KEY, semester char(1), departmentName varchar(50),
                     year int, building varchar(50), roomNumber int, professorID int,
                     FOREIGN KEY (building, roomNumber) REFERENCES classroom,
                     FOREIGN KEY professorID REFERENCES professor)
CREATE TABLE takes ( studentID int, courseID int, grade int, PRIMARY KEY(studentID,
                      courseID) FOREIGN KEY studentID REFERENCES student,
                      FOREIGN KEY courseID REFERENCES course)
```

2. Write a SQL query to find the name of every professor in the CS department. SELECT name FROM professor WHERE departmentName = "S"

3. If the salary attribute of professor represents monthly salary, write a SQL query to find every possible pairing of student ID and professor ID, where professor IDs included in the result consist of the IDs of the professors whose yearly salary is greater than $10,000 and student IDs included in the result consist of IDs of students who have completed fewer than 10 credits. SELECT S.ID, P.ID FROM student AS S, professor AS P WHERE P.salary * 12 > 10000 AND S.totalCredits < 10

**Warmup #4**

Write a SQL query for each of the following, using the schema from warmup 3:

1. If the salary attribute of professor represents monthly salary, find the names and yearly salaries of professors who hold classes in rooms that do not exist in classroom. SELECT name, salary * 12 FROM (professor JOIN course ON professor.ID = course.professorID) LEFT JOIN classroom ON course.building = classroom.building AND course.roomNumber = classroom.roomNumber WHERE building IS NULL AND roomNumber IS NULL

2. Find the name, total credits, and GPA of every student whose GPA is greater than 3.5. GPA can be calculated by summing a student’s grades and dividing by total credits. SELECT name, totalCredits, SUM( grade ) / totalCredits AS GPA FROM student JOIN takes ON student.ID = takes.studentID GROUP BY name, totalCredits HAVING GPA > 3.5

3. Write a SQL query for finding the ID and name of the professor(s) who earns the highest salary, without using aggregate functions. SELECT ID, name FROM professor WHERE ID NOT IN (SELECT P.ID FROM professor AS P, professor AS PP WHERE P.salary < PP.salary)

**Problem 5 (To Be Graded)**

Using the following schema, write SQL queries to answer the following questions.

```sql
advisor(s_ID, j_ID)
classroom(building, room_number, capacity)
course(course_id, title, dept_name, credits)
```
1. Find the students who have had classes in the highest number of distinct buildings.

   with stud_class as
      (select ID, count(distinct building) as building_count from student
       natural join takes natural join section as t
       group by ID)

   select id, from stud_class where building_count =
      (select max(building_count) from stud_class)

2. Find the instructor who has given the highest number of A’s.

   with takes_student as (select takes.id as stud_id, course_id,
                          sec_id, semester, year, grade from takes inner join student on stud_id = student.id)

   with count_table as
      (select id, count(stud_id) as grade_count
       from (takes_student natural join teaches) natural join instructor
       where grade = 'A'
       group by instructor.id)

   select id from count_table where grade_count = (select max(grade_count) from count_table)

3. Display the names of all students along with their advisor’s name. Note that not every student has an advisor.

   with advise as
      (select s_id as id, i_id from advisor)

   select student.name, instructor.name
   from student left outer join (advise inner join instructor on advise.i_id = instructor.id)

4. The university needs to determine whether a student is eligible for graduation. Here are the following requirements for graduation:
   
   • the student has taken at least 130 credits
   • the student has taken courses in at least five different departments
   • the student has taken at least seven courses in a single department. This requirement is not restricted to the department associated with their student ID.

   Find all students who are eligible to graduate.
with credits as
    (select id from student where total_cred > 130)

with depth as
    (select id, count(distinct course_id) as dept_depth from
    (course inner join takes on course.course_id = takes.course_id)
    inner join student on student.id = takes.id
    group by id, depart_name)

with breadth as
    (select id, count(distinct dept_name) as dept_breadth from
    course inner join takes on course.course_id = takes.course_id
    inner join student on student.id = takes.id
    group by id)

select id
    from credits inner join depth on credits.id = depth.id
    inner join breadth on breadth.id = depth.id
    where dept_depth > 6 and dept_breadth > 4

Problem 6 (To Be Graded)

For each set of relations and functional dependencies, answer the following questions:

1. What are all the nontrivial functional dependencies that follow for the given functional dependencies? You should restrict yourself to FD’s with single attributes on the right side.

2. What are the candidate keys of the relation? List all possible candidate keys.

3. Give a canonical cover for the given relation and functional dependencies.

\[ R(A, B, C, D) \]

\[ AB \rightarrow C \]

\[ C \rightarrow D \]

\[ D \rightarrow A \]

\[ C \rightarrow A, AB \rightarrow D, AC \rightarrow D, BC \rightarrow A, BC \rightarrow D, BD \rightarrow A, BD \rightarrow C, CD \rightarrow A, ABC \rightarrow D, ABD \rightarrow C, BCD \rightarrow A \]

AB, BC, and BD The given set are a canonical cover

\[ S(A, B, C, D) \]

\[ A \rightarrow B \]

\[ B \rightarrow C \]

\[ B \rightarrow D \]

\[ A \rightarrow C, A \rightarrow D, AC \rightarrow D, AD \rightarrow C AC \rightarrow B, AD \rightarrow B, BC \rightarrow D, BD \rightarrow C ABC \rightarrow D, ABD \rightarrow C, ACD \rightarrow B \]

A

\[ A \rightarrow B, B \rightarrow CD \]

\[ T(A, B, C, D) \]
\(AB \rightarrow C\)
\(BC \rightarrow D\)
\(CD \rightarrow A\)
\(AD \rightarrow B\)
\(ADC \rightarrow B\)

\(AB \rightarrow D, AD \rightarrow C, BC \rightarrow A, CD \rightarrow B\)
\(ABC \rightarrow D, ABD \rightarrow C, BCD \rightarrow A\),
\(AB, AD, BC, CD\)
\(AB \rightarrow C, BC \rightarrow D, CD \rightarrow A, AD \rightarrow B\)