CIS5516 HW2

Zhijia Chen

# Basic SQL queries:

1. Find the names of all the instructors from Biology department.

**SQL query:**

SELECT DISTINCT name FROM instructor WHERE dept\_name = 'Biology';

**Relational algebra query:**

**Tuple calculus query:**

**Domain calculus query:**

CIS5516=# SELECT DISTINCT name FROM instructor WHERE dept\_name = 'Biology';

 name

-------

 Crick

(1 row)

1. Find the names of courses in Computer Science department which have 3 credits.

**SQL query:**

SELECT title FROM course WHERE dept\_name = 'Comp. Sci.' AND credits = 3;

**Relational algebra query:**

**Tuple calculus query:**

**Domain calculus query:**

CIS5516=# SELECT title FROM course WHERE dept\_name = 'Comp. Sci.' AND credits = 3;

          title

--------------------------

 Robotics

 Image Processing

 Database System Concepts

(3 rows)

1. For the student with ID 12345 (or any other value), show all course\_id and title of all courses registered for by the student.

**SQL query:**

SELECT course.course\_id, title FROM (takes INNER JOIN course ON takes.course\_id = course.course\_id) WHERE id = '00128';

**Relational algebra query:**

**Tuple calculus query:**

**Domain calculus query:**

CIS5516=# SELECT course.course\_id, title FROM (takes INNER JOIN course ON takes.course\_id = course.course\_id) WHERE id = '00128';

 course\_id |           title

-----------+----------------------------

 CS-101    | Intro. to Computer Science

 CS-347    | Database System Concepts

(2 rows)

1. For the student with ID 12345 (or any other value), show the total number of credits taken by that student. Use SQL aggregation on courses taken by the student.

**SQL query:**

SELECT sum(credits) FROM (takes INNER JOIN course ON takes.course\_id = course.course\_id) WHERE id = '00128';

**Relational algebra query:** Not applicable.

**Tuple calculus query:** Not applicable.

**Domain calculus query:** Not applicable.

CIS5516=# SELECT sum(credits) FROM (takes INNER JOIN course ON takes.course\_id = course.course\_id) WHERE id = '00128';

 sum

-----

   7

(1 row)

1. Display the total credits for each student, along with the ID of the student; don't worry about the name of the student. (Don't bother about students who have not registered for any course, they can be omitted).

**SQL query:**

SELECT id, sum(credits) FROM (takes INNER JOIN course ON takes.course\_id = course.course\_id) GROUP BY id;

**Relational algebra query:** Not applicable.

**Tuple calculus query:** Not applicable.

**Domain calculus query:** Not applicable.

CIS5516=# SELECT id, sum(credits) FROM (takes INNER JOIN course ON takes.course\_id = course.course\_id) GROUP BY id;

  id   | sum

-------+-----

 76543 |   7

 19991 |   3

 00128 |   7

 98765 |   7

 54321 |   8

 12345 |  14

 45678 |  11

 98988 |   8

 55739 |   3

 76653 |   3

 44553 |   4

 23121 |   3

(12 rows)

1. Find the names of all students who have taken any Comp. Sci. course ever (there should be no duplicate names).

**SQL query:**

SELECT DISTINCT name FROM (student INNER JOIN takes ON student.id = takes.id INNER JOIN course ON takes.course\_id = course.course\_id) WHERE course.dept\_name = 'Comp. Sci.';

**Relational algebra query:**

**Tuple calculus query:**

**Domain calculus query:**

CIS5516=# SELECT DISTINCT name FROM (student INNER JOIN takes ON student.id = takes.id INNER JOIN course ON takes.course\_id = course.course\_id) WHERE course.dept\_name = 'Comp. Sci.';

   name

----------

 Bourikas

 Brown

 Levy

 Shankar

 Williams

 Zhang

(6 rows)

1. Display the IDs of all instructors who have never taught a course. (Note: Oracle uses the keyword minus in place of except).

**SQL query:**

SELECT id FROM instructor EXCEPT (SELECT id FROM teaches);

**Relational algebra query:**

**Tuple calculus query:**

**Domain calculus query:**

CIS5516=# SELECT id FROM instructor EXCEPT (SELECT id FROM teaches);

  id

-------

 33456

 58583

 76543

(3 rows)

# Intermediate SQL queries:

1. Find the maximum and minimum enrollment across all sections, considering only sections that had some enrollment. (Don't worry about those that had no students taking that section.).

**SQL query:**

WITH tmp AS (SELECT count(id) AS enrol\_num FROM (takes INNER JOIN section ON takes.course\_id = section.course\_id AND takes.sec\_id = section.sec\_id AND takes.year = section.year) GROUP BY takes.course\_id, takes.sec\_id, takes.year, takes.semester HAVING count(id) > 0) SELECT max(enrol\_num) AS max\_enrol, min(enrol\_num) AS min\_enrol FROM tmp;

**Relational algebra query:** Not applicable.

**Tuple calculus query:** Not applicable.

**Domain calculus query:** Not applicable.

CIS5516=# WITH tmp AS (SELECT count(id) AS enrol\_num FROM (takes INNER JOIN section ON takes.course\_id = section.course\_id AND takes.sec\_id = section.sec\_id AND takes.year = section.year) GROUP BY takes.course\_id, takes.sec\_id, takes.year, takes.semester HAVING count(id) > 0) SELECT max(enrol\_num) AS max\_enrol, min(enrol\_num) AS min\_enrol FROM tmp;

 max\_enrol | min\_enrol

-----------+-----------

         6 |         1

(1 row)

1. Find all sections that had the maximum enrollment (along with the enrollment), using a subquery.

**SQL query:**

WITH tmp AS (SELECT count(id) AS enrol\_num, takes.year, takes.semester, takes.course\_id, takes.sec\_id FROM (takes INNER JOIN section ON takes.course\_id = section.course\_id AND takes.sec\_id = section.sec\_id AND takes.year = section.year) GROUP BY takes.course\_id, takes.sec\_id, takes.year, takes.semester HAVING count(id) > 0) SELECT year, semester, course\_id, sec\_id, enrol\_num FROM tmp WHERE enrol\_num = (SELECT max(enrol\_num) FROM tmp);

**Relational algebra query:** Not applicable.

**Tuple calculus query:** Not applicable.

**Domain calculus query:** Not applicable.

CIS5516=# WITH tmp AS (SELECT count(id) AS enrol\_num, takes.year, takes.semester, takes.course\_id, takes.sec\_id FROM (takes INNER JOIN section ON takes.course\_id = section.course\_id AND takes.sec\_id = section.sec\_id AND takes.year = section.year) GROUP BY takes.course\_id, takes.sec\_id, takes.year, takes.semester HAVING count(id) > 0) SELECT year, semester, course\_id, sec\_id, enrol\_num FROM tmp WHERE enrol\_num = (SELECT max(enrol\_num) FROM tmp);

 year | semester | course\_id | sec\_id | enrol\_num

------+----------+-----------+--------+-----------

 2009 | Fall     | CS-101    | 1      |         6

(1 row)

1. Find all courses whose identifier starts with the string "CS-1".

**SQL query:**

SELECT title FROM course WHERE course\_id LIKE 'CS-1%';

**Relational algebra query:**

**Tuple calculus query:**

**Domain calculus query:**

CIS5516=# select title from course where course\_id like 'CS-1%';

           title

----------------------------

 Intro. to Computer Science

 Game Design

(2 rows)

# Advanced SQL queries:

1. Create a view faculty showing only the ID, name, and department of instructors.

**SQL query:**

CREATE VIEW faculty AS SELECT id, name, dept\_name FROM instructor;

**Relational algebra query:**

**Tuple calculus query:** Not applicable.

**Domain calculus query:** Not applicable.

CIS5516=# CREATE VIEW faculty AS SELECT id, name, dept\_name FROM instructor;

CREATE VIEW

CIS5516=# select \* from faculty limit 1;

  id   |    name    | dept\_name

-------+------------+------------

 10101 | Srinivasan | Comp. Sci.

(1 row)

1. Create a view CSinstructors, showing all information about instructors from the Comp. Sci. department.

**SQL query:**

CREATE VIEW CSinstructor AS SELECT \* FROM instructor WHERE dept\_name = 'Comp. Sci.';

**Relational algebra query:**

**Tuple calculus query:** Not applicable.

**Domain calculus query:** Not applicable.

CIS5516=# CREATE VIEW CSinstructor AS SELECT \* FROM instructor WHERE dept\_name = 'Comp. Sci.';

CREATE VIEW

CIS5516=# SELECT \* FROM CSinstructor LIMIT 1;

  id   |    name    | dept\_name  |  salary

-------+------------+------------+----------

 10101 | Srinivasan | Comp. Sci. | 65000.00

(1 row)

1. Find all rooms that have been assigned to more than one section at the same time. Display the rooms along with the assigned sections. (Hint: use WITH or views.)

**SQL query:**

SELECT course\_id, sec\_id, room\_number, building FROM section WHERE (building, room\_number, time\_slot\_id) IN (SELECT building, room\_number, time\_slot\_id FROM section GROUP BY building, room\_number, time\_slot\_id HAVING count(course\_id) > 1);

**Relational algebra query:** Not applicable.

**Tuple calculus query:** Not applicable.

**Domain calculus query:** Not applicable.

CIS5516=# select course\_id, sec\_id, room\_number, building from section where (building, room\_number, time\_slot\_id) in (select building, room\_number, time\_slot\_id from section group by building, room\_number, time\_slot\_id having count(course\_id) > 1);

 course\_id | sec\_id | room\_number | building

-----------+--------+-------------+----------

 CS-190    | 2      | 3128        | Taylor

 CS-319    | 2      | 3128        | Taylor

 CS-347    | 1      | 3128        | Taylor

 EE-181    | 1      | 3128        | Taylor

(4 rows)