COMP23111
2016-2017
EX03
Alex-Radu Malan
9770386

```
-------------------------------------------------------------------------
--                                PART 1                                --
-------------------------------------------------------------------------

-- Setting the pagesize and the line size in order to get a good-looking
-- table

SET linesize 250
SET pagesize 250

start create-University-tables.sql;
start populate-University-tables.sql;

-- Viewing the student table

SELECT * FROM student;

-- Find the names of all students who have taken at least one computer
science course, making sure there are no duplicate names in the result

SELECT DISTINCT name FROM student
JOIN takes  ON student.ID = takes.ID
JOIN course ON takes.course_id = course.course_id
WHERE course.dept_name = 'Comp. Sci.';

-- Find the IDs and names of all students who have not taken any course
-- offering before Spring 2009

SELECT DISTINCT student.name, student.ID, section.year
FROM student
JOIN takes   ON student.ID = takes.ID
JOIN section ON section.year >= 2009;

-- For each department, find the maximum salary of instructors in that
-- department. You may assume that every department has at least one
-- instructor.

SELECT dept_name, max(salary)
FROM instructor
GROUP BY dept_name;

-- Find the lowest, across all departments, of the per-department maximum
-- salary computed by the preceding query.

SELECT min(minimum)
FROM (SELECT dept_name, max(salary) AS minimum
FROM instructor
GROUP BY dept_name);
```

```
-- Create a new CS-001 course in computer science, titled Weekly Seminar,
-- with 10 credits.

INSERT INTO course
VALUES ('CS-001', 'Weekly Seminar', 'Comp. Sci.', '10');

-- Create a new CS-002 course in computer science, titled Monthly
Seminar, with 0 credits.

-- INSERT INTO course VALUES ('CS-002', 'Monthly Seminar', 'Comp. Sci.',
'0');

-- In order to put a limit to a value placed in a column there is a
-- constraint the constraints of credits is bigger than 0 so it gives an
-- error if we use 0

-- Create a section of the CS-001 course in Fall 2009, with section id of
-- 1

INSERT INTO section(course_id, sec_id, semester, year)
  VALUES('CS-001', '1', 'Fall', '2009');

-- Enrol every student in the Computer Science department in the section
-- you created in the previous statement.

INSERT INTO takes(ID, course_id, sec_id, semester, year)
  VALUES('00128', 'CS-001', '1', 'Fall', '2009');

INSERT INTO takes(ID, course_id, sec_id, semester, year)
  VALUES('12345', 'CS-001', '1', 'Fall', '2009');

INSERT INTO takes(ID, course_id, sec_id, semester, year)
  VALUES('19991', 'CS-001', '1', 'Fall', '2009');

INSERT INTO takes(ID, course_id, sec_id, semester, year)
  VALUES('23121', 'CS-001', '1', 'Fall', '2009');

INSERT INTO takes(ID, course_id, sec_id, semester, year)
  VALUES('44553', 'CS-001', '1', 'Fall', '2009');

INSERT INTO takes(ID, course_id, sec_id, semester, year)
  VALUES('45678', 'CS-001', '1', 'Fall', '2009');

INSERT INTO takes(ID, course_id, sec_id, semester, year)
  VALUES('54321', 'CS-001', '1', 'Fall', '2009');
```

```
INSERT INTO takes(ID, course_id, sec_id, semester, year)
  VALUES('55739', 'CS-001', '1', 'Fall', '2009');

INSERT INTO takes(ID, course_id, sec_id, semester, year)
  VALUES('70557', 'CS-001', '1', 'Fall', '2009');

INSERT INTO takes(ID, course_id, sec_id, semester, year)
  VALUES('76543', 'CS-001', '1', 'Fall', '2009');

INSERT INTO takes(ID, course_id, sec_id, semester, year)
  VALUES('76653', 'CS-001', '1', 'Fall', '2009');

INSERT INTO takes(ID, course_id, sec_id, semester, year)
  VALUES('98765', 'CS-001', '1', 'Fall', '2009');

INSERT INTO takes(ID, course_id, sec_id, semester, year)
  VALUES('98988', 'CS-001', '1', 'Fall', '2009');

-- Delete all enrolments in the above section where the student's name is
Zhang

DELETE FROM takes
  WHERE ID = (SELECT ID FROM student
              WHERE takes.ID = student.ID AND student.name = 'Zhang');

-- Delete all takes tuples corresponding to any section of any course
-- with the word database as a part of the title (you should make sure
-- that your code is case-insensitive when matching the word with the
-- title)

DELETE FROM takes AND course
WHERE takes.course_id = course.course_id, course.title LIKE 'Databases';

-- Delete the course CS-001.
-- If you delete a tuple then everything related to it will be deleted,
-- so it is like a waterfall

DELETE FROM course
WHERE course_id = 'CS-001';

-- Viewing the course table

SELECT * FROM course;

start drop-University-tables.sql;
```

```
--------------------------------------------------------------------------
--                                PART 2                                --
--------------------------------------------------------------------------

start create-Accident-tables.sql;
start populate-Accident-tables.sql;

-- Find the number of accidents in which the cars belonging to Jane
-- Rowling were involved.

SELECT COUNT(report_number) FROM accident
JOIN participated on accident.report_number = participated.report_number
JOIN car on participated.license = CAR.license
JOIN owns on car.license = owns.license
JOIN person on owns.driver_id = person.driver_id
where person.name = 'Jane Rowling';


-- Update the amount of damage for the car with license number KUY 629 in
-- the #accident with report number 7897423 to 2500.

SELECT * FROM participated;

UPDATE participated
  SET participated.damage_amount = 2500
  WHERE participated.license = 'KUY 629'
  AND participated.report_number = '7897423';

SELECT * FROM participated;

-- List the name of the persons that participated in accidents along with
-- the total damage caused (from the largest to the smallest) but only
-- include those whose total damage is above 3000.

SELECT name, damage_amount FROM person, participated
WHERE person.driver_id = participated.driver_id
AND participated.damage_amount > 3000
ORDER BY participated.damage_amount DESC;

-- Create a view that returns the locations where accidents have occurred
-- along with the average amount of damage in that location. Call this
-- view average_damage_per_location
```

```
CREATE VIEW average_damage_per_location AS
(SELECT location AS location_of_accident, AVG(damage_amount) AS
average_amount_of_damage)
FROM accident
JOIN participated
ON accident.report_number = participated.report_number
GROUP BY location);

SELECT * FROM average_damage_per_location;

-- Use the average_damage_per_location location you have just created to
-- find the location that has the highest average damage

SELECT MAX(average_amount_of_damage)
FROM average_damage_per_location;

DROP VIEW average_damage_per_location;

start drop-Accident-tables.sql;
```