COMP23111 2016 - 2017
EX3

Sameul Da Costa - 9708530

November 24, 2016

1 Comments

Two queries throw errors, these are parts 1(ii) and 1(iii), these *should* error as explained in the comments.

The following only includes my solution queries and not the scripts needed to create and populate the tables.

Assumption: In part 1 a (iii) I have assumed that the query should select out the department name as well as the maximum salary. Then in the following part have assumed the department name was not necessary, I hope this was the lecturer's intention.

```
-- 1a (i) ----------------

SELECT
DISTINCT    name AS "Student 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.';

-- 1a (ii) ----------------

COLUMN "Student ID" FORMAT A10;

SELECT  student.id  AS "Student ID",
        name        AS "Student Name"
FROM
    (
        SELECT ID AS studentID FROM takes
        MINUS
        SELECT ID FROM takes WHERE year < 2009
    )
INNER JOIN student ON student.id = studentID;

CLEAR COLUMNS

-- 1a (iii) ----------------

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

-- 1a (iv) ----------------

SELECT  MIN(MAX_SALARY)
FROM
    (
       SELECT   dept_name,
                MAX(salary) AS "MAX_SALARY"
       FROM     instructor
       GROUP BY dept_name
    );

-- 1b (i) ----------------
```

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

-- 1b (ii) ----------------
-- SHOULD ERROR
INSERT
INTO    course
VALUES ('CS-002',
         'Monthly Seminar',
         'Comp. Sci.',
         '0');

-- 1b (iii) ----------------

-- In the script that created the table we see
-- the code: check (credits > 0)
-- Which is a 'check constraint' in oracle which throws an error
-- if the condition isn't met, ie if the #credits given < 0
INSERT INTO course
VALUES ('CS-002',
         'Monthly Seminar',
         'Comp. Sci.',
         '0');

-- 1b(iv) and (v) ----------------

-- Have assumed missing out the other columns won't cause issues
-- since the schema doesn't define that they should be non null
-- (and obviously the missing ones don't make up part of the primary key)
INSERT INTO section (course_id,
                     sec_id,
                     semester,
                     year)
VALUES ('CS-001',
         1,
         'Fall',
         2009);

-- 1b (vi) ----------------

INSERT INTO takes (ID,
                   course_id,
                   sec_id,
                   semester,
                   year)
```

```
SELECT ID,
       'CS-001',
       '1',
       'Fall',
       '2009'
FROM   student
WHERE  dept_name='Comp. Sci.';

-- 1b (vii) ----------------

DELETE takes
WHERE  ID =
            (
                SELECT  ID
                FROM    student
                WHERE   name='Zhang'
            )
AND    sec_id=1
AND    course_id='CS-001'
AND    semester='Fall'
AND    year=2009;

-- 1b (viii) ----------------

DELETE  takes
WHERE   course_id IN
        (
            SELECT course_id
            FROM   course
            WHERE  LOWER(title) LIKE '%database%'
        );

-- 1b (ix) ----------------

-- Statement runs since the database schema has specified 'delete cascade' in
-- all cases where course_id is a foreign key, so the db knows to delete all
-- records where course_id is a foreign key (and takes this value!) so no error
-- is thrown - Source: Oracle docs
DELETE  course
WHERE   course_id='CS-001';

-- 2 (i) ----------------

SELECT  COUNT(report_number)
FROM    participated
INNER JOIN owns
ON      owns.license=participated.license
WHERE   owns.driver_id=
            (
```

```
                SELECT driver_id FROM person WHERE name='Jane Rowling'
            );

-- 2 (ii) ----------------

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

-- 2 (iii) ----------------

SELECT  *
FROM
(
    SELECT      name,
                SUM(damage_amount) AS "TOTAL_DAMAGE"
    FROM        person
    INNER JOIN  participated
    ON          person.driver_id=participated.driver_id
    GROUP BY    name
)
WHERE       TOTAL_DAMAGE > 3000
ORDER BY    TOTAL_DAMAGE ASC;

-- 2 (iv) ----------------

CREATE OR REPLACE VIEW average_damage_per_location
AS
    SELECT      location, AVG(damage_amount) "AVERAGE_DAMAGE"
    FROM        accident
    INNER JOIN  participated
    ON          accident.report_number=participated.report_number
    GROUP BY    location;

-- 2 (v) ----------------

SELECT  location
FROM    average_damage_per_location
WHERE   average_damage = (
                            SELECT  MAX(average_damage)
                            FROM    average_damage_per_location
                          );
```