Zheng Guo	A20368154
Name	CWID

Homework Assignment 2

Due Date: October 17th, 2017

CS425 - Database Organization

Please leave this empty!

2.1 ☐ 2.2 ☐

2.3 ☐ 2.4 ☐ 2.5 ☐ 2.6 ☐ 2.7 ☐ 2.8 ☐ 2.9 ☐ 2.10 ☐

2.11 ☐ 2.12 ☐

2.15 ☐ 2.16 ☐ 2.17 ☐ 2.18 ☐ 2.19 ☐ Sum ☐

Instructions

- Try to answer all the questions using what you have learned in class
- **When writing a query, write the query in a way that it would work over all possible database instances and not just for the given example instance!**
- Some questions are marked as bonus. You do not have to answer these questions to get full points for the assignment. However, you can get bonus points for these questions!
- **Please submit the homework electronically using blackboard**

Consider the following library database schema and example instance storing:

book

bookid	title	price	total_copies
1	Introduction of Algorithms	84.66	4
2	Database System Concepts	74.99	5
3	Stochastic Calculus for Finance I	41.02	3
4	Stochastic Calculus for Finance II	55.22	3

course

courseid	title	instructorid	textbookid
1	Algorithms	1	1
2	DB Organization	2	2
3	Advanced DB Organization	3	2
4	Math Finance I	1	3
5	Math Finance II	4	4

student

studentid	name	gpa
1	Tom	3.3
2	John	3.8
3	Mary	3.0
4	Kris	3.6
5	Alex	3.5

faculty

facultyid	name	salary
1	James	70000
2	Sarah	60000
3	Jay	80000
4	Rachel	70000
5	Paul	85000

enroll

studentid	courseid
1	1
1	2
2	1
4	3
4	4
5	5

book_checkout

date	bookid	studentid
2017-08-29	1	1
2017-09-02	4	4
2017-09-07	1	4

Hints:

- All the attributes that have integer values are of type INT; numbers with decimal point are of type NUMERIC; the attribute *date* of *book_checkout* relation is of type DATE; others are of type VARCHAR
- Attributes with black background form the primary key of an relation
- The attribute *instructorid* of relation *course* is a foreign key to relation *faculty*, and *textbookid* is a foreign key to relation *book*.

- The attribute *studentid* of relation *enroll* is a foreign key to relation *student*, and *courseid* is a foreign key to relation *course*.
- The attribute *bookid* of relation *book_checkout* is a foreign key to relation *book*, and *studentid* is a foreign key to relation *student*.

Part 2.1 SQL DDL (Total: 14 Points)

Question 2.1.1 (7 Points)

Write an SQL statement that adds an *advisorid* attribute to relation *student*, and sets it to be a foreign key to *facultyid* in relation *faculty*. In case a faculty's id is changed, the change would be reflected on *advisorid* attribute; in case a student's advisor left the school, *advisorid* would be set to NULL.

```
ALTER TABLE student
  ADD advisorid INT;
ALTER TABLE student
  ADD FOREIGN KEY (advisorid)
    REFERENCES faculty(facultyid)
    ON DELETE SET NULL
    ON UPDATE CASCADE
```

Question 2.1.2 (7 Points)

Write an SQL statement that adds a constraint to the *student* relation to make sure that the *gpa* attribute cannot be `NULL`, and that the value of this attribute has to be between 0 and 4. Furthermore, the default value for this attribute should be 3.

```
ALTER TABLE student
   ADD CONSTRAINT valcheck
   CHECK (gpa >= 0 and gpa <= 4);
ALTER TABLE student
   ALTER COLUMN gpa SET DEFAULT 3;
ALTER TABLE student
   ALTER COLUMN gpa SET NOT NULL;
```

Part 2.2 SQL Queries (Total: 56 + 10 BONUS Points)

Question 2.2.1 (5 Points)

Write an SQL query that returns the studentid and name of students who have overdue books (assume a book is due after 30 days), use construct `CURRENT_DATE` to access the current date in your query. Do not return any duplicate tuples.

```
SELECT distinct studentid,name from
   student natural join book_checkout
   WHERE date + interval '1 month' < current_date
```

Question 2.2.2 (5 Points)

Write an SQL query that returns the studentid and name of the student(s) whose gpa is higher than the average gpa of all students.

```
SELECT studentid,name FROM student
WHERE gpa > (SELECT avg(gpa) from student)
```

Question 2.2.3 (7 Points)

Write an SQL query that returns the facultyid and name of any faculty that does not teach any course but has a salary that is higher than 80,000.

```
SELECT facultyid,name FROM faculty
WHERE (facultyid NOT IN (SELECT instructorid FROM course) AND salary > 80000)
```

Question 2.2.4 (7 Points)

Write an SQL query that returns the bookid and title of books that are used as textbooks for more than one course.

```
SELECT bookid,title
FROM book NATURAL JOIN (SELECT textbookid AS bookid FROM course
    GROUP BY textbookid
    HAVING count(*) > 1)as duplicatebook
```

Question 2.2.5 (7 Points)

Write an SQL query that returns the studentid and name of students who have checked out books that are worth more than $100 in total.

```
SELECT distinct studentid FROM
book natural join (SELECT distinct studentid
                   FROM book natural join book_checkout
                   GROUP BY studentid
                   HAVING SUM(price)>100)as students
```

Question 2.2.6 (8 Points)

Write an SQL query that returns the studentid and name of students who checked out the textbook of a course that they did not enroll in.

```
SELECT studentid, name FROM student NATURAL JOIN
(SELECT distinct studentid FROM book_checkout
WHERE (studentid NOT IN (SELECT studentid FROM
enroll NATURAL RIGHT OUTER JOIN (SELECT studentid, courseid FROM
(SELECT courseid, textbookid as bookid From course)as books NATURAL JOIN
book_checkout)AS n
WHERE enroll.studentid = n.studentid)))AS nn
```

Question 2.2.7 (9 Points)

Suppose a student can check out as many books as the number of courses he/she is enrolled in. Write an SQL query that returns studentid and name of students who can not check out any more books. Your answer should include students who didn't enroll in any course.

```
SELECT studentid, name FROM student NATURAL JOIN
(SELECT studentid FROM student
WHERE (student.studentid NOT IN (SELECT studentid FROM enroll) OR student.studentid IN
(SELECT studentid FROM (SELECT studentid, count(bookid)AS n1 FROM book_checkout GROUP BY
studentid)as bo NATURAL JOIN
(SELECT studentid, count(courseid)AS n2 FROM enroll GROUP BY studentid)as en
WHERE n1>n2 OR n1= n2)))as ennn
```

Question 2.2.8 (8 Points)

Write an SQL query that returns the bookid and title of books that have 3 or more available copies.

```
SELECT title,bookid FROM book NATURAL LEFT OUTER JOIN

(SELECT title, textbookid as bookid FROM course)as co
WHERE total_copies >3 OR total_copies = 3
```

Question 2.2.9 BONUS (5 Points)

Write an SQL query which returns courseid and title of the course(s) that has/have the most expensive textbook.

```
SELECT co.courseid, title FROM

(SELECT bookid from book WHERE price = (SELECT max(price) FROM book))as coco

NATURAL JOIN

(SELECT courseid, title, textbookid as bookid FROM course)as co
```

Question 2.2.10 BONUS (5 Points)

Write an SQL query that returns the studentid and name of students that enrolled in all of the finance courses. A course is considered a finance course if the title of the course contains the string 'Finance'.

```
SELECT studentid FROM (SELECT studentid,count(courseid)as coun FROM (SELECT * FROM course

NATURAL JOIN enroll
   WHERE (title like '%Finance%'))as co
   GROUP BY studentid)as coco
WHERE (coun = (SELECT count(courseid) from course


where (title like '%Finance%')))
```

Part 2.3 SQL Updates (Total: 30 + 5 BONUS Points)

Question 2.3.1 (7 Points)

Delete all courses that no one is enrolled in.

```
delete from course
where course.courseid not in
(SELECT courseid FROM enroll CROSS JOIN student
WHERE enroll.studentid = student.studentid)
```

Question 2.3.2 (8 Points)

4 copies of a new book *Distributed and Cloud Computing* has been added to the library. The price is $50.00 for each copy. Add the information to the *book* relation. Assume that *bookid* is automatically maintained by the system.

```
insert into book(title,price,total_copies) values('Distributed and Cloud Computing',50.00,4)
```

Question 2.3.3 (6 Points)

One of the checkout records is incorrect. It turns out, the student with id 4 never checked out the book with id 1. He checked out the book with id 2. Update the information in *book_checkout* relation.

```
UPDATE book_checkout SET bookid = '2'
WHERE studentid = '4' AND bookid = '1'
```

Question 2.3.4 (9 Points)

Update the *gpa* in the *student* relation according to this rule:

- if it is negative, set it to 0
- if it is larger than 4, then set it to 4
- if it is `NULL`, set it to 3
- if none of the above applies do not change the gpa

Note that we expect you to write a single statement that implements this.

```
UPDATE student SET gpa = (CASE
                WHEN gpa is NULL THEN 0
                WHEN gpa > 4 THEN 4
                WHEN gpa < 0 THEN 0
                END)
WHERE (gpa is NULL OR gpa>4 OR gpa<0)
```

Question 2.3.5 BONUS (5 Points)

Update the salaries of faculty as their current salary + 10000 · (the number of courses they are teaching).

```
UPDATE faculty SET
salary = salary + coco.count * 1000 FROM
        (SELECT facultyid,count(courseid) FROM
            (SELECT courseid, instructorid as facultyid FROM course)as co
          NATURAL JOIN faculty
          GROUP BY facultyid)as coco
WHERE faculty.facultyid = coco.facultyid
```