Russi Sinha	A20411286
Name	CWID

Homework Assignment 2

Due Date: Oct. 18th, 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													

2.13		2.14		2.15		2.16		2.17		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!**
- **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 to 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: 10 Points)

Question 2.1.1 (5 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 references faculty(facultyId)
on delete set null
on update delete
```

Question 2.1.2 (5 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
modify gpa default 3 not null
add constraint chk_gpa check(gpa between 0 and 4)
```

Part 2.2 SQL Queries (Total: 60 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 unique studentId, name from student natural join book_checkout
where current_date -date > 30
```

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.

```
with average(value) as
(select avg(gpa) from student)
select studentId, name from student, average where student.gpa > average.value
```

Question 2.2.3 (6 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 faculty.facultyId, faculty.name from faculty left outer join course on course.instructorId =
faculty.facultyId where course.courseId is null and faculty.salary > 80000
```

Question 2.2.4 (6 Points)

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

```
with bk_count(id, count) as
(select textbookId, count(textbookId) from course group by textbookId having count(textbookId) > 1)
select book.bookId, book.title from book, bk_count where book.bookId = bk_count.id
```

Question 2.2.5 (6 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.

```
with book_price(studentId, price) as
(select studentId, sum(price) from book natural join book_checkout group by student)
select studentId, name from book_price, student where books.studentId = student.studentId and
books.price > 100
```

Question 2.2.6 (7 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 S where not exist
(select bookId from book_checkout where studentId = S.studentId except
(select textbookId from course natural join enroll where studentId = S.studentId))
```

Question 2.2.7 (8 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.

```
with ccount(sid, crs_count) as
(select studentId, count(courseId) from enroll group by studentId),
bcount(sid, bk_count) as
(select studentId, count(bookId) from book_checkout group by studentId),
check_count(sid) as
(select bcount.sid from ccount, bcount where bcount.sid = ccount.sid and bcount.bk_count >=
ccount.crs_count)
select studentId, name from student left outer join checkout on checkout.sid = student.studentId
```

Question 2.2.8 (7 Points)

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

```
with books_chk(bid, bcount) as
(select bookId, count(studentId) from book_checkout group by bookId),
allbooks(bid, title, btotal, bcount) as
(select books_chk.bid, book.title, book.total_copies,
case when bcount is null then 0 else bcount end as bcnt from book left outer join books_chk on
book.bookId = bookschk.bid)
select bid, title from allbooks where btotal – bcount >= 3
```

Question 2.2.9 (5 Points)

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

```
with book_max(value) as
(select max(price) from book),
bks(bid) as
(select bookId from book, book_max where book_max.value = book.price)
select courseId, title from course inner join bks on bks.bid = course.textbookId
```

Question 2.2.10 (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 s.studentId, s.name from student S where not exists
(select courseId from course where title like '%Finance%' except
(select E.courseId from enroll E where S.studentId = E.studentId))
```

Part 2.3 SQL Updates (Total: 30 Points)

Question 2.3.1 (5 Points)

Delete all courses that no one is enrolled in.

```
delete from course where courseId not in (select distinct courseId from enroll)
```

Question 2.3.2 (6 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 (5 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 bookId = 1 and student = 4
```

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 < 0 then 0
when gpa > 4 then 4
when gpa is null then 3
else gpa
end
```

Question 2.3.5 (5 Points)

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

```
update faculty F set salary = salary + 10000 *(select count(*) from course C where C.instructorId = I.facultyId)
```