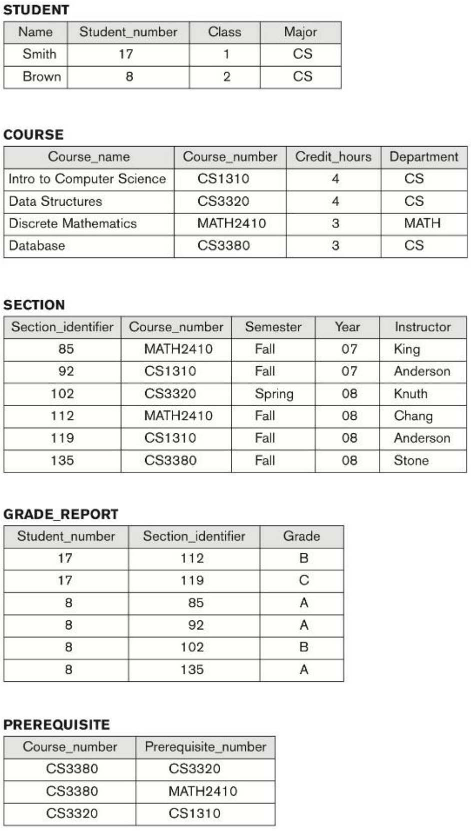
Introduction to Databases (Spring 2019)

Homework #1 (30 Pts, April 8, 2019)

Student ID _2014310407________

Name _Lee Jun Hyuk(이준혁)___

(1) [10 pts] Specify the queries in SQL on the following database schema.



STUDENT

Name	Student_number	Class	Major
Smith	17	1	CS
Brown	8	2	CS

COURSE

Course_name	Course_number	Credit_hours	Department
Intro to Computer Science	CS1310	4	CS
Data Structures	CS3320	4	CS
Discrete Mathematics	MATH2410	3	MATH
Database	CS3380	3	CS

SECTION

Section_identifier	Course_number	Semester	Year	Instructor
85	MATH2410	Fall	07	King
92	CS1310	Fall	07	Anderson
102	CS3320	Spring	08	Knuth
112	MATH2410	Fall	08	Chang
119	CS1310	Fall	08	Anderson
135	CS3380	Fall	08	Stone

GRADE_REPORT

Student_number	Section_identifier	Grade
17	112	B
17	119	C
8	85	A
8	92	A
8	102	B
8	135	A

PREREQUISITE

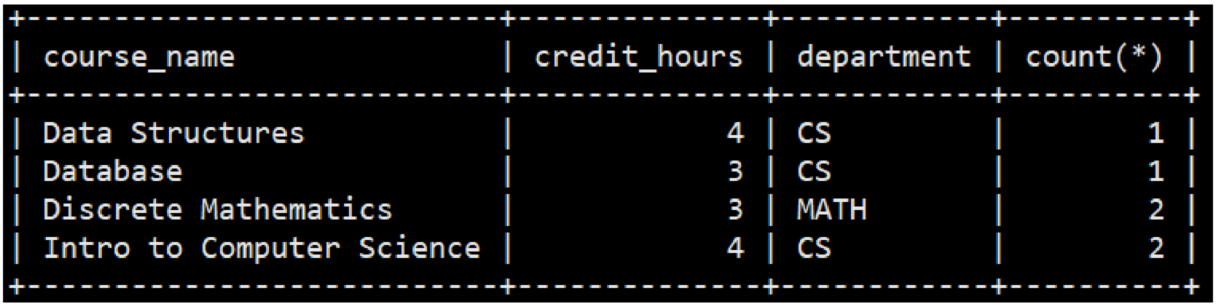
Course_number	Prerequisite_number
CS3380	CS3320
CS3380	MATH2410
CS3320	CS1310

(a) [3 pts] Retrieve the names, credit hours, departments of all courses along with the number of the sections of each course.

Answer:

```
SELECT course_name, credit_hours, department, count(*)

FROM      COURSE, SECTION

WHERE    COURSE.COURSE_NUMBER = SECTION.COURSE_ NUMBER

GROUP BY    SECTION_IDENTIFIER, COURSE_NAME, CREDIT_HOURS, DEPARTMENT ;
```

SQL Result:



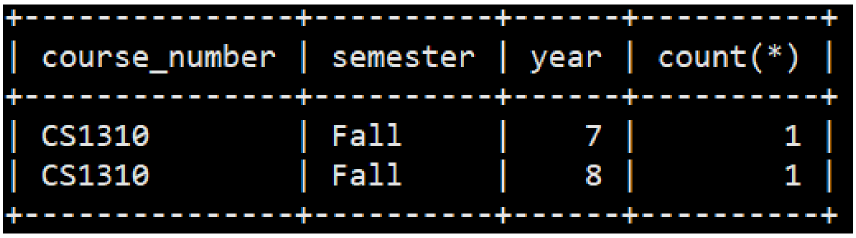
```
+---------------------------+--------------+------------+----------+
| course_name               | credit_hours | department | count(*) |
+---------------------------+--------------+------------+----------+
| Data Structures           |            4 | CS         |        1 |
| Database                  |            3 | CS         |        1 |
| Discrete Mathematics      |            3 | MATH       |        2 |
| Intro to Computer Science |            4 | CS         |        2 |
+---------------------------+--------------+------------+----------+
```

(b) [3 pts] For each section taught by Professor Anderson, retrieve the course number, semester, year, and number of students who took the section.

Answer:

```
SELECT   course_number, semester, year, count(*)

FROM     SECTION, GRADE_REPORT

WHERE    SECTION.SECTION_IDENTIFIER = GRADE_REPORT. SECTION_IDENTIFIER
         AND Instructor = ‘Anderson’

GROUP BY SECTION_IDENTIFIER, COURSE_NUMBER, SEMESTER, YEAR
```

SQL Result:



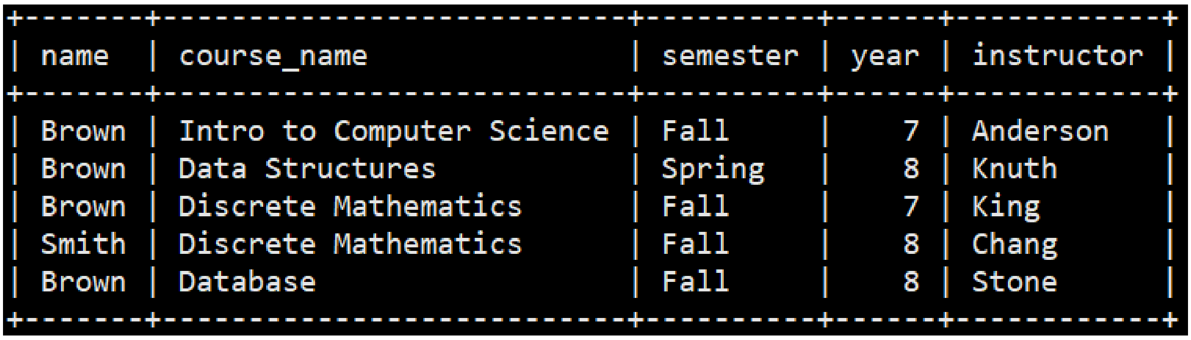
```
+---------------+----------+------+----------+
| course_number | semester | year | count(*) |
+---------------+----------+------+----------+
| CS1310        | Fall     |    7 |        1 |
| CS1310        | Fall     |    8 |        1 |
+---------------+----------+------+----------+
```

(c) [4 pts] In the grade records, if the grade is more than 'B' (‘A’ or ‘B’), retrieve student name, class name, semester, year and instructor. Join at least two tables!

Answer:

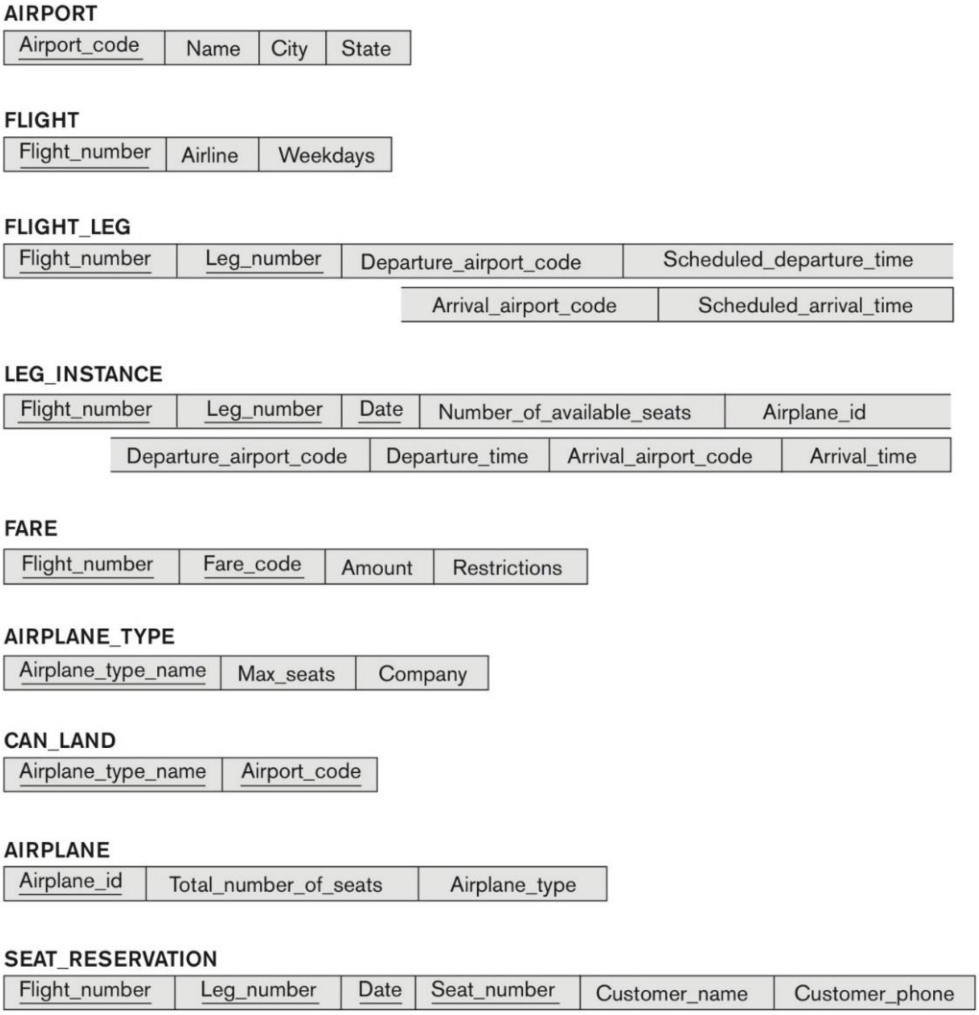
```
SELECT name, course_name, semester, year, instructor

FROM   STUDENT
LEFT JOIN GRADE_REPORT ON STUDENT.STUDENT_NUMBER = GRADE_REPORT.STUDENT_NUMBER
LEFT JOIN SECTION ON GRADE_REPORT.SECTION_IDENTIFIER = SECTION.SECTION_IDENTIFIER
LEFT JOIN COURSE ON COURSE.COURSE _NUMBER = SECTION.COURSE_NUMBER

WHERE   GRADE = ‘A’ OR GRADE = ‘B’;
```

SQL Result:



```
+-------+---------------------------+----------+------+------------+
| name  | course_name               | semester | year | instructor |
+-------+---------------------------+----------+------+------------+
| Brown | Intro to Computer Science | Fall     |    7 | Anderson   |
| Brown | Data Structures           | Spring   |    8 | Knuth      |
| Brown | Discrete Mathematics      | Fall     |    7 | King       |
| Smith | Discrete Mathematics      | Fall     |    8 | Chang      |
| Brown | Database                  | Fall     |    8 | Stone      |
+-------+---------------------------+----------+------+------------+
```

(2) [10 pts] Consider the AIRLINE relational database schema, which describes a database for airline flight information. Each FLIGHT is identified by a *Flight_number*, and consists of one or more FLIGHT_LEGs with *Leg_number*s 1, 2, 3, and so on. Each FLIGHT_LEG has scheduled arrival and departure times, airports, and one or more LEG_INSTANCEs— one for each Date on which the flight travels. FAREs are kept for each FLIGHT. For each FLIGHT_LEG instance, SEAT_RESERVATIONs are kept, as are the AIRPLANE used on the leg and the actual arrival and departure times and airports. An AIRPLANE is identified by an *Airplane_id* and is of a particular AIRPLANE_TYPE. CAN_LAND relates AIRPLANE_TYPEs to the AIRPORTs at which they can land. An AIRPORT is identified by an *Airport_code*.



AIRPORT

Airport_code | Name | City | State

FLIGHT

Flight_number | Airline | Weekdays

FLIGHT_LEG

Flight_number | Leg_number | Departure_airport_code | Scheduled_departure_time | Arrival_airport_code | Scheduled_arrival_time

LEG_INSTANCE

Flight_number | Leg_number | Date | Number_of_available_seats | Airplane_id | Departure_airport_code | Departure_time | Arrival_airport_code | Arrival_time

FARE

Flight_number | Fare_code | Amount | Restrictions

AIRPLANE_TYPE

Airplane_type_name | Max_seats | Company

CAN_LAND

Airplane_type_name | Airport_code

AIRPLANE

Airplane_id | Total_number_of_seats | Airplane_type

SEAT_RESERVATION

Flight_number | Leg_number | Date | Seat_number | Customer_name | Customer_phone

Specify all the referential integrity constraints that hold on the schema.

Ex) Table_name.column (FK) -> Table_name for PK

FLIGHT_LEG.FLIGHT_NUMBER --> FLIGHT

FLIGHT_LEG.DEPARTURE_AIRPORT_CODE --> AIRPORT

Answer:

FLIGHT_LEG.FLIGHT_NUMBER --> FLIGHT

FLIGHT_LEG.DEPARTURE_AIRPORT_CODE --> AIRPORT

FLIGHT_LEG.ARRIVAL_AIRPORT_CODE --> AIRPORT

LEG_INSTANCE.FLIGHT_NUMBER --> FLIGHT

LEG_INSTANCE.LEG_NUMBER --> LEG_INSTANCE

LEG_INSTANCE.DEPARTURE_AIRPORT_CODE --> AIRPORT

LEG_INSTANCE.ARRIVAL_AIRPORT_CODE --> AIRPORT

FARE.FLIGHT_NUMBER --> FLIGHT

CAN_LAND.AIRPLANE_TYPE_NAME --> AIRPLANE_TYPE

CAN_LAND.AIRPORT_CODE --> AIRPORT

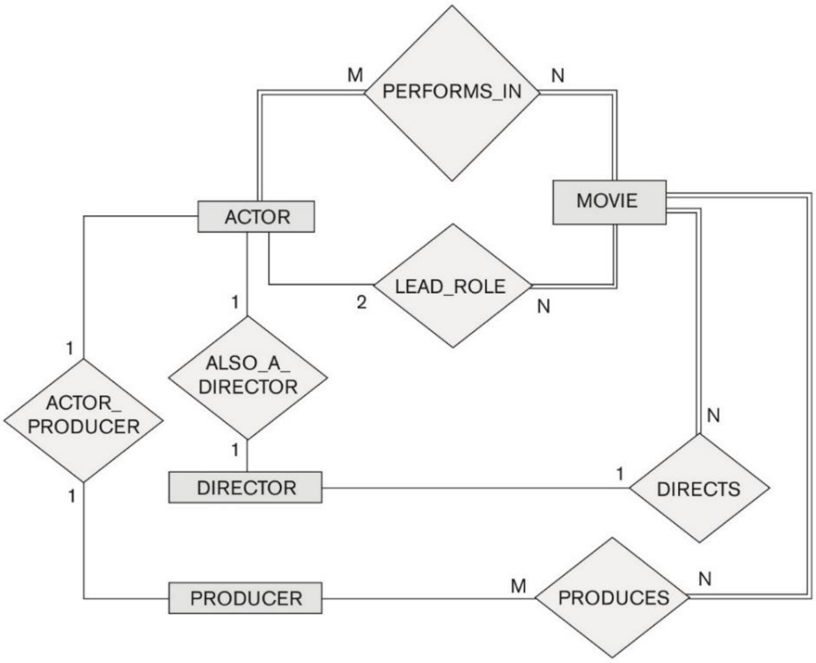
AIRPLANE.AIRPLNE_TYPE --> AIRPLANE_TYPE

SEAT_RESERVATION.FLIGHT_NUMBER --> FLIGHT

SEAT_RESERVATION.LEG_NUMBER --> LEG_INSTANCE

SEAT_RESERVATION.DATE --> LEG_INSTANCE

(3) [**10 pts**] Consider the ER schema for the MOVIES database as below. In the ER schema, ACTOR is used as a generic term and includes actresses. Given the constraints shown in the ER schema, answer the following statements with *true* or *false*. Explain why.



(a) There are no actors in this database that have been in no movies.

Answer

True

Since actor has total participation in a PERFORMS_IN relationship

(b) A movie can have no lead actor.

Answer

False

Since movie has total participation in a LEAD_ROLE relationship

(c) Every director has been an actor in some movie.

Answer

False

Since director has partial participation in a ALSO_A_DIRECTOR relationship

(d) An actor can be a director and producer at the same time.

Answer

True

There is no constraint about director and producer, actor can be both director and producer.

(e) Every producer has been an actor.

Answer

False

Since producer has partial participation in a ACTOR_PRODUCER relationship

(f) A movie has one director, but many producers.

Answer

True

Since DIRECTS relation is 1:N, while PRODUCES relation is M:N

(g) Every movie has a director who also acted in that movie.

Answer

False

Since director has partial participation in a ALSO_A_DIRECTOR relationship

(h) A producer cannot be an actor in some other movie.

Answer

False

There is 1:1 relationship at producer : actor and actor : director, so producer may be an actor

(i) Every producer should produce one or more movies.

Answer

False

Since producer has partial participation in a PRODUCES relationship

(j) A movie can have two lead actors.

Answer

True

Since LEAD_ROLE relationship is 2:N, so multiple lead actor may exist for a movie. Therefore, a movie can have two lead actors