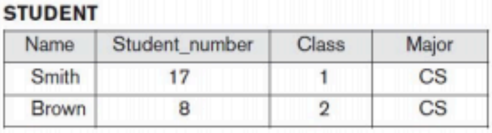
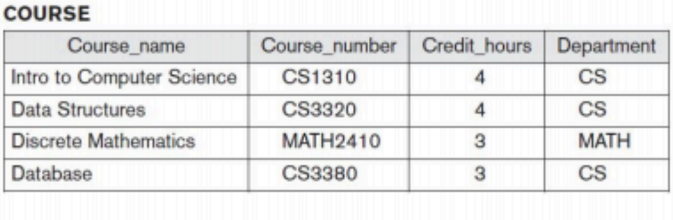
Naman Jain Vimal Kumar
1001670153

Consider the database below that stores student and course information. Answer questions 1, 2 and 3 based on this database.



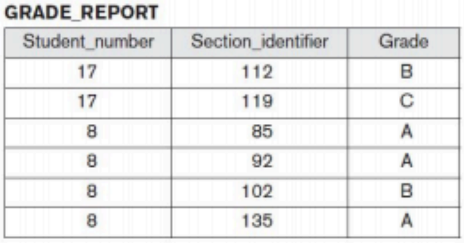
STUDENT

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



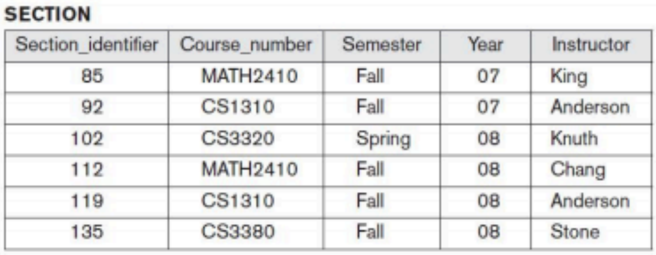
COURSE

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



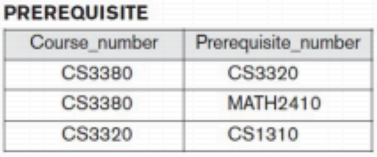
GRADE_REPORT

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



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



PREREQUISITE

Course_number	Prerequisite_number
CS3380	CS3320
CS3380	MATH2410
CS3320	CS1310

1) ***What are the referential integrity constraints that should hold on the database above? Write appropriate SQL DDL statements to define the database (Create table statements) Note: To show referential integrity Use the notation R.(A) --> S.(B) to represent a foreign key from the attribute A of R (the referencing relation) to S (the referenced relation). (30 points)***

We have multiple referential integrity constraints that hold for the above table. They are:

A. GRADE_REPORT.Student_number -> STUDENT.Student_number
B. GRADE_REPORT.Section_identifier -> SECTION.Section_identifier
C. SECTION.Course_number -> COURSE.Course_number
D. PREREQUISITE.Course_number -> COURSE.Course_number
E. PREREQUISITE.Prerequisite_number -> COURSE.Course_number

The DDL statements to define the database is:

- Creating STUDENT table –

```
CREATE TABLE STUDENT (
Name VARCHAR(30) NOT NULL,
Student_number INT PRIMARY KEY,
Class INT(2),
Major VARCHAR(20) NOT NULL
)
```

- Creating COURSE table –

```
CREATE TABLE COURSE (
Course_name VARCHAR(100) NOT NULL,
Course_number VARCHAR(15) PRIMARY KEY,
Credit_hours INT(2),
Department VARCHAR(20) NOT NULL
)
```

- Creating SECTION table –

```
CREATE TABLE SECTION (
Section_identifier INT(4) PRIMARY KEY,
Course_number VARCHAR(15),
Semester ENUM('Fall', 'Spring'),
Year YEAR NOT NULL,
Instructor VARCHAR(30) NOT NULL,
FOREIGN KEY (Course_number) REFERENCES COURSE(Course_number)
  ON DELETE CASCADE ON UPDATE CASCADE
)
```

- Creating GRADE_REPORT table –

```
CREATE TABLE GRADE_REPORT (
Student_number INT,
Section_identifier INT(4),
Grade ENUM('A', 'B', 'C', 'D', 'F'),
PRIMARY KEY (Student_number, Section_identifier),
FOREIGN KEY (Student_number) REFERENCES STUDENT(Student_number)
  ON DELETE CASCADE ON UPDATE CASCADE,
FOREIGN KEY(Section_identifier)REFERENCES SECTION(Section_identifier)
  ON DELETE CASCADE ON UPDATE CASCADE
)
```

- Creating PREREQUISITE table –

```
CREATE TABLE PREREQUISITE (
Course_number VARCHAR(15),
Prerequisite_number VARCHAR(15),
PRIMARY KEY (Course_number, Prerequisite_number),
FOREIGN KEY (Course_number) REFERENCES COURSE(Course_number)
  ON DELETE CASCADE ON UPDATE CASCADE,
FOREIGN KEY (Prerequisite_number)REFERENCES COURSE(Course_number)
  ON DELETE CASCADE ON UPDATE CASCADE
)
```

2) Specify the following queries in SQL on the database above. (30 pts.)

A. Retrieve the names of all students majoring in ‘CS’ (computer science).

```
SELECT Name
FROM STUDENT
WHERE MAJOR = 'CS'
```

B. Retrieve the names of all courses taught by Professor King in 2007 and 2008

```
SELECT C.Course_name
FROM COURSE AS C, SECTION AS S
WHERE C.Course_number = S.Course_number
  AND S.Instructor = 'King'
  AND (S.Year = '2007' OR S.Year = '2008')
```

C. For each section taught by Professor King, retrieve the course number, semester, year, and number of students who took the section.

```
SELECT S.Course_number, S.Semester, S.Year, COUNT(*) AS 'Number of Students'
FROM SECTION AS S, GRADE_REPORT AS G
WHERE S.Section_identifier = G.Section_identifier
  AND S.Instructor = 'King'
GROUP BY S.Section_identifier
```

D. Retrieve the name and transcript of each senior student (Class = 2) majoring in CS. A transcript includes course name, course number, credit hours, semester, year, and grade for each course completed by the student.

```
SELECT S.Name, C.Course_name, C.Course_number, C.Credit_hours,
  G.Grade, SE.Semester, SE.Year
FROM COURSE AS C, STUDENT AS S, SECTION AS SE, GRADE_REPORT AS G
WHERE C.Course_number =SE.Course_number
  AND G.Section_identifier = SE.Section_identifier
  AND G.Student_number = S.Student_number
  AND S.Class = 2
  AND S.Major = 'CS'
```

E. Retrieve the names and major departments of all straight A students (students who have a grade A in all their courses).

```
SELECT S.Name, S.Major
FROM STUDENT AS S, GRADE_REPORT AS GR
WHERE S.Student_number = GR.Student_number
  AND GR.Grade LIKE 'A'
GROUP BY S.Student_number
HAVING COUNT(*)=(SELECT COUNT(*)
                 FROM GRADE_REPORT AS GR2
                 WHERE GR2.Student_number = S.Student_number
                )
```

F. Retrieve the names and major departments of all students who do not have any grade of A in any of their courses.

```
SELECT S.Name, S.Major
FROM STUDENT AS S, GRADE_REPORT AS GR
WHERE S.Student_number = GR.Student_number
AND GR.Grade NOT LIKE 'A'
GROUP BY S.Student_number
HAVING COUNT(*)=(SELECT COUNT(*)
                 FROM GRADE_REPORT AS GR2
```

```
                    WHERE GR2.Student_number = S.Student_number
                   )
```

3) ***Write SQL update statements to do the following on the database shown above. (20 pints)***

- ***Insert a new student <'Johnson', 25, 1, 'MATH'> in the database.***

```
INSERT INTO STUDENT VALUES ('Johnson', 25, 1, 'MATH')
```

- ***Change the class of student 'Smith' to 2.***

```
UPDATE STUDENT SET Class = 2 WHERE Name LIKE 'Smith'
```

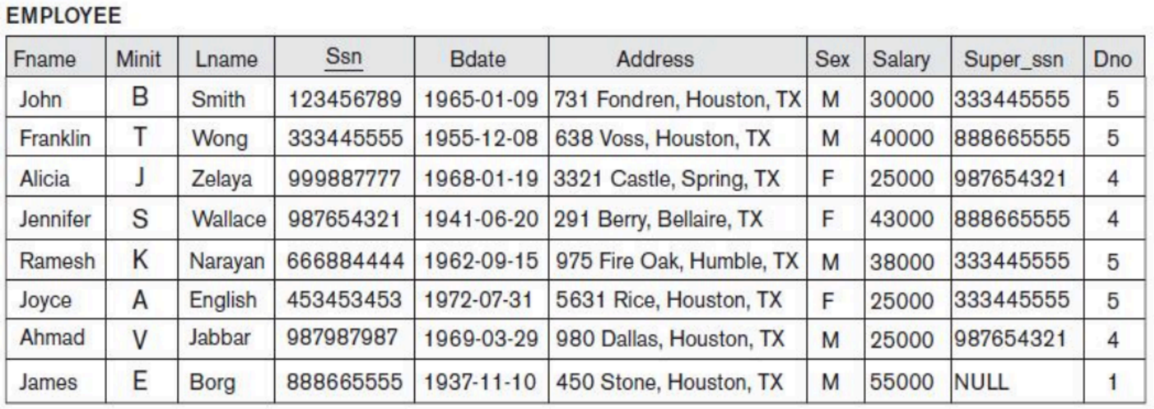
- ***Insert a new course <'Knowledge Engineering','COSC4390', 3,'COSC'>.***

```
INSERT INTO COURSE VALUES
('Knowledge Engineering', 'COSC4390', 3, 'COSC')
```

- ***Delete the record for the student whose name is 'Smith' and student number is 17***

```
DELETE FROM STUDENT WHERE Name LIKE 'Smith' AND Student_number = 17
```

Consider the snapshot of Employee table from Company database. Answer Questions 4 & 5 based on this table.



EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

4) ***Write SQL statement to create a table EMPLOYEE_BACKUP backup of EMPLOYEE table shown above. (5 pts.)***

```
CREATE TABLE EMPLOYEE_BACKUP (
Fname VARCHAR(30) NOT NULL,
Minit CHAR(1),
Lname VARCHAR(30) NOT NULL,
Ssn INT(9) PRIMARY KEY,
Bdate DATE,
Address VARCHAR(100),
Sex ENUM('M', 'F'),
Salary INT(8),
Super_ssn INT(9) REFERENCES EMPLOYEE_BACKUP(Ssn),
Dno INT(2)
);
```

```
INSERT INTO EMPLOYEE_BACKUP SELECT * FROM EMPLOYEE;
```

5) ***Consider the EMPLOYEE table's constraint EMPSUPERFK as follows***

```
CREATE TABLE EMPLOYEE ( . . . ,
Dno INT NOT NULL DEFAULT 1,
CONSTRAINT EMPSUPERFK
FOREIGN KEY (SUPERSSN) REFERNCES EMPLOYEE(SSN)
ON DELETE CASCADE
ON UPDATE CASCADE,
```

Answer the following questions:

- ***What happens when the following command is run on the database containing Employee table specified above. (5 pts.)***

  DELETE FROM EMPLOYEE WHERE LNAME = 'Borg'

  Here the definition of the table, reads that there is cascade function to be applied when there is a delete or update for the table Employee. Let us see what happens when we use the above query step wise:

  I. Since a record with Ssn '888665555' is being deleted, that implies first we need to delete records are linked to Ssn, that is the Super_ssn with '888665555'.
  II. The direct reports of Borg that is Franklin and Jennifer are supposed to be removed. But they also have direct reports of John, Ramesh & Joyce, and Alicia & Ahmad respectively.
  III. Eventually all the records in the Employee table are deleted.

- ***Is it better to CASCADE or SET NULL in case of EMPSUPERFK constraint ON DELETE? (5 points)***

  As we saw above with CASCADE, the complete EMPLOYEE table becomes empty. And as we all know, this is a very crucial data and deletion of such kind could loose a lot of information in the database. In this case, it is better to use SET NULL ON DELETE. Let us take the same query and see what the following:

  Since it us the SET NULL, the Super_ssn of Franklin and Jennifer will be set to NULL and not all records will be deleted except for Borg

6) List the data types that are allowed for SQL attributes. (5 pts.)

SQL Attributes have a huge range of data types for their attributes. They can be classified based on following:

Category	*Data Type*	*Description*
TEXT	CHAR(size)	Text of specific size. Can hold upto 255 characters.
	VARCHAR(size)	Text of variable size. Can hold upto 255 characters.
	TINYTEXT	Max. size of 255 characters
	TEXT	Max. size of 65,535 characters
	BLOB	Binary Large Objects which holds up 65,535 bytes of data
	MEDIUMTEXT	Max. size of 16,777,215 characters.
	MEDIUMBLOB	Binary Large Objects which holds up 16,777,215 bytes of data

	LONGTEXT	Max. size of 4,294,967,295 characters.
	LONGBLOB	Binary Large Objects which holds up 4,294,967,295 bytes of data
	ENUM(a, b, c, ..)	List of possible values. Can hold upto 65,535 values in an ENUM List
	SET	Array of possible values. Contains upto 64 list items and can store more than one choice.
NUMERIC	TINYINT(size)	Can hold integer value from -128 to 127 for signed or 0 to 255 for unsigned.
	SMALLINT(size)	Can hold integer value from -32,768 to 32,767 for signed or 0 to 65,535 for unsigned.
	MEDIUMINT(size)	Can hold integer value from -8,388,608 to 8,388607 for signed or 0 to 16,777,215 for unsigned.
	INT(size)	Can hold integer value from -2,147,483,648 to 2,147,483,647 for signed or 0 to 4,294,967,295 for unsigned.
	BIGINT(size)	For even more large integers
	FLOAT(size, d)	Small number with floating decimal point.
	DOUBLE(size, d)	Large number with floating decimal point.
	DECIMAL(size, d)	Stored as string, based on the size.
BIT	BIT(size)	Stores series of 0s and 1s of specified size
	BIT VARYING(size)	Stores series of 0s and 1s of variable size
BOOLEAN	True	Positive
	False	Negative
	NULL	None
DATE	DATE()	Format: YYYY-MM-DD
	DATETIME()	Format: YYYY-MM-DD HH:MI:SS
	TIMESTAMP()	Format: YYYY-MM-DD HH:MI:SS Here values are stored as number of seconds
	TIME()	Format: HH:MI:SS
	YEAR()	Format: YY or YYYY

References:

- https://stackoverflow.com/questions/1462497/creating-enum-variable-type-in-mysql
- https://stackoverflow.com/questions/2914936/mysql-foreign-key-constraints-cascade-delete
- https://www.w3schools.com/sql/default.asp
- Fundamentals of Database Systems, Sixth Edition, by Elmasri/Navathe