HW #4

- The SQL parts of this homework allow for, if you prefer, using MongoDB instead.

Part A

- **Consider the following scenario modeling courses, students, professors, departments, and the like at a single university in a single semester.**
- **Each student has a name, a unique PID, and an address.**
- **A professor has a name, a unique PID, and belongs to a department. The age of a professor can be one of "*old*," "*very old*," and "*still alive*".**

HW #4 (2)

- **Each course has a name, a number, an offering department, a maximum enrollment, and an actual enrollment. The actual enrollment must be at most the maximum enrollment.**

- **Each department offers only one course with each number.**

HW #4 (3)

- **Each department has a unique name. Each department has at most one chairperson who is its head (there are times when a department may not have a chairperson). Each chairperson can be the head of at most one department.**

- **Each student enrolls in a certain number of courses in the semester. At most one professor teaches each course.**
- **Each student receives a grade in each course he/she is enrolled in. In turn, each student evaluates the professor teaching the course.**

HW #4 (4)

- **A course can have multiple pre-requisites. A course can be a pre-requisites for multiple courses. A course can**not **be a pre-requisites for itself! A student enrolled in a course must have enrolled in all its pre-requisites.**

- **Suppose we come up with the following relations (or a very similar set of relations) to model this scenario:**

Students(PID: string, Name: string, Address: string)

Professors(PID: string, Name: string, Age: string, DepartmentName: string)

HW #4 (5)

Courses(Number: integer, DeptName: string, CourseName: string, MaxEnrollment: integer, ActualEnrollment: integer)

Departments(Name: string, ChairmanPID: string)

Take(StudentPID: string, Number: integer, DeptName: string, Grade: string, ProfessorEvaluation: integer)

Teach(ProfessorPID: string, Number: integer, DeptName: string)

PreReq(Number: integer, DeptName: string, PreReqNumber: integer, PreReqDeptName: string)

HW #4 (6)

- **Write MySQL-compatible SQL queries that solve each of the following problems:**

A1. **What are the PIDs of the students whose name is "David"?**

A2. **Which pairs of students live at the same address? It is enough to return the names of such students pairs.**

A3. **Which department have course that have pre-requisites in other departments?**

HW #4 (7)

A4. Compute the set of all courses that are their own pre-requisites? (have cycles)

A5. What are the names and address of the students who are taking “CS186”?

A6. What are the courses that the head of the CS department is teaching?

A7. Is there any department head who teaches a course in another department?

A8. Are there any students who are taking at least two courses taught by department heads?

HW #4 (8)

A9. Is there any professor whose age is “still alive” and who receives an average evaluation above 2.5?

A10. Is there any “straight A” student?

A11. Are there any students who are taking courses and receiving more grade A than grade B?

Sample Test Dataset

Departments

Name	ChairmanPID
CS	Ullman
EE	Knuth
ME	Lam
BIO	null
PHY	Reiss
MATH	Wegner

Professors

PID	Name	Age	DepartmentName
Widom	Jennifer	old	BIO
Canny	John	very old	EE
Ullman	Jeff	still alive	CS
Reiss	Steve	very old	PHY
Karp	Richard	still alive	MATH
Lam	Monica	old	ME
Chien	Andrew	old	PHY
Wegner	Peter	still alive	MATH
Hart	John	very old	BIO
Katz	Randy	very old	CS
Knuth	Don	still alive	EE
Barsky	Brian	old	EE

Courses

Number	Dept Name	Course Name	Max Enroll	Actual Enroll
132	ME	Dynamic Systems	120	118
61	CS	Data Structure	100	90
1	MATH	Calculus	150	132
123	EE	Digital Signal Proc	80	72
111	PHY	Modern Physics	40	39
109	ME	Heat Transfer	10	8
54	MATH	Linear Algebra	50	50
162	CS	Operating Systems	50	32
137	PHY	Quantum Mech	10	3
145	BIO	Genomics	5	2
186	CS	Database Systems	50	48
224	EE	Digital Comm	30	22

Teach

ProfessorPID	Number	DeptName
Knuth	123	EE
Reiss	54	MATH
Widom	145	Genomics
Ullman	61	CS
Karp	224	EE
Lam	132	ME
Reiss	111	PHY
Wegner	1	MATH
Ullman	186	CS
Reiss	137	PHY
Chien	109	ME
Barsky	162	CS

Students

PID	Name	Address
Zadeh	Lofti	Seattle, WA
Patterson	David	Los Angeles, CA
Smith	Alan	San Francisco, CA
Feiner	Steven	Boston, MA
Kuck	David	Bloomington, IN
Kender	John	Los Angeles, CA
Huang	Thomas	Atlanta, GA
Fischer	Michael	Madison, WI
Appel	Andrew	Miami, FL
Dobkin	David	Salt Lake City, UT
Li	Kai	Las Vegas, NV
Peterson	Larry	Chicago, IL

Take

StudentPID	Number	Dept Name	Grade	Professor Evaluation
Appel	111	PHY	B	2
Patterson	186	CS	B	3
Li	137	PHY	A	3
Huang	186	CS	A	4
Smith	109	ME	A	3
Appel	1	MATH	C	2
Huang	123	EE	A	4
Fischer	145	BIO	A	2
Zadeh	61	CS	A	1
Dobkin	123	EE	B	4
Huang	111	PHY	B	3
Li	162	CS	A	3
Kender	54	MATH	B	4

PreReq

Number	DeptName	PreReqNumber	PreReqDeptName
137	PHY	1	MATH
186	CS	61	CS
186	CS	54	MATH
145	BIO	132	ME
224	EE	54	MATH
162	CS	186	CS
111	PHY	132	ME
109	ME	224	EE
224	EE	61	CS
111	PHY	1	MATH
132	ME	145	BIO
54	MATH	162	CS
123	EE	54	MATH

HW #4 (9)

Part B

- **See Exercise 5.4**

- **Consider the following relational schema. An employee can work in more than one department; the "pct_time" field of the Works relation shows the percentage of time that a given employee works in a given department.**

Emp(eid: integer**, ename:** string**, age:** integer**, salary:** real**)**

Works(eid: integer**, did:** integer**, pct_time:** integer**)**

Dept(did: integer**, dname:** string**, budget:** real**, managerid:** integer**)**

HW #4 (10)

- **Write the following queries in SQL:**

B1. **Print the names and ages of each employee who works in both the Hardware department and the Software department.**

B2. **For each department with more than 20 full-time-equivalent employees (i.e., where the part-time and full-time employees add up to at least that many full-time employees), print the "did" together with the number of employees that work in that department.**

HW #4 (11)

B3. Print the names of each employee whose salary exceeds the budget of all of the departments that he or she works in.

B4. Find the "managerids" of managers who manage only departments with budgets greater than $1 million.

B5. Find the "enames" of managers who manage the departments with the largest budgets.

HW #4 (12)

B6. If a manager manages more than one department, he or she "controls" the sum of all the budgets for those departments. Find the "managerids" of managers who control more than $5 million.

B7. Find the "managerids" of managers who control the largest amounts.

B8. Find the "enames" of managers who manage only departments with budgets larger than $1 million, but at least one department with budget less than $5 million.

HW #4 (13)

Part C: For both questions, you are given the following relational schema.

Students (sid: integer, sname: string, age: integer)

Enrolled (sid: integer, cid: integer, grade: integer)

Courses (cid: integer, cname: string, credits: integer)

- Students are identified uniquely by **sid**, and courses by **cid**. Students enroll to take courses, and for each course they obtain a **grade** which is an integer. **sname** is the student name (string), **age** represents the student age and is an integer. **cname** is the course name (string), and **credits** is the number of credits for a particular course (integer).

HW #4 (14)

Write **SQL queries** for the following:

C1. Write a statement to create the table **Enrolled**. You do **not** need to provide create table statements for the other tables. Include necessary key constraints.

C2. Find the name(s) of students(s) with the youngest age.

C3. Find the ages of students who take only courses with less than four credits (implies they take at least one course).

C4. Find the ages of students who got grade 10 in a course named ‘Databases’.

HW #4 (15)

C5. Find the course identifier **cid** and the average age over enrolled students who are 20 or older for each course that has at least 50 enrolled students (of any age).

C6. Find the names of students who take all the four-credit courses offered and obtained at least grade 7 in every such course.

C7. Find the name(s) of students(s) who have the highest GPA (assume the GPA is computed only based on grades available in **Enrolled**).

HW #4 (16)

C8. Find the ages of students who take some course with 3 credits.

C9. Find the names of students who obtained grade at least 8 in some course that has less than 4 credits.

C10. Find the names of students who obtained only grades of 10 (implies that they took at least one course).

C11. Find the names of students who took a course with three credits or who obtained grade 10 in some course.

C12. Find the names of students who are enrolled in a single course.

HW #4 (17)

PartD

- You must manage a database of recipes for your favorite food show.

  Dishes(did:integer, dname:string, origin:string, popularity:integer)

  Recipes(did:integer, iid:integer, quantity:integer)

  Ingredients(iid:integer, iname:string, unitprice:integer)

HW #4 (18)

- There is a table of dishes, with a unique identifier, dish name, origin of the dish (e.g., ‘Italy’ or ‘SouthEast Asia’) and popularity (this is a numerical score calculated based on how many “likes” each dish obtains).

- There is also a table of ingredients: each ingredient has a unique identifier, a name, and price per unit (in dollars). Finally, the recipes table stores how much of each ingredient is needed for each dish (assume that the quantity given in recipes is of the same unit as the unit price is measured in for table ingredients).

HW #4 (19)

- Write SQL for the following queries:

D1. Find the dish names that do NOT contain any of the following ingredients: sugar, butter, starch澱粉.

D2. Find the ingredient names that cost at least $10 per unit and that appear in at least one dish with popularity higher than 10,000.

D3. Find the origin of dishes that use at least one unit of an ingredient called ‘saffron’番紅花粉香料.

D4. List the popularity of “exclusive” dishes, defined as dishes that contain only ingredients costing at least $50 per unit.

D5. Find the name and unit price of rare ingredients, i.e., those that appear in a single dish.