# Normalization Exercise – Chapter 4

Create a schema for tables containing the following information. Make sure to designate the primary key as underlined and the foreign keys as *italics*.

Students: We want to track: name, ID number, phone, classes they are enrolled in.

Faculty: We want to track: name, ID number, phone, classes they teach, department they belong to.

Classes: We want to track: Class ID information (e.g. CPTR 319 A), Location, Meeting times and of course enrollment.

---

ASSUMPTIONS: Its always important to state your assumptions!

1. Only one faculty member teaches a class
2. Classes may not meet at standard times although labs are listed separately.

Students(StudentID, FirstName, LastName, Phone)

Faculty(FacultyID, FirstName, LastName, Phone, *DepartmentID*)

Department(DepartmentID, DepartmentName)

Class(ClassID, *FacultyID*, ClassName, Location, *DepartmentID*)

Enrollment(*ClassID*, *StudentID*)

ClassTimes(*ClassID*, WeekDay, StartTime, EndTime) – We didn't do this in class because of time, but it is a multi-dependency as we noted. This is how you could take care of that problem.

Think about this in terms of a hierarchy of existence:

- A department has multiple faculty → Faculty has a foreign key of *DepartmentID*
- A department has multiple courses → Class has a foreign key of *DepartmentID*
- A faculty has many classes → Class has a foreign key of FacultyID
- You can't have enrollment without classs and students → Enrollment has foreign keys of both ClassID, StudentID – this is unique because enrollment depends on both of these for its existence. Hence the Primary key is composed of foreign keys.

# SQL Exercise

Using the schema you created, write SQL queries to answer the following questions.

1. Find the names of all students in any class taught by Dr. Munger
2. Find the names of all students in CPTR 327.
3. Find the names of all professors in the Computing Department
4. Find the count of all students in CPTR 327.
5. Find the average number of students in computing classes.
6. Find all classes with a total enrollment of greater than 30 (Order by count descending - added).

```
SELECT S.FirstName, S.LastName
FROM Students AS S, Enrollment AS E, Class AS C, Faculty AS F
WHERE S.StudentID=E.StudentID AND E.ClassID=C.ClassID AND C.FacultyID=F.FacultyID AND F.LastName='Munger';

SELECT DISTINCT S.FirstName, S.LastName
FROM Students AS S, Enrollment AS E, Class AS C
WHERE S.StudentID=E.StudentID AND E.ClassID=C.ClassID AND C.ClassID LIKE '%CPTR 327%';

SELECT COUNT(DISTINCT FirstName,LastName)
FROM Faculty, Department
WHERE Faculty.DepartmentID=Department.DepartmentID AND DepartmentName='Computing';

SELECT COUNT(*) AS CPTR327_Count
FROM (
        SELECT DISTINCT S.FirstName, S.LastName
        FROM Students AS S, Enrollment AS E, Class AS C
        WHERE S.StudentID=E.StudentID AND E.ClassID=C.ClassID AND C.ClassID LIKE '%CPTR 327%'
);  --  for a good discussion of using Distinct with count see:
    --  http://answers.google.com/answers/threadview/id/235114.html

SELECT AVG(ClassCount)
FROM (
        SELECT ClassID, COUNT(*) AS ClassCount
        FROM Enrollment AS E, Class AS C, Department D
        WHERE E.ClassID=C.ClassID AND C.DepartmentID = D.DepartmentID AND D.DepartmentName = 'Computing'
        GROUP BY ClassID
);

SELECT ClassID, COUNT(*) AS ClassCount
FROM Enrollment AS E, Class AS C, Department D
WHERE E.ClassID=C.ClassID AND C.DepartmentID = D.DepartmentID AND D.DepartmentName = 'Computing'
GROUP BY ClassID
HAVING ClassCount >30
ORDER BY ClassCount DESC;
```