**Model Student Course Registration – Use whatever technology you prefer, not necessarily JPA/Hibernate (preferred since that is what we use).**

I have created **schoolmanager** project for this. Following is the link >

<https://github.com/stambade/schoolmanager>

Student: ID, Name

Course: ID, Name

A student can take many courses and a course can have many students.

1).Write skeleton code of entity bean (or whatever technologies you like) classes to model Student and Course and student course registration.

> Please refer to entity StudentVO and CourseVO classes in com.demo.schoolmanager.entities package

2). Write a skeleton Student DAO class that support

> refer to StudentDao Class, [I have skipped service layer], we have all db access related methods here

2.1). add a new student along with their course registrations.  
 2.2). Delete a student.

2.3).Get all students, sorted by their name, for a given course with course name as input.

**Bonus Points**:

2.4). What if we want to record course scores? What possible changes need to be made?   
 Explain briefly.

> For this particular point :

Approach 1 > We can create new table of StudentMarksheet and here one column

will be student\_id and other columns will be named as per course\_<Score> for

ex. computer\_score

Approach 2 > This is Good approach

We can add one more column to the 'student\_course' table, which will be named

as course\_score, as below [Refer data.sql file for table queries]

course\_score varchar(50) DEFAULT NULL > we can call this third table as

marksheet as well

2.5). How to find all students who don’t register for a given course? > refer to the main class

> For all above 5 things, I have written SchoolmanagerMain class, which is entry point for application and dao methods will be called from here, I have mentioned details in the comments

**Notes**:

1. Code skeleton is enough. No need to write every single line. > tried writing explanatory code
2. Need both entity bean classes as well as table DDLs. > refer entity classes and **data.sql**
3. For the DAO/Repository classes need query details for relevant questions.
4. Show proper transaction management. > it is handled at dao layer in each method, we can also use @transactional at service layer methods if needed
5. **Show best practice(s)** when you can solve the problems with multiple approaches. If possible comment why one approach is better than the others.
6. We love Hibernate and also hate it!! Would love to hear your opinion on best practices. >

* Hibernate has capability to generate primary keys automatically.
* it has own query language which deals with Java objects and fields, it can also interact with db through sql query > we can use both whenever needed depending upon the situation
* if we change the database, then also our application will works as HQL is database independent > so we can write generic java code for multiple databases, here we can use separate data source and session factory for each db
* hibernate can auto create DB tables for you > we can enable / disable this
* hibernate supports caching mechanism, which can reduce our number of round trips to database > we can use second level cache if we have huge data to be loaded

I Don’t think there are many disadvantages of hibernate, but hibernate is little slower than pure JDBC, actually the reason being hibernate used to generate many SQL statements in run time, but i guess this is not the disadvantage

**Time**: 45 – 60 minutes.