**Problem set 1: First Normal Form**

1. **Give a set of FDs for the relation schema *R (A, B, C, D)*with primary key *AB*under which *R*is in 1NF but not in 2NF.**

AB → CD

B → D

1. **Convert the following Relation into a database tables by applying 1st normalization. R = {Student ID, Last Name, First Name, Course ID, Course Section, Course Name, Grade, Professor Last Name, Professor First Name, Bldg,  Office #}**

Many students can apply for same course. One student can apply for many courses. One professor can teach more than one courses. In these cases, relation R will have redundancies. Hence, the given relation R should be split into three relations:

Student = {Student ID, Last Name, First Name, Course ID, Grade}

Functional Dependencies:

Student ID → Last Name, First Name

Student ID, Course ID → Grade

Course = {Course ID, Course Section, Course name, Professor ID}

Functional Dependency:

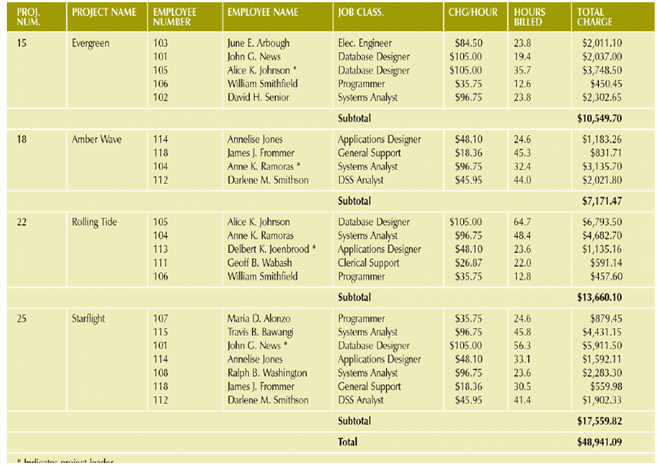
Course ID → Course Section, Course Name, Professor ID

Professor = {Professor ID, Professor Last Name, Professor First Name, Bldg, Office#}

Functional Dependency:

Professor ID → Professor Last Name, Professor First Name, Bldg, Office#

1. **Convert the below image into a database tables by applying 1st normalization.**



Project = {Project Number, project name}

Project Number → Project Name

Employee = {Employee Number, Employee Name, Job class, charge/hr}

Employee Number → Employee Name, Job class, charge/hr

Project\_Employee = {Project Number, Employee Number, hours billed, total charge}

Project Number, Employee Number → hours billed, total charge

1. **Grade\_report (StudNo, StudName, (Major, Adviser,  (CourseNo, Ctitle, InstrucName, InstructLocn,Grade)))**

**Advisor -> Major,  
StudNo,CourseNo,Major -> Grade,   
StudNo,Major -> Advisor  
Convert the above relation into 1NF**

To reduce redundancies in the above relation, it should be split into four relations.

Student = {StudNo, StudName}

StudNo → StudName

Advisor = {Advisor, Major}

Advisor → Major

Course = {CourseNo, Ctitle, InstrucName, InstrucLocation}

CourseNo → Ctitle, InstrucName, InstrucLocation

Grade\_report = {StudNo, Major, CourseNo, Grade}

StudNo, CourseNo, Major → Grade

**Problem set 2: Second Normal Form**

1. **Give a set of FDs for the relation schema *R (A, B, C, D)*with primary key *AB*under which *R*is in 2NF but not in 3NF.**

AB → C

C → D

1. **Apply 2nd Normalization for the above 2nd problem?**

In Student table, Student ID, Course ID is the candidate key. But last name and first name is dependant only on student ID which means a partial dependency is present in the table. So, the student table should be split into two tables.

Student = {Student ID, Last Name, First Name}

Student ID → Last Name, First Name

Student\_Course = {Student ID, Course ID, Grade}

Student ID, Course ID → Grade

Course = {Course ID, Course Section, Course name, Professor ID}

Course ID → Course Section, Course Name, Professor ID

Professor = {Professor ID, Professor Last Name, Professor First Name, Bldg, Office#}

Professor ID → Professor Last Name, Professor First Name, Bldg, Office#

1. **Apply 2nd Normalization for the above 3rd problem?**

As there is no partial dependency in 1st normalized form of the tables, they are also in second normal form.

Project = {Project Number, project name}

Employee = {Employee Number, Employee Name, Job class, charge/hr}

Project\_Employee = {Project Number, Employee Number, hours billed, total charge}

1. **Apply 2nd Normalization for the above 4th problem?**

There is no partial dependency in 1st normalized form of the tables. They are in 2NF.

Student = {StudNo, StudName}

Advisor = {Advisor, Major}

Course = {CourseNo, Ctitle, InstrucName, InstrucLocation}

Grade\_report = {StudNo, Major, CourseNo, Grade}