4.2 Normalisation

4.2.1 Database Design

Examine the following database (consisting of one table) that was designed to store the following

information:

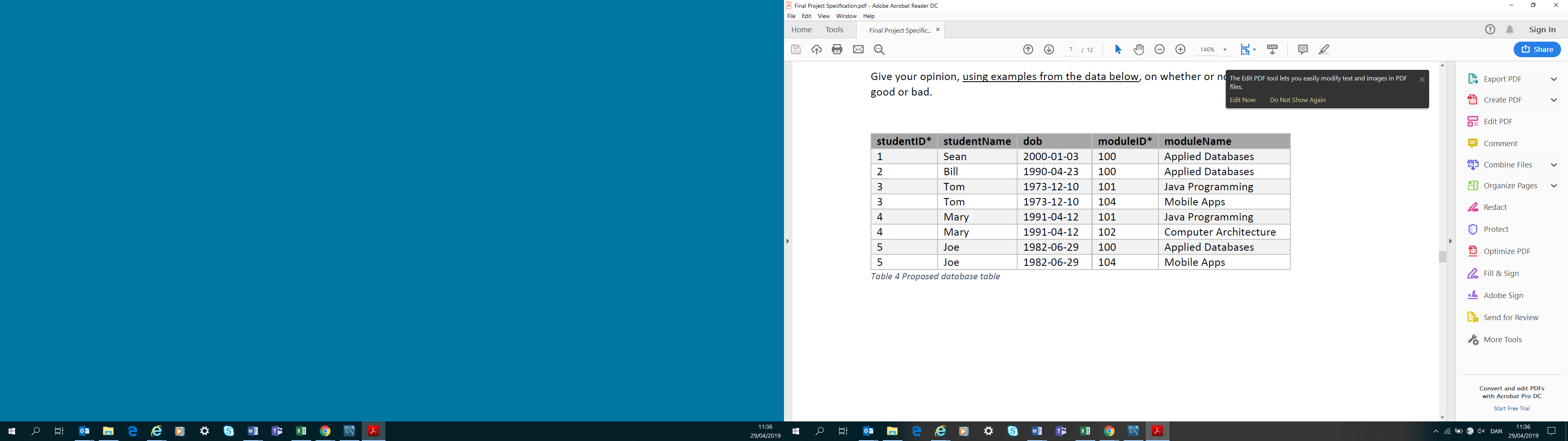
• Student ID

• Student Name

• Student Dob

• Modules Student is studying

Students can enrol in the college before deciding which modules to take, and not all modules are offered each year. The following database, consisting of one table with the primary key = studentID and moduleID, was designed. Give your opinion, using examples from the data below, on whether or not the current database is good or bad.



**ANSWER:**

The database is not optimal.

In order to improve it, I would recommend to firstly split it into 2 tables with a foreign key of moduleID. In this way a left join or inner join could be used to connect the modules with the students. Repetition will reduce and one change to table 2 will then be effective for all relevant students.

|  |  |  |  |
| --- | --- | --- | --- |
| **Table 1** | | **Table 2** | |
| **Student ID** | Primary Key, Sequential/unique | **Module ID** | Primary Key & Foreign Key matching with Module ID in Table 1  Sequential/unique |
| **Dob** | Date Format | **moduleName** | Varchar(x number of characters) |
| **Module ID** | Foreign Key matching with Module ID in Table 2 |  |  |

I would ensure that the StudentID and ModuleID reference numbers are at least sequential to ensure no duplicate Primary Keys are formed. In this case it appears this may be the case as each student has a unique ID, as does each module.

I would ensure that the surname of the student is included in table 1 to further reduce the risk of confusion.