4.2 Normalisation

# 4.2.1

This database is a bad design for the following reasons.

1. Firstly there are duplicates in every column. A good database design would have no duplicates if possible. If an item needed to be changed for example mobiles apps changed its name to app development. This would have to be done across multiple rows in this table. If this table was scaled to the size of a college module that could increase to hundreds of rows. This increases the chances of mistakes.
2. If a student dropped out of college or a module was dropped all occurrences of these modules or students would have to be deleted across multiple rows in this case;
3. If students are supposed to be able to register before they have selected there modules which is stated above this would be impossible as there is a dual primary key of the students id and module id. Primary keys cannot be null so without a moduleid t match the student id registering would be impossible.
4. If not every module is offered every year on years that modules are not offered they would have to be completely deleted and added back in again on years they were being offered again. Again time consuming.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **StudentID\*** | **studentName** | **dob** | **moduleID\*** | **moduleName** |
| 1 | Sean | 2000-01-03 | 100 | Applied Databases |
| 2 | Bill | 1990-04-23 | 100 | Applied Databases |
| 3 | Tom | 1973-12-10 | 101 | Java Programming |
| 3 | Tom | 1973-12-10 | 104 | Mobile Apps |
| 4 | Mary | 1991-04-12 | 101 | Java Programming |
| 4 | Mary | 1991-04-12 | 102 | Computer Architecture |
| 5 | Joe | 1982-06-29 | 100 | Applied Databases |
| 5 | Joe | 1982-06-29 | 104 | Mobile Apps |

1. Below I have normalised this table into two separate tables A module table and a Student table. This design fixes all of the problems faced above. The module table holds the module name and the module id while the student table holds studentid, studentname and dob.
2. The tables are broken up into parts that make sense having all module data and all student data separate. The similar data can be seen from the highlighted sections above.
3. There is now no duplicate data in either table
4. If a module or student has to be deleted or updated this only has to be done across one row now.
5. The removal of duplicates makes the tables easier to read.
6. It is possible for a student to now register before they have selected there modules as there is only a primary key on student id now in the student table.
7. It possible with this schema for a module that is not being run this year to stay in the module table until it is being run again and not need to be deleted and re added as it would have in the previous design.
8. The design below also features too more tables which are associated tables made up of the primary keys of the module and student tables set as foreign keys.
9. The students module table shows the module in which a student is enrolled in. This has the module id set as its primary key. And shows this information much clearer than the previous tables design. If a module was cancelled or deleted it would be deleted from here as well.
10. The second table is the module enrolment table this is also made up of the student id and moduleid but in this case this case the student is the primary key. This shows all those enrolled in a much clearer format. Again if a student dropped out of college and was removed from the student table it would update here

|  |  |  |
| --- | --- | --- |
| **Student Table** | | |
| **StudentId\* (PK)** | **studentName** | **Dob** |
| 1 | Sean | 2000-01-03 |
| 2 | Bill | 1990-04-23 |
| 3 | Tom | 1973-12-10 |
| 4 | Mary | 1991-04-12 |
| 5 | Joe | 1982-06-29 |

|  |  |
| --- | --- |
| **Module Enrolment** | |
| **Student ID (FK) (PK)** | **Module ID (FK)** |
| 1 | 100 |
| 2 | 100 |
| 5 | 100 |

|  |  |
| --- | --- |
| **Students Modules** | |
| **Student ID (FK)** | **Module ID (FK) (PK)** |
| 3 | 101 |
| 3 | 104 |

|  |  |
| --- | --- |
| **Module Table** | |
| **moduleID\* (PK)** | **moduleName** |
| 100 | Applied Databases |
| 101 | Java Programming |
| 102 | Computer Architecture |
| 104 | Mobile Apps |