G00007918
Applied Databases Project Normalisation Section

Q. 4.2.1

There is a lot of data redundancy in the given table, as it hasn't been normalised. This is not a very efficient way to store data. Without any normalisation of a relational database, we would create giant tables that have a lot of repetitive data and take up a large amount of memory. By increasing the complexity of the data model, and partitioning the data into separate tables, we increase the efficiency of the database.

Mary, Joe and Tom are entered twice each into the StudentName columns. To improve the efficiency of the data storage, I would partition the StudentName column, along with the dob column to a separate table, to have a unique record per student. This table would have a 1:1 relationship. See figure 1 below.

| /* Student table */ | | |
|---|---|---|
| studentID (primary key) | 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 |

*Figure 1*

There is also a lot of repetition/redundancy of data in the moduleID and moduleName columns of the original table. I would also separate out these columns into another 1:1 table with unique/distinct values (see example table in figure 2), and then reference them in a 3rd (1:many) table that combines the studentIds with the moduleIDs (see example table in figure 3). The primary keys of both the Module table and the Student table are referenced in the StudentModules table as foreign keys.

| /* Module table */ | |
|---|---|
| moduleID (primary key) | moduleName |
| 100 | Applied Databases |
| 101 | Java Programming |
| 102 | Mobile Apps |
| 103 | Computer Architecture |

*Figure 2*

| /* StudentModules table */ | | |
|---|---|---|
| studentModuleID (primary key) | moduleID (foreign key) | studentID (foreign key) |
| 1 | 100 | 1 |
| 2 | 100 | 2 |
| 3 | 101 | 3 |
| 4 | 104 | 3 |
| 5 | 101 | 4 |
| 6 | 102 | 4 |
| 7 | 101 | 5 |
| 8 | 104 | 5 |

*Figure 3*

A join query can be done between the primary keys of the Student table, the Module table and their foreign keys in the StudentModule table to find which students are taking which courses, thus reducing data redundancy in the data model. See an example design of the new data model ERD below in figure 4.

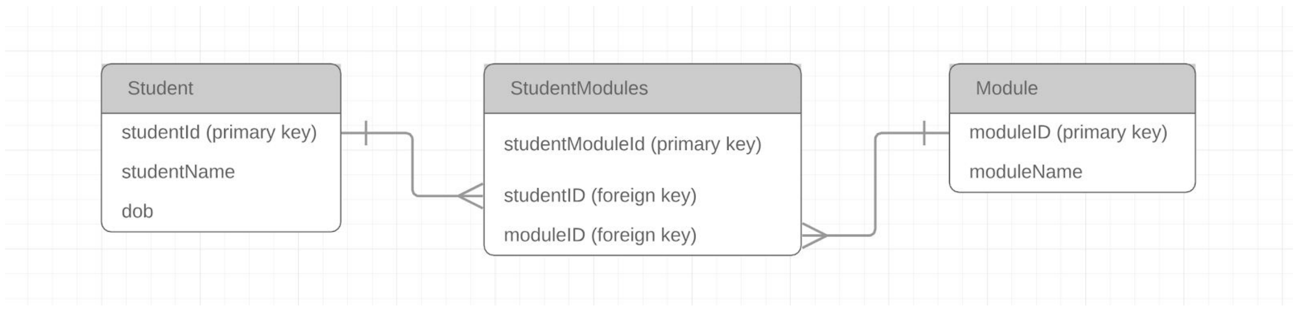

*Figure 4*

The above data model design (figure 4) follows the 3rd normal form, which **i**n most practical applications, is the most appropriate form of normalisation, and adheres to the following design rules: (1)(2)

- Each table cell should contain a single value (1st normal form).
- Each record in each table is unique (1st normal form).
- Single column primary keys (2nd normal form) - Users will only be able to insert values into a foreign key field that exist as a primary key in the parent table ie: we cannot insert a new studentID in the StudentModules table if it doesn't exist in the Student table. This helps to maintain referential integrity.
- There are no transitive functional dependencies  (3rd normal form) – if we update a non-key column, it won't cause data in the other columns to change.

References:
1 Guru99, 'What is Normalization? 1NF, 2NF, 3NF & BCNF with Examples', accessed on 10th May 2019, <https://www.guru99.com/database-normalization.html>
2 Codd, E. F., 'Relational Database: A Practical Foundation for Productivity', Communications

of the ACM  Volume 25 Issue 2, Feb 1982, New York, Pages 109-117