4.2 Normalisation

4.2.1 Database Design

In this section of the project, we have been provided with the design of a database which is to be used by a college to store information regarding which students are studying which subjects. As defined in the project brief, the following information is to be stored in the database:

- Student ID
- Student Name
- Student DOB
- Modules that the student is studying

There are some other points to note about this database:

- Students can enrol in the college before deciding which modules to take
- Not all modules are offered each year

The database is currently designed as follows:

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

Note that this design stores all information on the same table, and uses the studentID and moduleID as the primary key.

In my opinion, this is a very simple design. Keeping all data on one table means that it is easier to design and set up the database as there is only one table to create. It also makes the schema for the database more simplistic. As this database is so simple, the design is adequate for the needs of it. For example, the information that is stored about the students (ID, Name and DOB), is not likely to change so maintenance of this part of the database is unlikely to ever occur.

Where it starts to fall down is when we take into account the fact that students can enrol in the college before selecting any modules. As the database has been designed to use both the StudentID and ModuleID for the primary key, in means that values are required for both in order to create a record. MySQL does not allow NULL values in a primary key. Therefore if we have a new

student, Bob, who wants to enrol in the college but is waiting to see what modules are available before selecting them we could try to update the table as follows:

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
6	Bob	1984-05-18	NULL	NULL

However, this would not be allowable in MySQL. This means that the database is poorly designed when considering the user requirements for it. Another issue is that not all modules are offered each year, so there is no place to store the modules that are not in use as MySQL will not allow us to enter a module unless it is linked to student (again this is because records in this table require both a ModuleID and a StudentID).

In order to improve the database I would suggest separating the data into 'master' data and 'transactional' data, and storing the master data on separate tables, but allowing them to be linked to the transactional data tables as required. So, to store the data in this database I would propose using three tables instead of one. We would have 2 two master data tables (one for the Students, one for the Modules), and one transactional data table (called 'Enrolments', to store the list of course enrolments in the college).

The schema, and table for the proposed 'students' table is shown below:

students schema
studentID integer Primary Key
studentName varchar(20)
dob (date)

students		
studentID*	**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

According to the schema this table has 3 attributes, studentID (which is an integer and the Primary key for the table), studentName which is a varchar, and dob which is a date. This allows us to create a full list of students and keep it maintained separately from any enrolments.

The schema and table for the proposed ‘modules’ table is shown below:

modules schema
moduleID integer Primary Key
moduleName varchar(40)

modules	
moduleID*	**moduleName**
100	Applied Databases
101	Java Programming
102	Computer Architecture
104	Mobile Apps

According to the schema, the modules table has 2 attributes, moduleID (which is an integer and the Primary key for the table)and moduleName which is a varchar. Note the benefit of this table is that it allows us to maintain a full list of modules whether or not they are currently on offer. The last table in the database is the ‘enrolments’ table. The proposed enrolments table is shown below with its schema:

enrolments schema	
studentID integer Primary Key	FOREIGN KEY REFERENCES students(studentID)
moduleID integer Primary Key	FOREIGN KEY REFERENCES modules(moduleID)

enrolments	
studentID*	**moduleID***
1	100
2	100
3	101
3	104
4	101
4	102
5	100
5	104

It is proposed that this table will have two attributes, the studentID and moduleID, both of which are integers and both making up the Primary Key. Note that both attributes reference the ID field their respective tables as a Foreign Key.

The final part of this exercise is to demonstrate the MySQL query that would be required in order to show the original dataset:

```
SELECT
     et.studentID,
     st.name,
     st.dob,
     et.moduleID,
     mt.moduleName
FROM
     enrolments et
INNER JOIN students st
     ON st.studentID = et.studentID
INNER JOIN modules mt
     ON mt.moduleID = et.moduleID;
```

In this section of the project, I have discussed the concept of database normalisation. I have assesses an example database that was provided and looked at the positive and negative aspects for designing a simple database in that manner. I then proposed what I consider to be a better, more usable and more scalable method for designing this database. Finally I described the best way to interrogate my new database design in order to present the information in the way that the original database presented it.