Normalising Databases

Portfolio Exercise

Brief

SQL and programming languages can approach databases in different ways. Both deal with entities that you store information about (using classes in programming languages) but how they connect to each other can be quite different. SQL relies heavily on primary keys to link tables whereas, in a programming language, you can use arrays and references. So, for example, instead of linking a student table to a class table, the student object in a programming language could include an array of the classes the student is taking. Each class in the array, in turn, is a reference to a class over in the class array.

While this works in programming, it is considered bad design for databases. Databases tend to be larger, have more complex relationships and need to be fast. Arrays and references are not supported and to link two tables in a relationship requires you store a foreign key.

Your job is to take a database program written using classes and objects redesign it as a fully normalised SQL database. No coding or queries are required and you don't need to know any particular programming language, as this is a design exercise. You do need to be familiar with programming languages in general, particularly classes, objects and arrays.

I recommend you re-read the normalisation notes as the database will need some tweaks to make it 3NF.

The database in the program is set up as follows.

Customer

- Customer ID
- First name
- Surname
- Age
- Date of birth
- Phone number
- Address
- Customer loyalty level (Silver, Gold or Platinum)
- Discount (2% for silver, 5% for gold, 10% for platinum)
- Orders (an array of orders that the customer has made. See below.)

Orders

- Customer (a reference to the customer object representing the customer that made the order)
- Products (an array of products that make up the order)

Products

- Product ID
- Name
- Company
- Price
- Stock