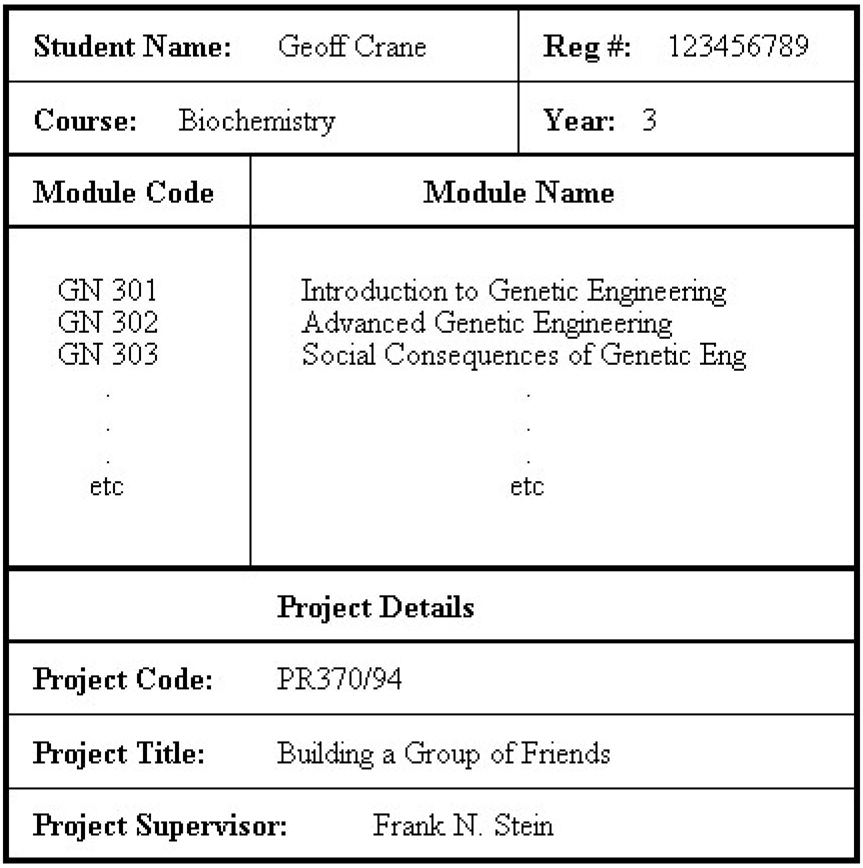
Normalization Step by Step

Extract Data from Data Source and represent it in an Un-normalized Form

The data source will usually be an input or an output of the system, whether a screen, a form, a format or a report. Let's take a student module registration form



Student Name: Geoff Crane	**Reg #:** 123456789
Course: Biochemistry	**Year:** 3
Module Code	**Module Name**
GN 301 GN 302 GN 303 . . . etc	Introduction to Genetic Engineering Advanced Genetic Engineering Social Consequences of Genetic Eng . . . etc
Project Details	
Project Code:	PR370/94
Project Title:	Building a Group of Friends
Project Supervisor:	Frank N. Stein

Data to be Normalized

To represent that data in un-normalized form, simply write the *name* of each data item, showing where there are repeating groups. Do this by enclosing them in brackets, which may be nested if there are repeating groups in the repeating groups. So, for the above document:

```
UNF
Student Number
Course
Year
Student Name
(Module Code
Module Name)
Project Code
Project Title
Project Supervisor
```

When you have your data in UNF you need to pick a key. This should ideally be unique on the data source, but you may have to use a combination of items to get a unique identifier. In theory, the key can be anything, but it makes sense to have a "reasonable" key, and to avoid textual keys. So here, although Project Code is unique on the document, and could in theory be used as the key, it makes more sense to use Student Number, as student is what we are storing data about we have underlined it.

Move from Un-normalized Form to First Normal Form

To go to 1NF, take out the repeating groups, *taking the key with them*, and remove them to separate "relations", and identify new key(s) for them as well, so that they too are uniquely identifiable.

So for our example:

```
UNF
Student Number
Course
Year
Student Name
(Module Code
Module Name)
Project Code
Project Title
Project Supervisor
```

```
1NF
Student Number
Course
Year
Student Name
Project Code
Project Title
Project Supervisor

Student Number
Module Code
Module Name
```

Move from First Normal Form to Second Normal Form

To go from 1NF to 2NF, just look at the relations with more than one key. Check that each data item within them *depends on all keys*. If it doesn't, remove it, together with the key(s) on which it does depend, to a new relation.

In our example, we only have one relation with a compound key:

Student Number

Module Code

Module Name

To work out whether the data item (in this case), depends on both keys, try thinking about the actual data values. If the data item depends on a key, its value will change if there is a different value for the key. So, looking at this relation, we can consider the value of Module Name with different keys;

Student Number:	123456789
Module Code:	GN301
Module Name:	Introduction to Genetic Engineering

With a different Student Number, we would get the following values:

Student Number:	987654321
Module Code:	GN301
Module Name:	Introduction to Genetic Engineering

With a different Module Code, we would get the following values:

Student Number:	987654321
Module Code:	GN302
Module Name:	Advanced Genetic Engineering

Therefore, we can see that Module Name depends *only* on Module Code, so our 2NF looks like this:

```
UNF
Student Number
Course
Year
Student Name
(Module Code
Module Name)
Project Code
Project Title
Project Supervisor
```

```
1NF
Student Number
Course
Year
Student Name
Project Code
Project Title
Project Supervisor

Student Number
Module Code
Module Name
```

```
2NF
Student Number
Course
Year
Student Name
Project Code
Project Title
Project Supervisor

Student Number
Module Code

Module Code
Module Name
```

We are left with a relation containing only the codes for student and module, but this is fine; we may well find data associated with such a relation on another document, (such as a record of student marks), and in any case, we do need to know which student is taking which module.

Move From Second Normal Form to Third Normal form

To go to 3NF, we examine all relations, to check that the data items that each contains depends on the key(s), rather than on another data item. The test is a similar one as to 2NF - would the data item's value change if the value of another data item was changed?

So for our example, we can see that Project Title and Project Supervisor would change if Project Code were altered. These data items should therefore be removed to another relation, using the data item they depend on as their key. This data item should be left in the original relation, and should be marked as a foreign key, thus:

UNF	**1NF**	**2NF**	**3NF**
Student Number Course Year Student Name (Module Code Module Name) Project Code Project Title Project Supervisor	Student Number Course Year Student Name Project Code Project Title Project Supervisor	Student Number Course Year Student Name Project Code Project Title Project Supervisor	Student Number Course Year Student Name Project Code#
	Student Number Module Code Module Name	Student Number Module Code	Student Number Module Code
		Module Code Module Name	Module Code Module Name
			Project Code Project Title Project Supervisor

The relations defined in 3NF are said to be normalized.

Exercise: Normalizing data to third normal form

Using the following data from a Pizza Delivery Shop – first of all convert into un-normalized form (UNF) then –go through the steps of converting it into 1NF, then into 2NF and finally into 3NF.

OrderNo:1259
OrderDate:12/11/2012
Customer ID: BB123
Customer Name: Barry Baker
CustomerAddress:42NewStreet,Birmingham
Postcode: HH12WER

Pizza Code	Pizza Name	Quantity	Price
P	Pepperoni	1	5.00
MF	Meat Feast	1	5.50
V	Veggie	2	10.00
		Total Price	20.50