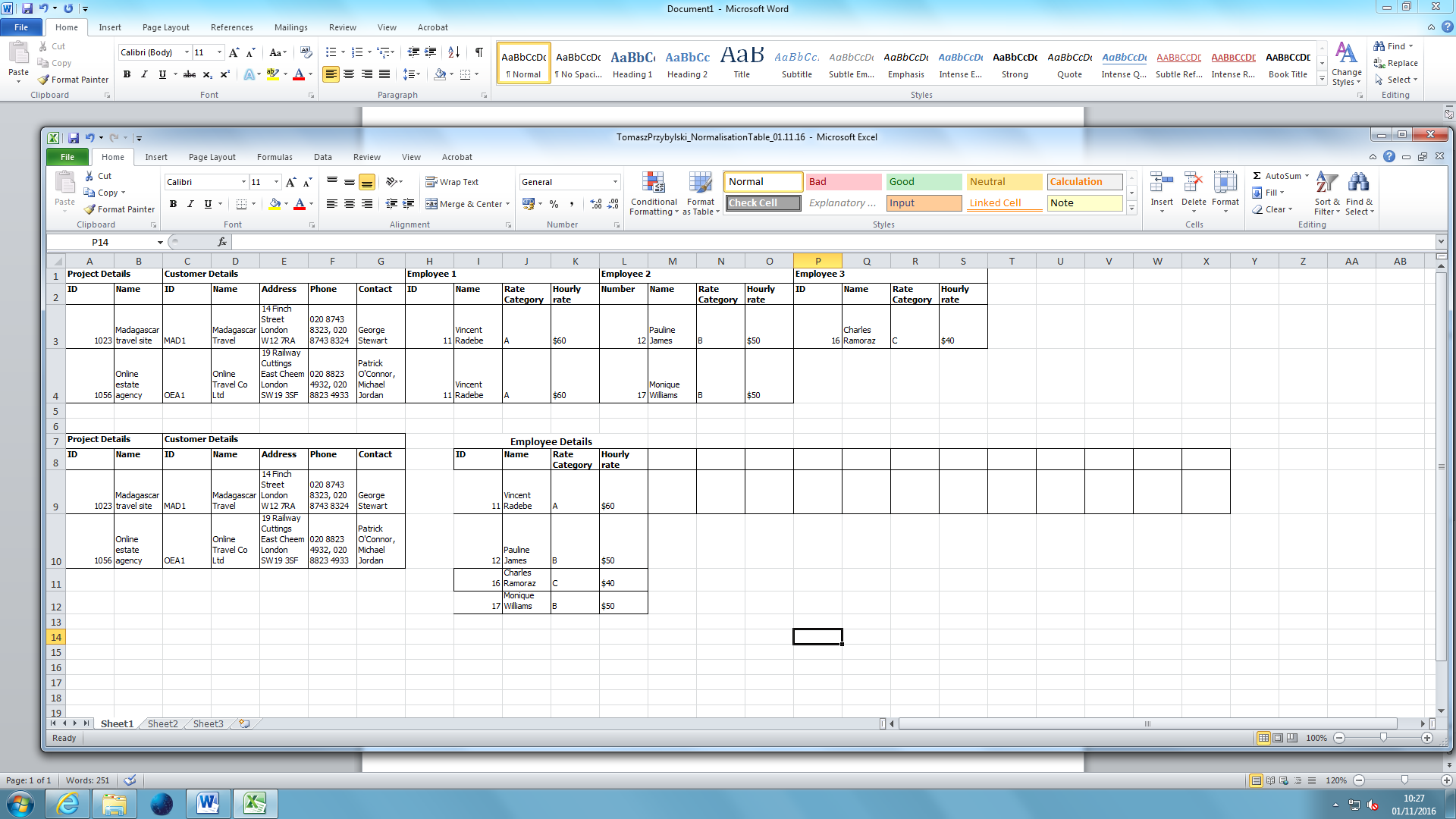
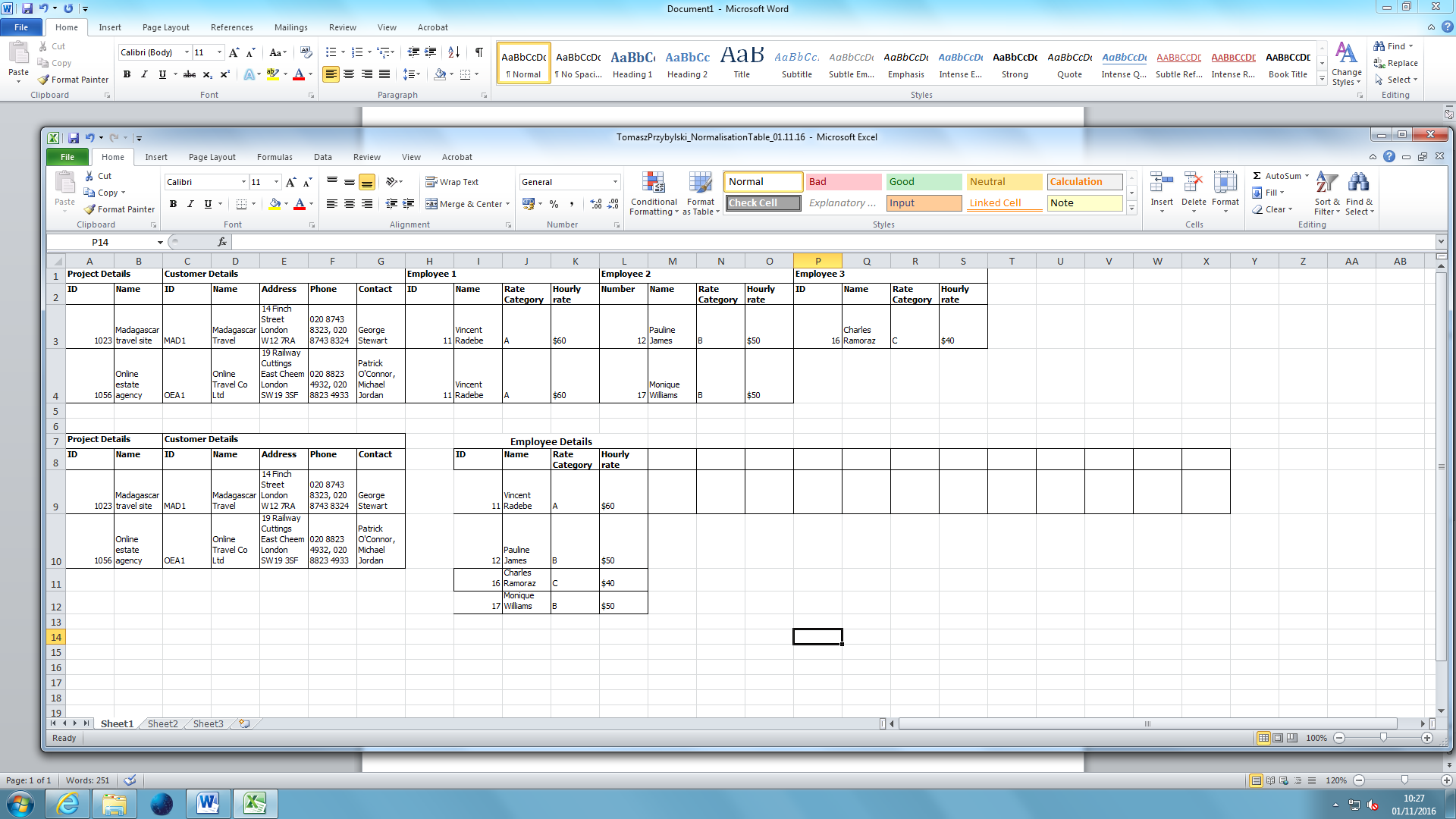
Report on Database Normalisation & Normal Forms

Normalisation is a tool by which databases and data sets are reduced in size and are made easier to manipulate by removing duplicate or data redundancies and grouping entities together. In this case an entity is anything that has a quantifiable attribute or characteristic that would be interesting to a user. Here is an example of a 0NF database:



Here, data such as the Employee ID is repeating. While this database only contains two projects and therefore won’t take up much space, a 0NF database containing hundreds or thousands of projects will take up an incredible amount of space compared to a database in 3rd Normal Form. It would also be near impossible to change existing data, for example is a major customer changes contact details or address.

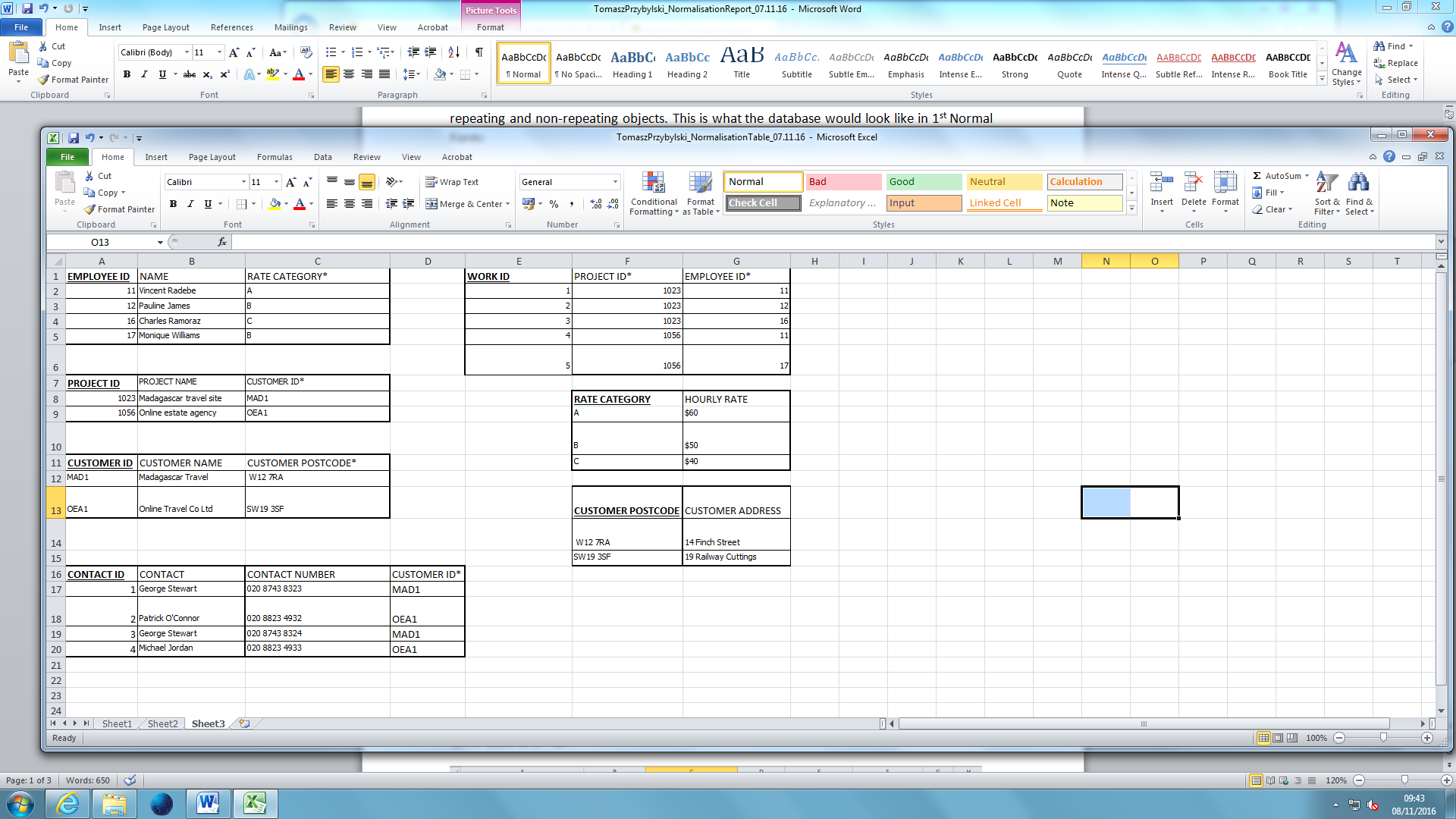
The answer to this is to normalise the database, primarily by putting it into 1st Normal Form, as this should reduce the size of the database and make it much easier to manipulate and handle. 1st Normal Form is achieved by atomising the data and by splitting up and isolating repeating and non-repeating objects. This is what the database would look like in 1st Normal Form:



In this form, the data does not repeat anymore but there are still unnecessary bits of information so the answer is to normalise the database into 2nd and 3rd Normal Form.

2nd Normal Form would create a compound key that could be used to access two or more different data tables. For example, a compound key of the project ID and Employee ID could be made, however in some cases 2nd Normal Form is unnecessary. I created a new “WORK ID” which stops repetition of Employee IDs and means that it is possible to alter employees working on a project without much effort.

3rd Normal Form means that there is no data depending on a non-primary key. All data is corresponding to a unique identifier such as the Customer ID or Employee ID. This is the same table in 3rd NF:



As you can see, this database would have many more tables than the rest, and while that seems like it would mean a larger database, this is actually a much more efficient way of storing data than the other two tables. This is because the data does not repeat and everything is dependent on a primary key.

Every entity is split up and has its own primary key. The **underlined bold headings** are primary keys while the headings with stars (\*) are foreign keys. A user with access to a project ID can find out all other data contained, from full details of the Customer requesting the project to all data on all employees working that project.

This table is definitely in 3rd NF because the only repetition present is from primary or foreign keys.

The “Employee ID” table would only have the “Rate Category” repeat but this is the most efficient way of implementing this data.

The “Project ID” table may have “Customer ID” repeat but this is understandable as a customer may request multiple projects. The “Project ID” and “Project Name” will remain unique.

The “Work ID” table will have repetitions of “Project ID” and “Employee ID” but that is because they are foreign keys. An employee can work on multiple projects and a project may have multiple employees working on it. However “Work ID” will always be unique to one employee working on one project.

The “Customer ID” table will not have any repetitions as it is a table of unique details such as name and postcode.

The “Contact ID” table may have repetitions of the “Customer ID” as many contacts may be assigned to a single customer. It may also have repetitions of “Contact” as the same person may have different contact numbers and it may have repetitions of “Contact Numbers” as one number may correspond to a number of different potential contacts. However this is the most efficient way to categorise this data as it enables quick alteration.