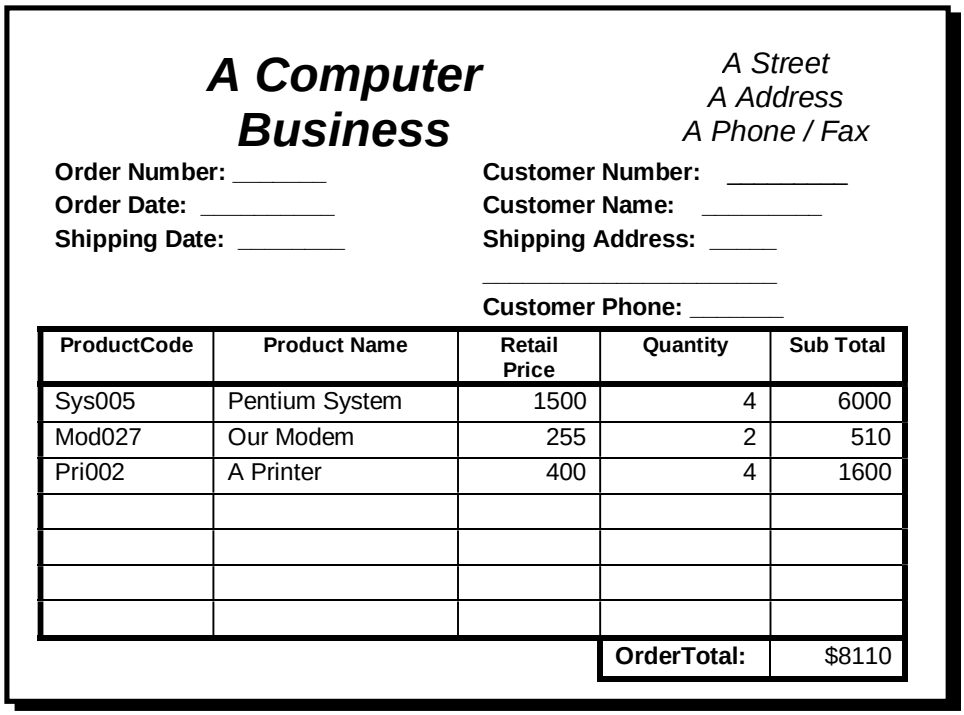
# Normalisation



***A Computer Business***

*A Street*
*A Address*
*A Phone / Fax*

**Order Number:** _____
**Order Date:** _____
**Shipping Date:** _____

**Customer Number:** _____
**Customer Name:** _____
**Shipping Address:** _____
_____
**Customer Phone:** _____

| ProductCode | Product Name | Retail Price | Quantity | Sub Total |
|---|---|---|---|---|
| Sys005 | Pentium System | 1500 | 4 | 6000 |
| Mod027 | Our Modem | 255 | 2 | 510 |
| Pri002 | A Printer | 400 | 4 | 1600 |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | **OrderTotal:** | $8110 |

## *Some Background on Normalisation*

Normalisation is a process that complements and reinforces the process of Entity-Relationship diagramming. It is possible to fully identify a set of entities and their relationships and consequently a set of database tables using Entity-Relationship alone. Nevertheless, Normalisation can allow you to supplement or validate these by starting with data from existing forms or reports and breaking them down into individual relations.

Normalisation can also naturally follow from the data elements identified in the Contents column of the data flow and/or data stores information in a data dictionary.

As an example, let us say that we have a copy of a simple order from a business that sells PC's. In the Data Flows section of their data dictionary let us assume we identified the content of the order form to be:

{ **OrderNumber**, OrderDate, ShippingDate, CustomerNumber, CustomerName, Shipping Address, Phone,
{ ProductCode, ProductName, RetailPrice, Quantity, Subtotal }, Ordertotal }

The process of Normalisation would take the element content of this form and break it down systematically into relations. This process is known to have a number of steps although only the first 3 receive significant attention in general Systems Analysis texts. The output of each step is known as a ***normal*** form. For example:

### Original Un-normalised Data Elements

*Note: Identifying Attributes have been identified with an underline within the relations below..*

{ **OrderNumber**, OrderDate, ShippingDate, CustomerNumber, CustomerName, Shipping Address, Phone, { **ProductCode**, ProductName, RetailPrice, Quantity, Subtotal }, OrderTotal }

The Subtotal and Total elements could be removed in the initial stages, realising that these would likely be calculated fields in the resulting database application and would be displayed when data was presented in forms and reports.

{ **OrderNumber**, OrderDate, ShippingDate, CustomerNumber, CustomerName, Shipping Address, Phone, { **ProductCode**, ProductName, RetailPrice, Quantity } }

### First Normal Form

In this step, the Inner Repeating Groups are removed. The repeating groups are identified by parentheses - { … }. This step aims to remove or separate the inner repeating group(s) from the outer repeating group. In the process we need to add the Identifying Attribute from the outer group to the inner group so as to retain the link or relationship between the two resulting relations. The Identifying Attribute for the new relation may become a combination of the identifier from the outer group and the identifier for the inner group.

{ **OrderNumber**, OrderDate, ShippingDate, CustomerNumber, CustomerName, ShippingAddress, Phone }

{ **OrderNumber**, **ProductCode**, ProductName, RetailPrice, Quantity },

### Second Normal Form

In this step, the Partial Dependencies are removed. This step is only applicable to relations that have a concatenated Identifying Attribute set – ie there are two or more attributes that make up the 'Identifying Attribute'.

It would be a fair statement to say that in this example an element such as ProductName would be better represented by the ProductCode identifier on its own, rather than by the combination of OrderNumber and ProductCode. IE: the ProductName should be in a relation separate from the OrderNumber-ProductCode relation. Thus we need to draw out a new relation from the OrderNumber-ProductCode relation above.

In doing so we need to copy the ProductCode to the new relation so as to maintain the link or relationship between the two relations.

{ **OrderNumber**, OrderDate, ShippingDate, CustomerNumber, CustomerName, ShippingAddress, Phone }

{ **OrderNumber**, **ProductCode**, RetailPrice, Quantity },

{ **ProductCode**, ProductName }

*NOTE: we could debate whether the RetailPrice goes to the later relation or stays in the original.*

## Third Normal Form

In this step, the Transitive Dependencies are removed. This step is applicable to any section of the current relations where there are attributes that don't link directly to the current Identifying Attribute(s) IE: these attributes are not best represented by the current Identifying Attribute.

Again going back to our example, it can be seen that the Customer Details in the first relation are not best represented by the Identifying Attribute - OrderNumber. The Customer details are more appropriately represented by the CustomerNumber, which is not an Identifying Attribute in the first relation. This type of dependency (on a non-key attribute) is often referred to as a transitive dependency. To remove this dependency, the customer information can be removed to its own relation.

To maintain the link or relationship, a copy of the CustomerNumber needs to be left in the OrderNumber relation.

These relations might be named so follows:

**CUSTOMERS**: { **CustomerNumber**, CustomerName, ShippingAddress, Phone }

**ORDERS**: { **OrderNumber**, OrderDate, ShippingDate, CustomerNumber }

**ORDERLINES**: { **OrderNumber**, **ProductCode**, RetailPrice, Quantity },

**PRODUCTS**: { **ProductCode**, ProductName }

*NOTE:* *Something that is embedded in this set of relations is that a given product can only be added as a line item once for any given order. This is not a problem as such. If a customer wants two of a single product, then two must be entered as the Quantity. Where relations such as OrderLines have two attributes that combine as the identifying attribute, an alternate Identifying Attribute could be added such as OrderLinesID.*