# AS91902 Use complex techniques to develop a database

## Brief

You are required to design a multi-table database for a supermarket who want to keep track of both their stock and their customers and the customers' orders.

For the stock the data required will be barcode number, description, manufacturer id, manufacturer name and address, retail and wholesale price or the item, the amount in stock, reorder level, reorder amount and amount sold. The final item will be a Boolean field that will have a default value of False but becomes True when new stock has been reordered and goes back to False again once the stock has been delivered. Data repetition must be avoided.

The organisation also needs data on its customers. The data here will be customer id and their name and address. A customer can place as many orders as they wish. Each order will be given a unique id number as well as the date of the order. It will also have an Boolean field which is set to true once the order has been printed. This is to avoid the same order being printed more than once. The order will also have multiple stock items. For each stock item on the order the amount sold is recorded and a Boolean field is set to False to indicate that the stock table has not been updated. Later an update query will be used to update toe stock table. This query will reduce the amount in stock for a particular item by the amount of that item that has been ordered.

## Data Design

The above data must be normalised, indicating both key fields and foreign keys, and from this design database tables must be created. The appropriate fields must be set as key fields and foreign keys must comply with the appropriate key fields. Data integrity rules must be applied to the appropriate fields.

As well as the tables above an extra table must be provided. This will be linked to a spreadsheet and is to be named StockDelivery. It will contain only two fields, barcode number and the amount of incoming stock. Its purpose is to emulate data from suppliers indicating that new stock has been delivered. An update query will be used to extract data from this table in order to update the stock table.

## Forms

Data is not to be entered directly into any of the tables; all manual data entry for the manufacturer, stock and customers must be done through appropriate forms. Forms should also be provided for browsing, deleting or modifying data in the same tables. Care should be taken that each form performs only  its allocated task and that data relating to inter-table relationships is not modified.

For filling in customer orders the form-subform system should be used.

## Queries

### Stock Reorder Query

In order to maintain a reasonable level of stock within the supermarket, new stock must be periodically ordered. The stock table has three related fields, one for the amount in stock one for the other the reorder level and the third for the reorder amount, i.e. once we reorder new stock, how much do we reorder.

Once the amount in stock for a particular item falls below the reorder level, this will be a signal for reordering new stock.  This query will display the values for the manufacturer name, barcode, description, wholesale price and reorder amount, as well as a calculated field which will be the product of the wholesale price and reorder amount and the Boolean field to indicate that the item has been reordered is False.  The reason for including the last field is to ensure that the same item is not reordered twice.

This will be printed for each record where the amount in stock is less than the reorder level and the

### Query for printing Receipts

The purpose of this query is to display data equivalent to that from the supermarket till once you have paid for your purchases. This however has somewhat more detail in it.

The query will display the values for the customer id and name, the order number and the date and the barcode, description, sale amount and a calculated field which will be the product of sale amount and retail price.  This will apply to all records where the Boolean field for order being printed has a value of False.

Below we shall discuss how this query's output will be displayed as a report.

### Flag Stock being reordered Query

This update query is run immediately after the Stock Reorder Query above. Both queries work on the same criteria, i.e. that the amount in stock is less than the reorder level and that the Boolean field indicating that the stock has been ordered is False.  If this is the case then the field reordered will be set to True, which will prevent the stock from being reordered twice.

Once the stock has been delivered the next query below sets the same Reordered field back to False.

### Process incoming Stock Query

This is a complimentary query to the stock reorder query above. The data for the incoming stock goes into the spreadsheet file which is linked to the table StockDelivery. This query uses that table to update the new stock levels in the stock table.

The query adds the value of the amount delivered field to the amount in stock field of the stock table. It also sets the Boolean field for reordered to False This occurs only if the barcode field of the StockDelivery table is equal to the barcode field of the stock table.

### Query for adjusting stock levels after a customer's sale has been processed

This query runs once the user has completed adding all of the stock items to fill a customer's order. It scans the order table for orders where the Boolean field indicating that the stock file has been updated is False. If this is the case then it subtracts the value of the sale amount from the amount in stock field and adds the value of the sale amount to the amount sold field. It also sets the Boolean field indicating that the stock file has been updated to True.

### Order Printed Query

This query is run immediately after recent orders have been printed. The records whose contents were printed had been selected on the basis that the field OrderPrinted had a value of False, as well as the amount in stock being lower than the reorder level. In order to ensure that the same orders will not be printed twice the OrderPrinted field is set to True. The criteria for this query is the same as for the stock reorder query.

## Reports

### Stock Reorder Report

Create a report based on the stock reorder query above. All fields of the query are to be shown apart from the field Reordered and the report is to be grouped on the manufacturer name. Appropriate subtotalling and formatting is to be applied.

### Customer Receipt Report

This report is based on the Query for printed receipts above. The report must have appropriate headings and footers, appropriate grouping of the orders and appropriate totals and subtotals. Again appropriate formatting must be applied.

## Macros

### Customer Order Macro

To process a customer's purchases the following actions must take place:

1. The customer order form is opened to allow the user to enter sales details.

2. Once all of the sales details are entered the adjust stock after sale query is run in order to reduce the amount in stock values for the items that have been sold
3. The report for printing customer receipts is then created
4. Finally the Order Printed query is run

Create a macro that will automatically action each of the above activities.

### Stock Reorder Report Macro

This macro will load the stock reorder report which allows the user to inspect the reorder.

## A front-end Main Menu

For ease of access to the database's functionality you must create an extra form that guides to user to the application's features. For the purpose of this form you may presume that all of the database's tables have been created and populated with appropriate data. You may also presume that queries and macros have already been created.

The controls on the form should allow the user to access the database's facilities or adding, browsing, deleting and modifying records in the customer, manufacturer and stock tables. It will also have controls for running the two macros specified above.

## General

Forms and reports should, within reason, comply with accepted usability heuristics and be accessible to both the physically and visually impaired.

Unless this application is being designed for an actual client, names of real organisations or products should not be used.

## Marking scheme

| | | |
|---|---|---|
| Achievement | A  normalised design has been produced | |
| | Tables have been designed according to specifications | |
| | Key fields and foreign keys used to ensure data integrity | |
| | Forms' functionality confined to what is specified for each form | |
| | Documentation showing incorrect data being rejected | |
| | Only required data shown in select queries | |
| | Documentation provided showing that update queries have worked correctly | |
| | Appropriate data appears in reports | |
| | No real life names of organisations or products are used | |
| | Basic usability heuristics applied | |
| | Documentation indicating possible improvements on most parts of the project | |
| Merit | Suggestions in the above document have been applied | |
| | Users prevented from unnecessary access to data in forms | |
| | Forms' presentation accommodates the visually and physically impaired | |
| | Reports have appropriate header and footers and subtotals | |
| | Macros used to ensure correct sequence of data processing | |
| | Documentation indicating possible improvements on most parts of the project | |
| Excellence | Suggestions in the above document have been implemented | |
| | Front end menu used to access internal functionalities | |
| | Simultaneous processing prevented | |
| | Report on how relevant implications have been addressed throughout the project | |
| | | |