My Xmas Shop

1. Introduction

The developed application is used to management an online Christmas Shop which sells Christmas supplies including Christmas tree, stockings, etc. The app consists of a database, and an Excel frontend with VBA middleware.

2. Database

An Access database named MyXmasShop.accdb is created to store customer, order, and product data, with the following tables and queries.

Tables

Customers		
CustomerID	Short Text	Alphanumeric key uniquely identifying a customer (Primary Key)
CustomerName	Short Text	Name of the customer
DOB	Date/Time	Date of birth of the customer
ShippingAddress	Long Text	Shipping address of the customer

Orders		
OrderID	Short Text	Alphanumeric key uniquely identifying an order (Primary Key)
CustomerID	Short Text	Foreign key for the customer that placed this order.
OrderDate	Date/Time	Date the order was made
ShipDate	Date/Time	Date the order was shipped
OrderStatus	Short Text	Order status of the order (i.e. processing, shipped, delivered)

Products		
ProductID	Short Text	Alphanumeric key uniquely identifying a product (Primary Key)
ProductName	Short Text	Name of the product
Category	Short Text	Name of the category to which the product belongs
UnitPrice	Currency	Price per unit in £GBP

OrdersProducts		
OrderID	Short Text	Foreign key identifying an order in which the product was included.
ProductID	Short Text	Foreign key identifying a product.
Quantity	Integer	How many units of the product were included in this order.

Queries

Top 3 Best-selling Products	
Description	List the top 3 products with the highest total quantity sold across the entire time period recorded in the database, so that staff know which products to stock more.

```
SELECT TOP 3
   Products.ProductName,
   Products.Category,
   SUM(OrdersProducts.Quantity) AS TotalQuantity,
   SUM(OrdersProducts.Quantity * Products.UnitPrice) AS TotalSales
FROM OrdersProducts
INNER JOIN Products
    ON OrdersProducts.ProductID = Products.ProductID
GROUP BY Products.ProductName, Products.Category
ORDER BY SUM(OrdersProducts.Quantity) DESC;
```

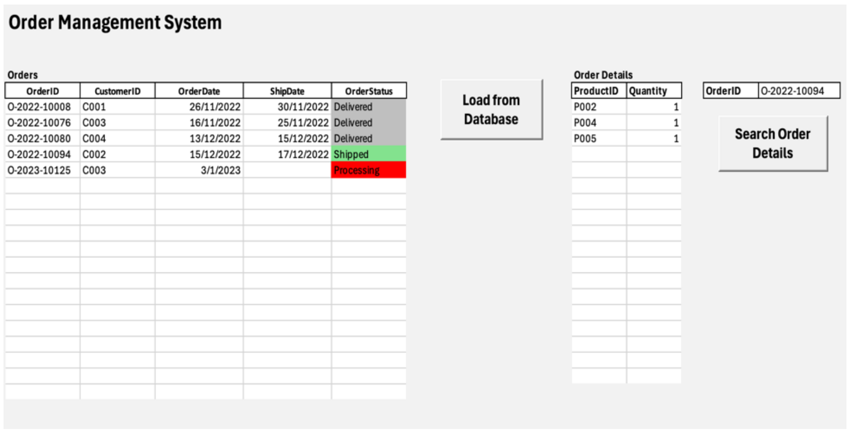
Unfulfilled Orders List	
Description	List the details of orders that are unfulfilled (i.e. processing), so that staff can follow up with these orders as soon as possible.

```
SELECT
   Orders.OrderID,
   Orders.CustomerID,
   Customers.CustomerName,
   Customers.ShippingAddress,
   Orders.OrderDate,
   Orders.ShipDate,
   Orders.OrderStatus
FROM Orders
INNER JOIN Customers
   ON Orders.CustomerID = Customers.CustomerID
WHERE Orders.OrderStatus = "Processing";
```

Customer Orders Search	
Description	Take a single parameter named [Given Customer Name], then list the details of orders of that customer (i.e. order search by customer name)

```
PARAMETERS [Given Customer Name] CHAR;
SELECT
    OrdersProducts.OrderID,
    Products.ProductName,
    OrdersProducts.Quantity,
    OrdersProducts.Quantity*Products.UnitPrice AS Total
FROM ((Products
  INNER JOIN OrdersProducts
     ON Products.ProductID = OrdersProducts.ProductID)
INNER JOIN Orders
     ON Orders.OrderID = OrdersProducts.OrderID)
INNER JOIN Customers
     ON Orders.CustomerID = Customers.CustomerID
WHERE Customers.CustomerName = [Given Customer Name];
```

3. Front-end


Order Management System

Orders

OrderID	CustomerID	OrderDate	ShipDate	OrderStatus
O-2022-10008	C001	26/11/2022	30/11/2022	Delivered
O-2022-10076	C003	16/11/2022	25/11/2022	Delivered
O-2022-10080	C004	13/12/2022	15/12/2022	Delivered
O-2022-10094	C002	15/12/2022	17/12/2022	Shipped
O-2023-10125	C003	3/1/2023		Processing

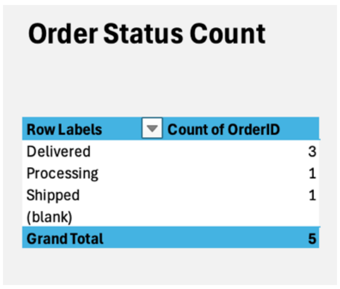
Order Details

ProductID	Quantity
P002	1
P004	1
P005	1

OrderID	O-2022-10094

OrderManagement Sheet

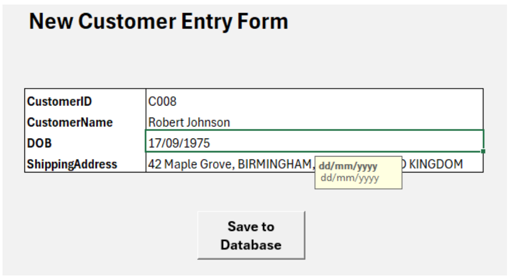
The Excel file MyXmasShop.xlsm contains three sheets that serve as a front-end for staff to check and update records in the database. The 'OrderManagement' sheet displays all records from the Orders table, with the *OrderStatus* colour-coded based on different statuses (i.e., red for processing, green for shipped, grey for delivered). The records are populated by clicking the 'Load from Database' button. The 'OrderManagement' sheet also includes a search function that allows staff to input a specific *OrderID* to view the products and quantities ordered for that *OrderID*. This search function is activated by clicking the "Search Order Details" button.


Order Status Count

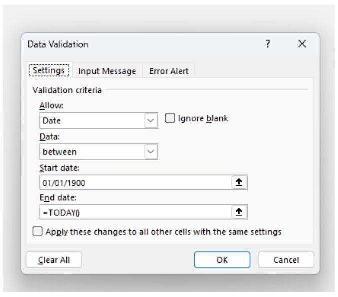
Row Labels	Count of OrderID
Delivered	3
Processing	1
Shipped	1
(blank)	
Grand Total	5

OrderStatusCount Sheet

The second tab, 'OrderStatusCount,' contains a pivot table that shows the count for each *OrderStatus*. This table enables staff to keep track of orders and determine how many are not processed and require follow-up.


New Customer Entry Form

CustomerID	C008
CustomerName	Robert Johnson
DOB	17/09/1975
ShippingAddress	42 Maple Grove, BIRMINGHAM,) KINGDOM


Data Validation

Settings | Input Message | Error Alert

Validation criteria

Allow: Date

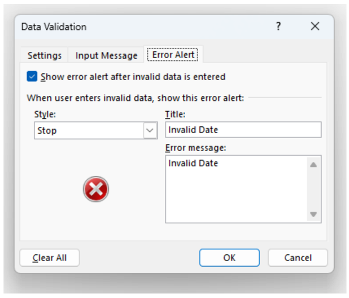
[ ] Ignore blank

Data: between

Start date: 01/01/1900

End date: =TODAY()

[ ] Apply these changes to all other cells with the same settings


Data Validation

Settings | Input Message | Error Alert

[✓] Show error alert after invalid data is entered

When user enters invalid data, show this error alert:

Style: Stop

Title: Invalid Date

Error message: Invalid Date

NewCustomerForm Sheet

The third tab, 'NewCustomerForm', contains a form where staff can input new customer information and click the "Save to Database" button to save the data to the database. Data validation is applied to the *DOB*. An input message of "dd/mm/yyyy" reminds staff of the required date format. An error message is also displayed if the date falls within an invalid range. For instance, the data validation rule ensures that the *DOB* is between 01/01/1900 and TODAY().

4. VBA Middleware

The following VBA middleware is used to perform the actions mentioned in the Excel file.

CustomerDataType	A user-defined type that stores customer-related data, including CustomerID, CustomerName, DOB, and ShippingAddress.
GetCustomerData(CustomerData)	A subroutine that retrieves customer data from the 'NewCustomerForm' sheet and stores it in a CustomerDataType variable.
Open_Database_Connection()	A public function that establishes and returns a connection to the Access database file. It dynamically builds the database file path by combining the folder path of the current workbook with the database filename.
LoadFromDatabase()	A subroutine that loads data from the 'Orders' table of the database into the Excel sheet, using Open_Database_Connection().
SearchOrderDetails()	A subroutine that searches for order details by OrderID input by users in the 'OrderManagement' sheet. It retrieves the relevant product and quantity information from the database and displays it in Excel.
SaveToDatabase()	A subroutine that saves new customer data to the database. It checks if the customer already exists and inserts new customer information if not. If successful, a message box will pop up.

5. Conclusion

The source file for the above application can be found at: https://github.com/chyuenn/MyXmasShop

To scale up this app for a real business, several improvements can be made. First, the database tables could include additional information, such as product dimensions, shipping methods, and payment details. This would allow for enhanced data analytics to improve sales and customer satisfaction. Second, the Excel file could be expanded with more functions for staff to perform on the front end, such as inputting new orders or modifying customer or order information. Third, further automation like generating reports and sending email updates to customers can also be developed using VBA subroutines.