# Project Proposal: Online Reseller Management Application

## Overview

A Microsoft Access database application designed to track basic online sales information and inventory for an online electronics reseller that uses multiple platforms such as Amazon, e-Bay, etc. to deliver goods to customers from a home-based business.

## Context

The application will be provided free of charge from the wholesale distributor to its reseller who opt into a special program.

### Participants

Online resellers

### Purpose

The application will provide a means of importing and exporting inventory data; and generate various management and tracking reports to make the resellers more efficient including:

* Customer and Vendor lists
* Vendor and Product indexes
* Monthly Sales report
* Inventory tracking, verification, and data import/export

## Resources

A single software developer will complete the project using standard Microsoft Office and Visual Studio tools.

### Timeframe Estimate

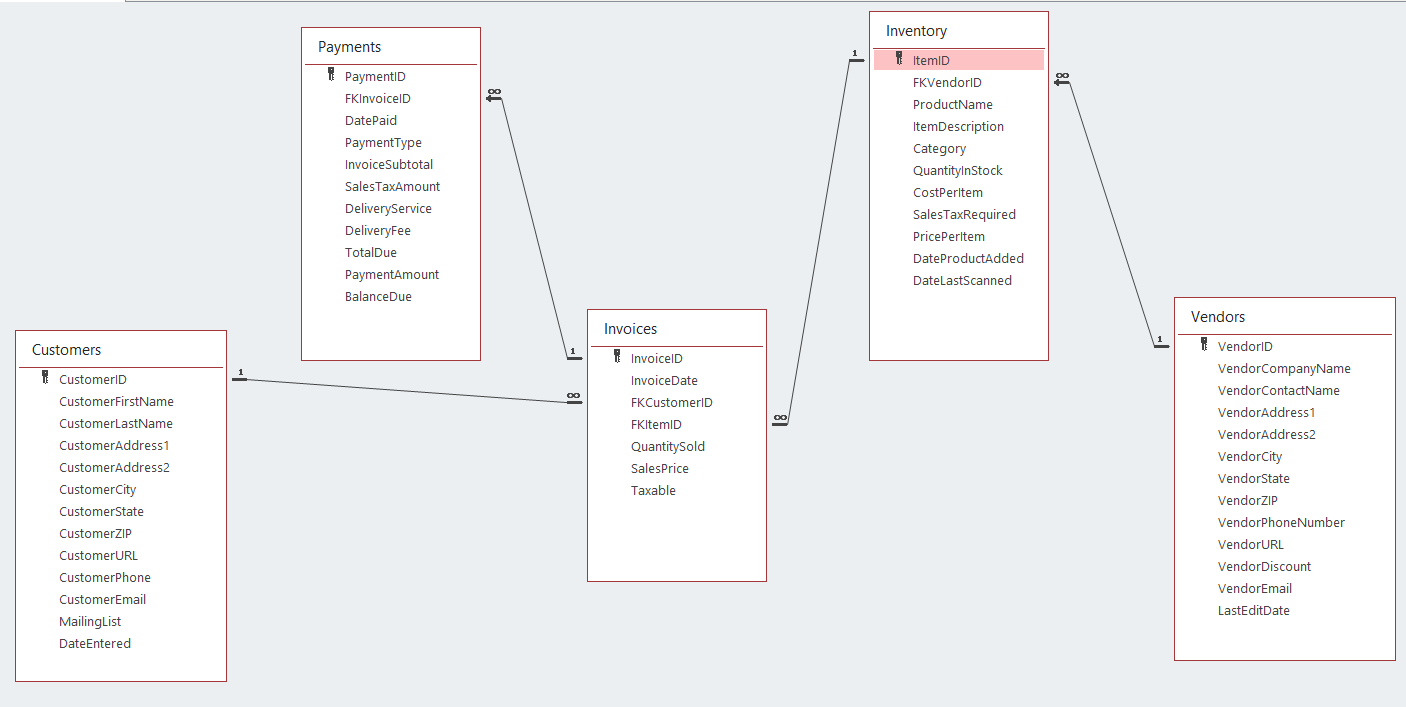
|  |  |  |
| --- | --- | --- |
|  | Task | Start and End Dates (Task estimate) |
| **Design** |  | **April 4-22, 2022** |
|  | Database & Queries | (Approx. 5 days) |
|  | Forms | (Approx. 2.5 day) |
|  | Reports | (Approx. 2.5 days) |
| **Development** |  | **April 22-25, 2022** |
|  | Tools & Other components | (Approx 3 days) |

## Project Overview

This simple desktop application is designed to track basic retail sales information for a home-based business setup to be an online electronics reseller. The underlying data model, sample screens, report definitions, and other functionality are defined below.

### Data Model

Two contact information tables were included for Customer and Vendor entities, along with an Inventory table to track which items are for sale, and a set of parent-child tables for recording Sales invoices and Payments.



### Application Screens

The primary application interface will be a tabbed window setup to serve as dashboard that surfaces all the functionality in an easily navigated package as illustrated below:

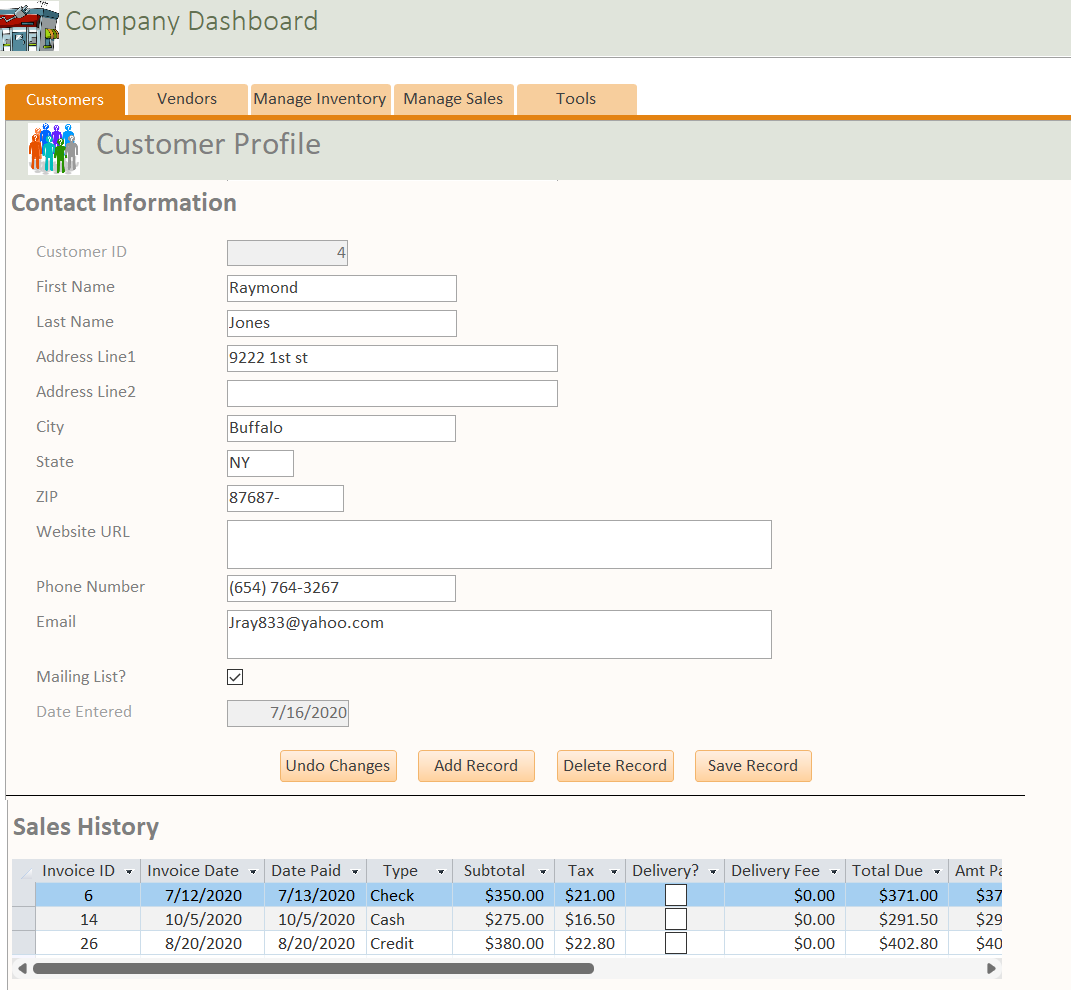


Figure 1 Company Dashboard Customers Tab

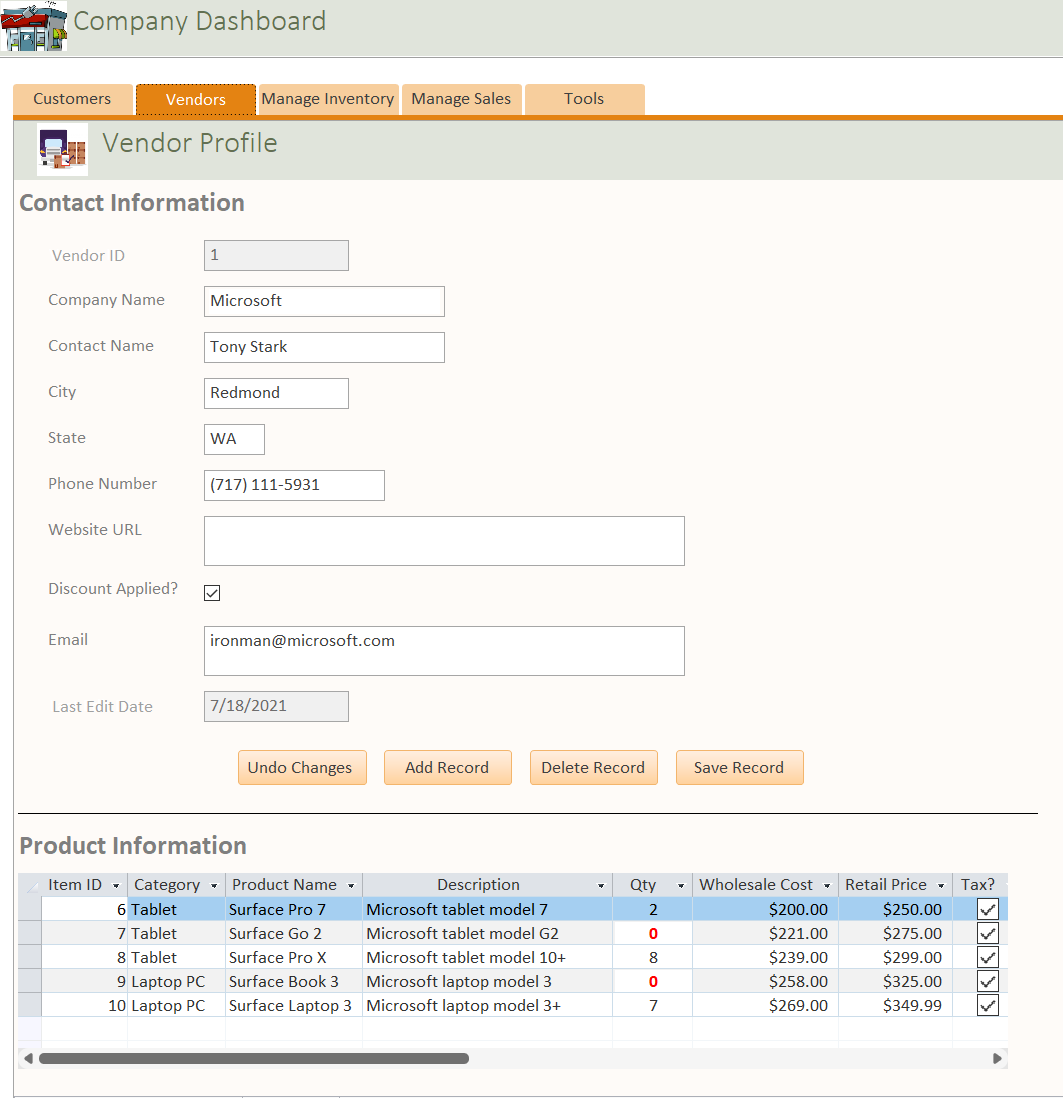


Figure 2 Company Dashboard Vendors Tab

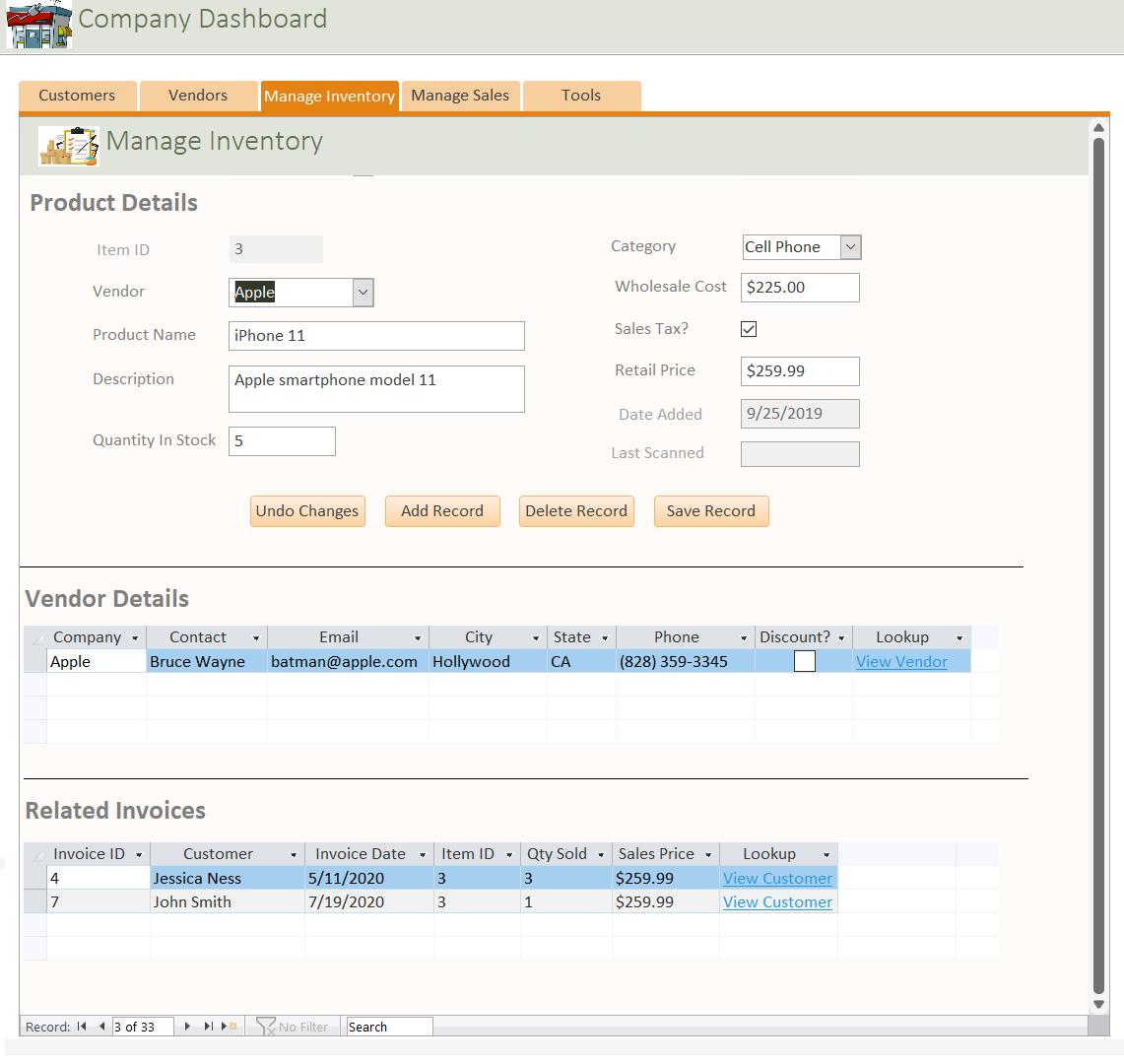


Figure 3 Company Dashboard Manage Inventory Tab

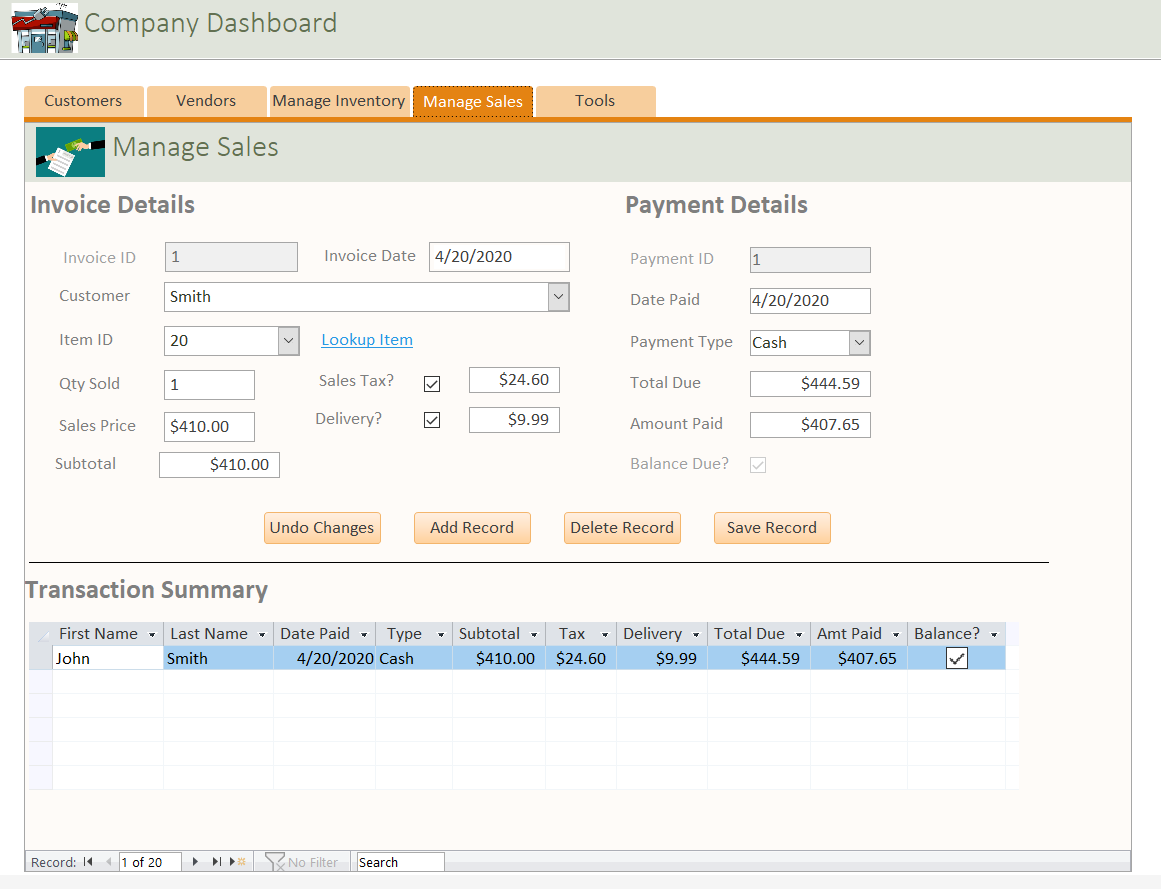


Figure 4 Company Dashboard Manage Sales Tab

A secondary application interface will be another tabbed window embedded within the Company Dashboard Tools tab, which surfaces a combination of reports, queries, and advanced functionality as shown below.

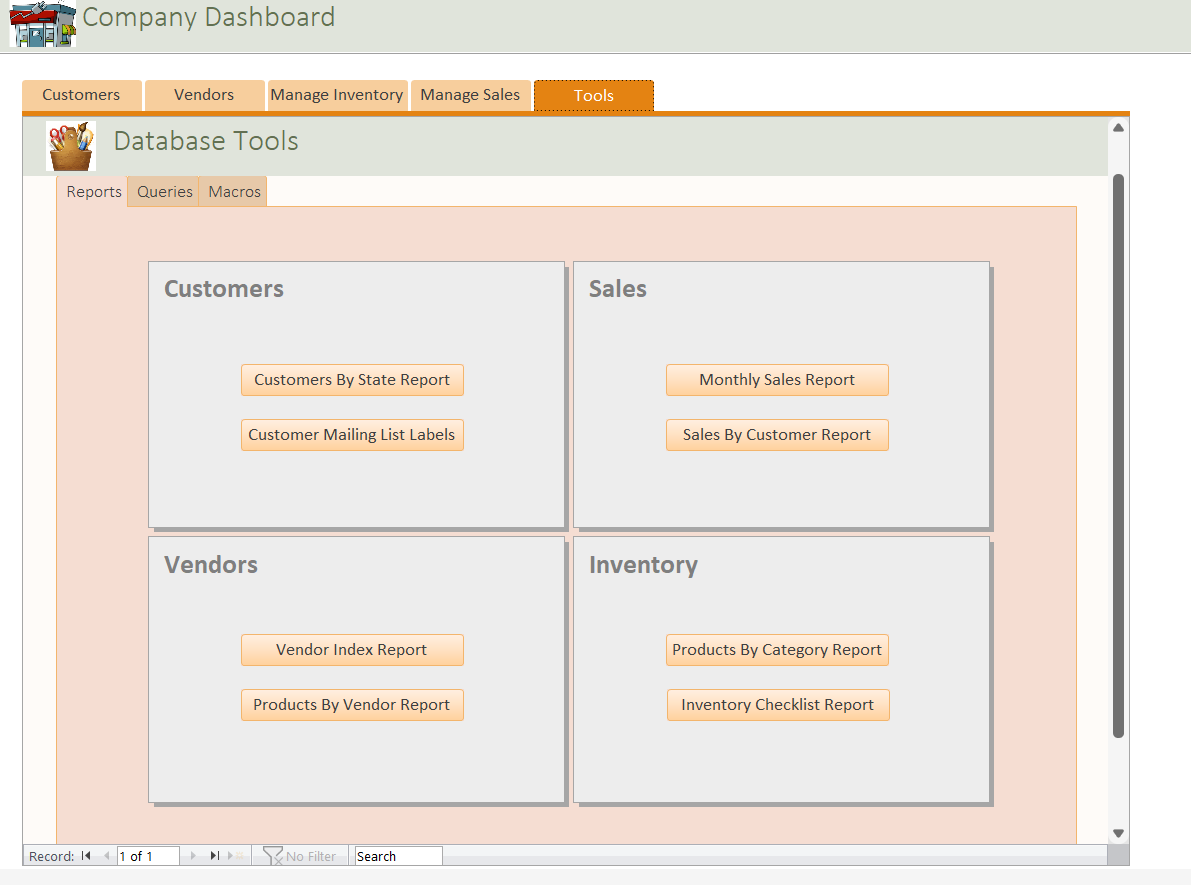


Figure 5 Company Dashboard Tools Tab, Reports Menu

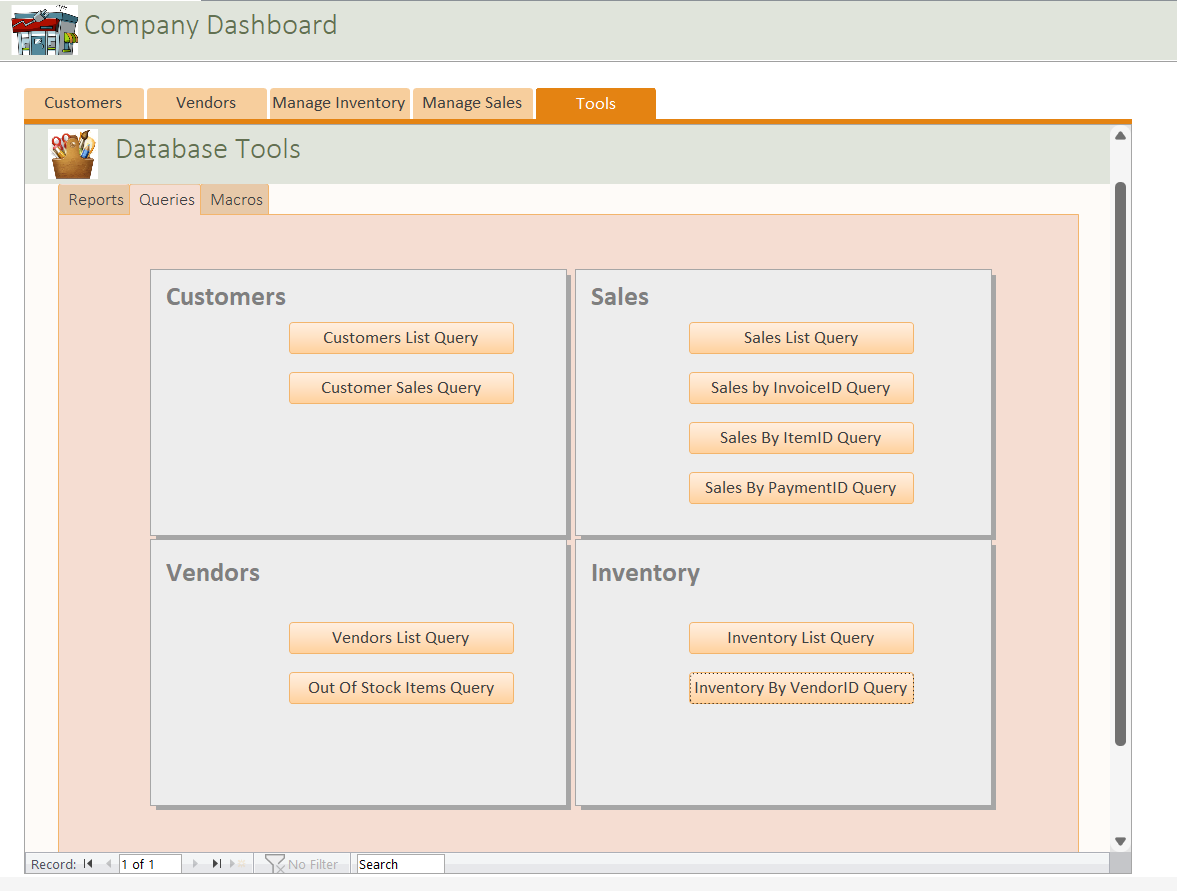


Figure 6 Company Dashboard Tools Tab, Queries Menu

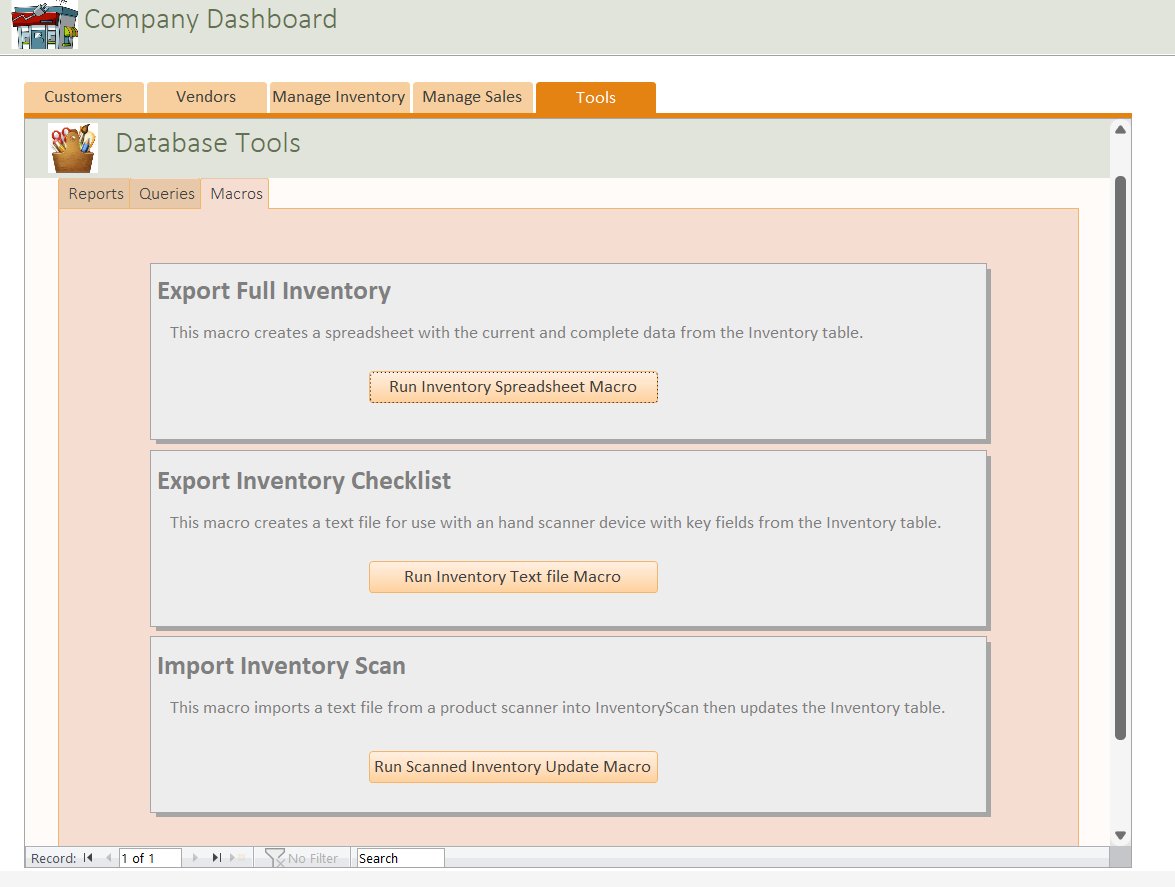


Figure 7 Company Dashboard Tools Tab, Macros Menu

### Application Output

The application will produce output in four forms: standardized reports, mailing labels, onscreen queries, and text files for two data export functions. Text files will be generated for export operations in Microsoft Excel spreadsheet format, or a comma-separated (CSV) format, depending on the functionality being used and is defined further in the next major section of this proposal.

**Report Definitions**

The following list of reports and mailing labels will be generated through a normal Windows Print dialog, so users will have various options for both hard copy and digital output.

|  |  |
| --- | --- |
| *Title* | *Description* |
| Customers By State Report | A full index of Customer contact information grouped by state. Uses a row-based format, landscape page orientation, and includes the mailing list indicator flag. The data source will be the Customers table. |
| Customer Mailing List Labels | A partial listing of customers who have opted into the seller’s mailing list program. Includes street address information only in a 3-column format for use with standard Avery 5160/8160 adhesive label stock. The data source will be an embedded query. |
| Monthly Sales Report | A detailed transaction ledger, summarized by month, which provides customer, invoice, and payment information for tracking sales. Uses a row-based format, and landscape page orientation. The data source will be an external query that can also be accessed separately. |
| Sales By Customer Report | A detailed transaction listing, summarized by customer, which provides invoice and payment history. Uses a row-based format, and landscape page orientation. The data source will be an external query that can also be accessed separately. |
| Vendor Index Report | A full index of Vendor contact information. Uses a row-based format, and landscape page orientation. The data source will be an external query that can also be accessed separately. |
| Products By Vendor Report | A detailed product inventory listing, summarized by vendor, which provides vendor, product, and pricing information. Uses a row-based format, and landscape page orientation. The data source will be an external query that can also be accessed separately. |
| Products By Category Report | A detailed product inventory listing, summarized by product category, which provides vendor, product, and pricing information. Uses a row-based format, and landscape page orientation. The data source will be an external query that can also be accessed separately. |
| Inventory Checklist Report | A product listing, summarized by vendor, which provides an inventory checklist for verifying the expected quantity in stock for each item on-hand versus the actual quantity in stock. Uses a row-based format, and portrait page orientation. The data source will be an external query that can also be an embedded query. |

**Query Definitions**

The following list of Queries will be setup independently of the various screens, reports and other functionality that uses some of them as a data source, in order to provide an additional means of viewing and listing data.

|  |  |
| --- | --- |
| *Name* | *Syntax* |
| Customers List Query | SELECT Customers.CustomerID, Customers.CustomerFirstName, Customers.CustomerLastName, Customers.CustomerAddress1, Customers.CustomerAddress2, Customers.CustomerCity, Customers.CustomerState, Customers.CustomerZIP, Customers.CustomerURL, Customers.CustomerPhone, Customers.CustomerEmail, Customers.MailingList, Customers.DateEntered  FROM Customers; |
| Customer Sales Query | SELECT Customers.CustomerID, Customers.CustomerFirstName, Customers.CustomerLastName, Customers.CustomerCity, Customers.CustomerState, Customers.CustomerPhone, Invoices.InvoiceID, Invoices.InvoiceDate, Payments.PaymentID, Payments.DatePaid, Payments.PaymentType, Payments.InvoiceSubtotal, Payments.SalesTaxAmount, Payments.DeliveryService, Payments.DeliveryFee, Payments.TotalDue, Payments.PaymentAmount, Payments.BalanceDue  FROM (Customers RIGHT JOIN Invoices ON Customers.CustomerID = Invoices.FKCustomerID) LEFT JOIN Payments ON Invoices.InvoiceID = Payments.FKInvoiceID; |
| Sales List Query | SELECT Invoices.InvoiceDate, Invoices.InvoiceID, Customers.CustomerID, Customers.CustomerFirstName, Customers.CustomerLastName, Payments.DatePaid, Payments.PaymentID, Payments.PaymentType, Payments.InvoiceSubtotal, Payments.SalesTaxAmount, Payments.DeliveryFee, Payments.TotalDue, Payments.PaymentAmount, Payments.BalanceDue  FROM (Customers RIGHT JOIN Invoices ON Customers.CustomerID = Invoices.FKCustomerID) LEFT JOIN Payments ON Invoices.InvoiceID = Payments.FKInvoiceID; |
| Sales By InvoiceID Query | SELECT Invoices.InvoiceDate, Invoices.InvoiceID, Customers.CustomerID, Customers.CustomerFirstName, Customers.CustomerLastName, Payments.DatePaid, Payments.PaymentID, Payments.PaymentType, Payments.InvoiceSubtotal, Payments.SalesTaxAmount, Payments.DeliveryFee, Payments.TotalDue, Payments.PaymentAmount, Payments.BalanceDue  FROM (Customers RIGHT JOIN Invoices ON Customers.CustomerID = Invoices.FKCustomerID) LEFT JOIN Payments ON Invoices.InvoiceID = Payments.FKInvoiceID  WHERE (((Invoices.InvoiceID)=[Enter Invoice ID])); |
| Sales By ItemID Query | SELECT Invoices.InvoiceDate, Invoices.InvoiceID, Customers.CustomerID, Customers.CustomerFirstName, Customers.CustomerLastName, Payments.DatePaid, Payments.PaymentID, Payments.PaymentType, Payments.InvoiceSubtotal, Payments.SalesTaxAmount, Payments.DeliveryFee, Payments.TotalDue, Payments.PaymentAmount, Payments.BalanceDue  FROM (Customers RIGHT JOIN Invoices ON Customers.CustomerID = Invoices.FKCustomerID) LEFT JOIN Payments ON Invoices.InvoiceID = Payments.FKInvoiceID  WHERE (((Invoices.FKItemID)=[Enter Item ID])); |
| Sales By PaymentID Query | SELECT Invoices.InvoiceDate, Invoices.InvoiceID, Customers.CustomerID, Customers.CustomerFirstName, Customers.CustomerLastName, Payments.DatePaid, Payments.PaymentID, Payments.PaymentType, Payments.InvoiceSubtotal, Payments.SalesTaxAmount, Payments.DeliveryFee, Payments.TotalDue, Payments.PaymentAmount, Payments.BalanceDue  FROM (Customers RIGHT JOIN Invoices ON Customers.CustomerID = Invoices.FKCustomerID) LEFT JOIN Payments ON Invoices.InvoiceID = Payments.FKInvoiceID  WHERE (((Payments.PaymentID)=[Enter Payment ID])); |
| Vendors List Query | SELECT Vendors.VendorID, Vendors.VendorCompanyName, Vendors.VendorContactName, Vendors.VendorAddress1, Vendors.VendorAddress2, Vendors.VendorCity, Vendors.VendorState, Vendors.VendorZIP, Vendors.VendorPhoneNumber, Vendors.VendorURL, Vendors.VendorDiscount, Vendors.VendorEmail, Vendors.LastEditDate  FROM Vendors; |
| Out Of Stock Items Query | SELECT Inventory.ItemID, Inventory.Category, Vendors.VendorCompanyName, Inventory.ProductName, Inventory.ItemDescription, Inventory.QuantityInStock, Inventory.CostPerItem, Vendors.VendorDiscount, Inventory.PricePerItem, Inventory.SalesTaxRequired, Inventory.DateProductAdded, Inventory.DateLastScanned  FROM Vendors RIGHT JOIN Inventory ON Vendors.VendorID = Inventory.FKVendorID  WHERE (((Inventory.QuantityInStock)=0)); |
| Inventory List Query | SELECT Inventory.ItemID, Vendors.VendorID, Vendors.VendorCompanyName, Inventory.ProductName, Inventory.ItemDescription, Inventory.QuantityInStock, Inventory.CostPerItem, Vendors.VendorDiscount, Inventory.PricePerItem, Inventory.SalesTaxRequired, Inventory.DateProductAdded, Inventory.Category, Inventory.DateLastScanned  FROM Vendors LEFT JOIN Inventory ON Vendors.VendorID = Inventory.FKVendorID  WHERE (((Inventory.Category)<>"")); |
| Inventory By VendorID Query | SELECT Inventory.ItemID, Inventory.Category, Vendors.VendorCompanyName, Inventory.ProductName, Inventory.ItemDescription, Inventory.QuantityInStock, Inventory.CostPerItem, Vendors.VendorDiscount, Inventory.PricePerItem, Inventory.SalesTaxRequired, Inventory.DateProductAdded, Inventory.DateLastScanned  FROM Vendors RIGHT JOIN Inventory ON Vendors.VendorID = Inventory.FKVendorID  WHERE (((Vendors.VendorID)=[Enter VendorID])); |

### Import/Export Functionality

The application will also include three advanced features under the Macros menu of the Tools tab on the Company Dashboard screen. These additional features surface the following import/export functionality.

|  |  |
| --- | --- |
| *Action* | *Description* |
| Export Full Inventory | A macro that generates a full index of inventory items in Microsoft Excel spreadsheet (\*.xls) format. |
| Export Inventory Checklist | A macro that generates a text file with key inventory fields for use with handheld bar code scanner. |
| Import Inventory Scan | A macro that updates the quantity in stock for inventory items processed with a handheld bar code scanner. |