Where-House Inventory Management



Athena Castillo, Danielle Lambion, Calvin Gunther

**Abstract**

Where-House Inventory Management is a warehouse management application designed for use by a warehouse chain. Our application will help warehouse workers automate logistics, by helping keep track of inventory, orders, the status of those orders, and which shipment truck those orders will be shipped out on and from where. Workers will more easily find products within the warehouse to prepare orders, view what stock is low, and which truck the orders will be shipped on.

**Introduction**

While businesses continue to grow, keeping track of inventory becomes more difficult and more important. Inventory management is very important to a company, as loss of goods or an inefficient system to track inventory will negatively impact the company. As such, developing efficient inventory management software can be very beneficial to companies and the market in general. Additionally, creating inventory management software from the ground up for each business can become expensive, so creating versatile inventory software that only needs small changes to adapt to different customers can be beneficial to the industry as a whole.

**Objective/Scope of Project**

The objective of this project is to develop inventory management software to be used for managing the inventory of a warehouse and its shipments, where massive influxes of shipments are expected. We chose to focus on warehouse users as that is where the benefits of efficient inventory management would be most needed. A simpler design would ideally be adaptable to different warehouse companies and easily modifiable to fit the unique needs of each user. This will provide a platform for database users just to put in item information and then be able to reference it whenever needed. This will allow certain information to be dynamically updated as well, such as the stock and sale prices. We originally wanted to add features to allow the input of new products and a search by function, but there is no implementation for that within our web application thus far. We would consider this for future implementations.

**Relation to Existing Work**

There are already many existing warehouse management systems available to companies. An example of one is Logiwa. Logiwa is used by Amazon, eBay, Walmart, FedEx, and many more. Logiwa features a cloud-based system that can be accessed by mobile devices or a web browser. This system includes several fronts, such as the shopping front for users (marketplace), their shopping carts, shipping carriers, and accounting support. Logiwa’s idea is that having all interworking features of shipping products in one system will drastically improve efficiency and therefore save businesses money. For future additions, we thought of adding the feature of scanning barcodes instead of typing in data, and that is a feature Logiwa’s system has as well.

**Main Body of Work**

Our database needed to find a way to link important information about inventory products, orders, and shipping. We decided on six tables in a relational schema to track this information :PRODUCTS, ITEMS, WAREHOUSE, ORDERS, SHIPMENTS, and CUSTOMERS. After normalization, we resulted with seven tables, where SHIPMENTS was split into another table, SHIPS\_FROM. (Refer to Appendix B and C)

PRODUCTS stores information about different types of products that are stored within the warehouse.

ITEMS stores the information about those products within an order.

WAREHOUSE stores information about an individual warehouse within the warehouse chain.

ORDERS stores information for orders placed by customers.

SHIPMENTS stores information about truck shipments and the orders it contains. (Example: a single shipment is one truckload of orders leaving a warehouse)

SHIPS\_FROM stores where orders are shipping from.

CUSTOMERS stores information for commercial customers to this warehouse chain and their information.

**Architecture Overview**

(Refer to Appendix C)

PRODUCTS contains seven attributes. These attributes are as listed:

ProductID is the key to this table and gives an identifying integer to this product.

Quantity is the stock of the product in the warehouse.

Aisle is the aisle it is located.

Plevel is the shelved location within the aisle.

Pcategory refers to the item type (example: HDMI Cable would be under electronics).

Pname is the name of the product.

WarehouseID is a foreign key that refers to the particular warehouse it is stored in.

ITEMS contains five attributes. These attributes are as listed:

OrderID is part of the key that has two attributes. This is a foreign key that refers to the order these item(s) are in.

ProductID is the second part of the key. This is a foreign key that refers to the type of product that is ordered.

Quantity is the amount of that particular item in a single order.

Price is the base price of the product.

Total is the total price of the ordered item(s).

WAREHOUSE contains three attributes. These attributes are as listed:

WarehouseID is the key and gives an identifying integer to each warehouse location.

Email is the email address for a particular warehouse location.

Address is the address for a particular warehouse location.

ORDERS contains five attributes. These attributes are as listed:

OrderID is the key and gives an identifying integer to each order.

OrderDate is the date the order was placed.

StoreID is a foreign key that references the customer that placed the order.

Total is the total price of the order.

Status is the shipping status of the order.

SHIPMENTS contains three attributes. These attributes are as listed:

ShipmentID is the key and gives an identifying integer to each shipment.

OrderID is a foreign key referring to orders in ORDERS and in SHIPS\_FROM that refers to the order on that shipment.

ArrivalDate is the expected date for delivery to the customer.

SHIPS\_FROM contains two attributes and was the result of the process of normalization to BCNF. The attributes are as listed:

OrderID is the key and is the order number to be shipped.

WarehouseID refers to warehouse and is the specific warehouse that that order will be shipped from.

CUSTOMERS contains four attributes. These attributes are as listed:

StoreID is the key and gives an identifying integer to each store.

StoreName is the name of that commercial store.

StoreEmail is the email for that specific store.

StoreAddress is the address for that specific store.

(Refer to Appendix A)

The WAREHOUSE relation is linked to PRODUCTS and SHIPS\_FROM. A warehouse does not need to contain any shipments leaving from SHIPS\_FROM or products, but those shipments must have a warehouse to ship from and products must have a warehouse to be stored in. A product may only be stored in a single warehouse, but a warehouse contain several products up to a set capacity. PRODUCTS is linked to ITEMS as well as WAREHOUSE. Items must contain a product to exist, but a product does not have to be in items, as it may not be ordered. ITEMS then is also linked to ORDERS. An instance of items must have an order and an order must have items it contains. Multiple items may be contained in an order, but each item cannot have more than one order. ORDER is also linked to CUSTOMERS and SHIPMENTS. A customer can have many orders and can also exist without making an order. An order cannot exist without a customer making that order and can only be linked to a single customer. An order does not need a shipment to exist if it is still within the preparation stage (status). A shipment must contain an order to exist though. SHIPMENTS and SHIPS\_FROM are linked as a shipment must ship from somewhere and ships\_from can only exist with a shipment to ship.

**Conclusion**

We were able to create a functional database for our warehouse inventory management system and display the contents of the database on a web interface. The relations are properly linked and each relation works as intended. The database has been normalized in BCNF as well. However, there were features that we wanted the interface to have but didn’t have time to fully implement, with the search bar and the add form being those features.

Contributions (only reflective of Phase III):

E-R Diagrams: Danielle

Relational Schema Diagram: Danielle

Normalization: Calvin and Danielle

Final Report Write-Up: Athena, Danielle, and Calvin.

SQL and Code Files: Athena

Powerpoint: Athena and Danielle

**Future Work**

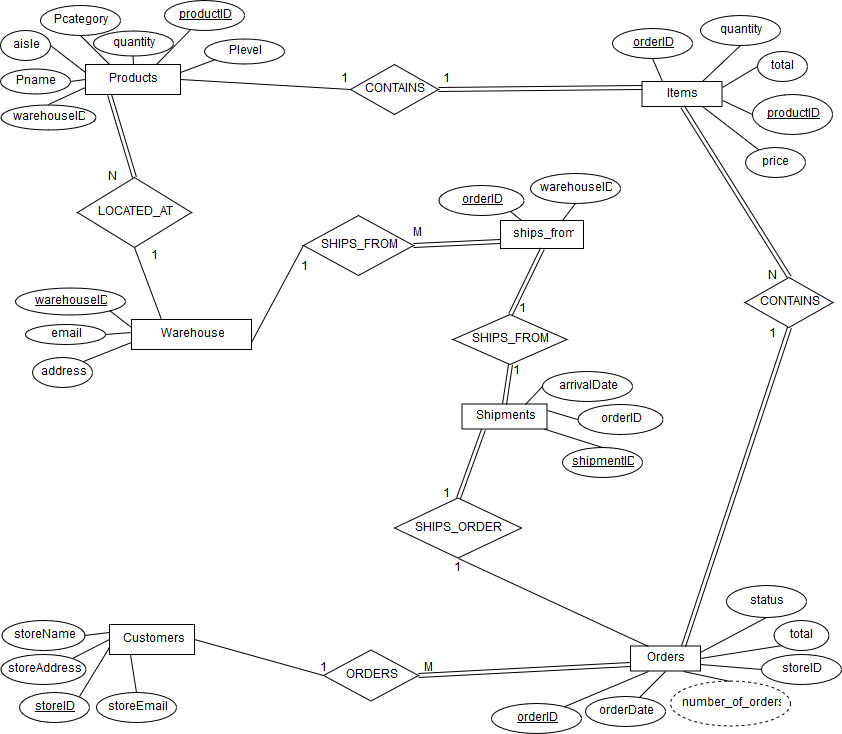
Future work on this project would be to incorporate features for the customers instead of just the warehouses. This could be a feature for customer stores that would allow the customer to keep inventory in their business as well. This inventory system could automatically adjust stock when purchases are made from the business. This would make way for a feature that would allow automatic ordering to the warehouse for low stock items. This would give businesses efficiency and allow less potential for mistakes as they have less calculations to do manually.

Another potential addition on the warehouse side would be integrating hardware, such as scanners, to allow changes to inventory be made by scanning the goods that the warehouse receives and ships out. Orders from customers automatically deduct the inventory, but this would remove the need to manually enter new stock within the warehouse.

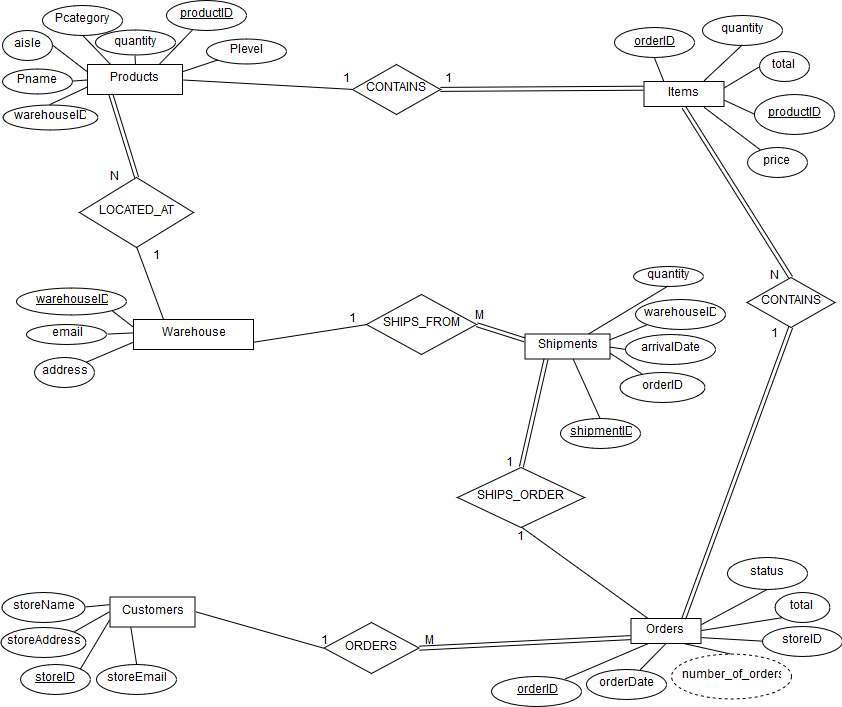
**References**

“Ecommerce Inventory Management Software Online Retail-Logiwa WMS.” *Logiwa Cloud Based Inventory Management Software*, Logiwa, www.logiwa.com/ecommerce-inventory-management-software/.

Appendix A: E-R Diagram:



E-R Diagram Prior to Normalization:



Appendix B: Normalization:

Step 1: Functional Dependencies:

**WAREHOUSE**

warehouseID -> {address, email}

email -> {warehouseID, address}

address -> {warehouseID, email}

**ORDERS**

orderID -> {orderDate, status, total, storeID}

**PRODUCTS**

productID -> {Pname, Pcategory, Plevel, quantity, aisle, warehouseID}

**ITEMS**

{productID, orderID} -> {price, total, quantity}

**CUSTOMERS**

storeID -> {storeName, storeAddress, storeEmail}

storeAddress -> storeName

**SHIPMENTS** (each shipment can have multiple orders)

shipmentID -> {orderID, warehouseID}

orderID -> warehouseID

Step 2: Candidate Keys:

**WAREHOUSE:** warehouseID, address, email

**ORDERS:** orderID

**PRODUCTS:** productID

**ITEMS:** orderID, productID

**CUSTOMERS:** storeID, storeAddress

**SHIPMENTS:** shipmentID

Step 3: Create new relations:

**old:**

**Shipments**

{ShipmentId, ProductID,quantity,warehouseID,arrival date}

**new:**

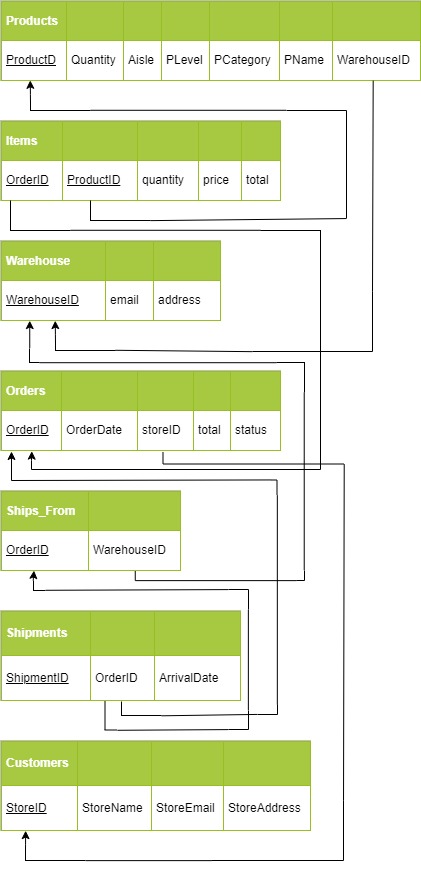
**Shipments**

{arrival\_date, orderID, ShipmentId}

**Ships\_from**

{order\_ID,WarehouseID}

Appendix C: Relational Schema Diagram:



Appendix D: Screenshots of SQL Queries: (Queries had to be updated from Phase II due to normalization)

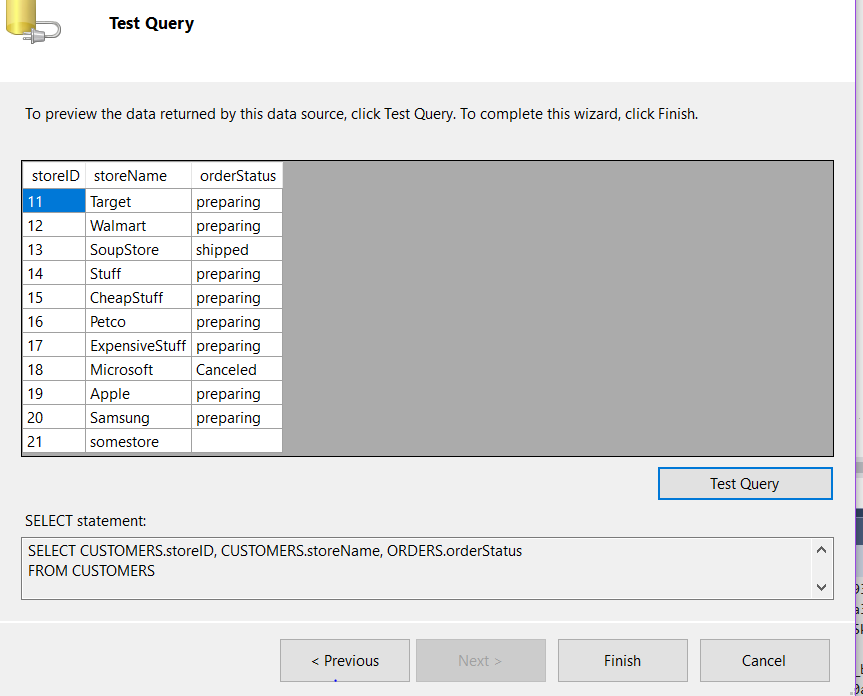
Purpose: Shows order status of customers

SELECT CUSTOMERS.storeID, CUSTOMERS.storeName, ORDERS.orderStatus

FROM CUSTOMERS

LEFT JOIN ORDERS

ON CUSTOMERS.storeID = ORDERS.storeID



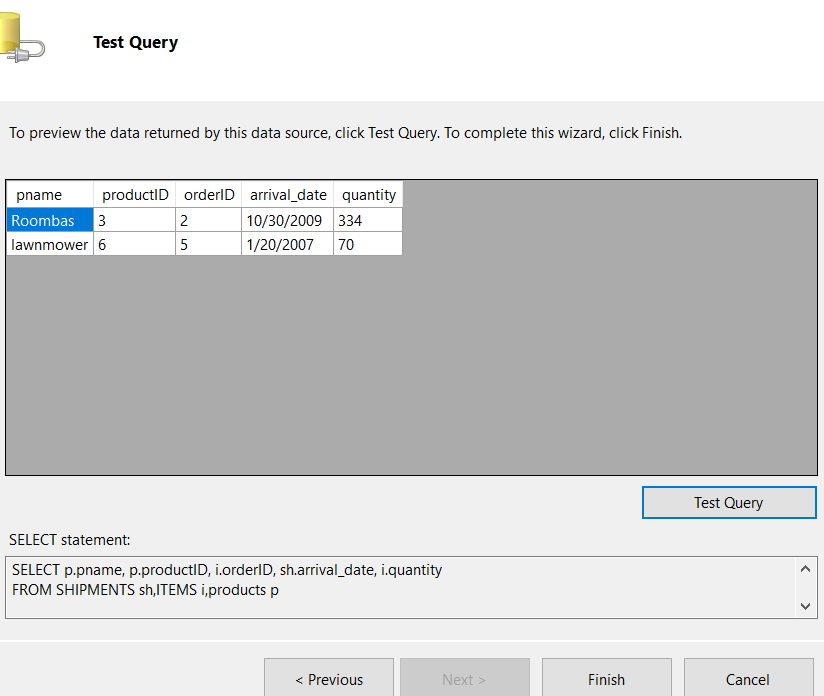
Purpose: Shows most popularly ordered items.

SELECT p.pname, p.productID, i.orderID, sh.arrival\_date, i.quantity

FROM SHIPMENTS sh,ITEMS i,products p

WHERE i.orderID = sh.orderID and i.productID = p.productID and i.quantity > (

SELECT AVG(i.quantity) FROM ITEMS i, SHIPMENTS sh WHERE i.orderID = sh.orderID);

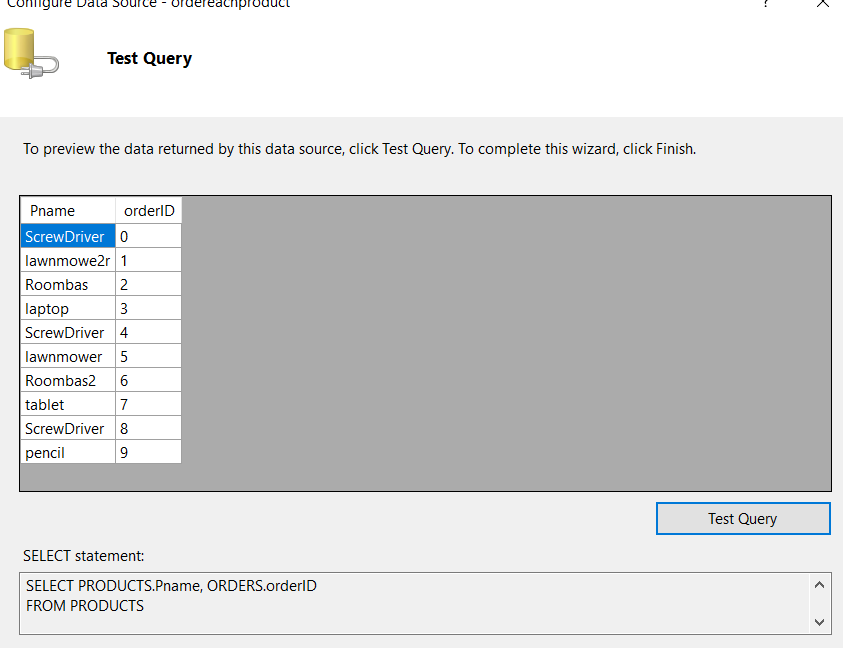


Prurpose: Shows the all the products and orderIDs.

SELECT PRODUCTS.Pname, ORDERS.orderID

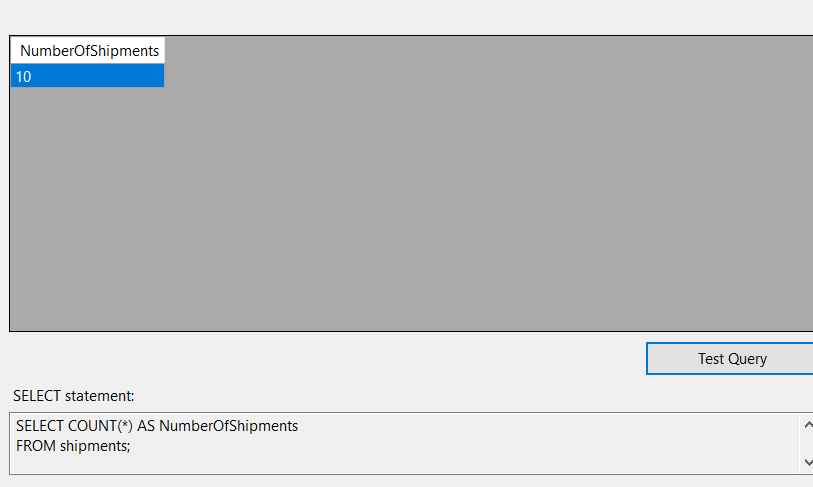
FROM PRODUCTS

FULL OUTER JOIN ORDERS ON PRODUCTS.productID = (select Items.productID from Items where Items.orderID = Orders.orderId);



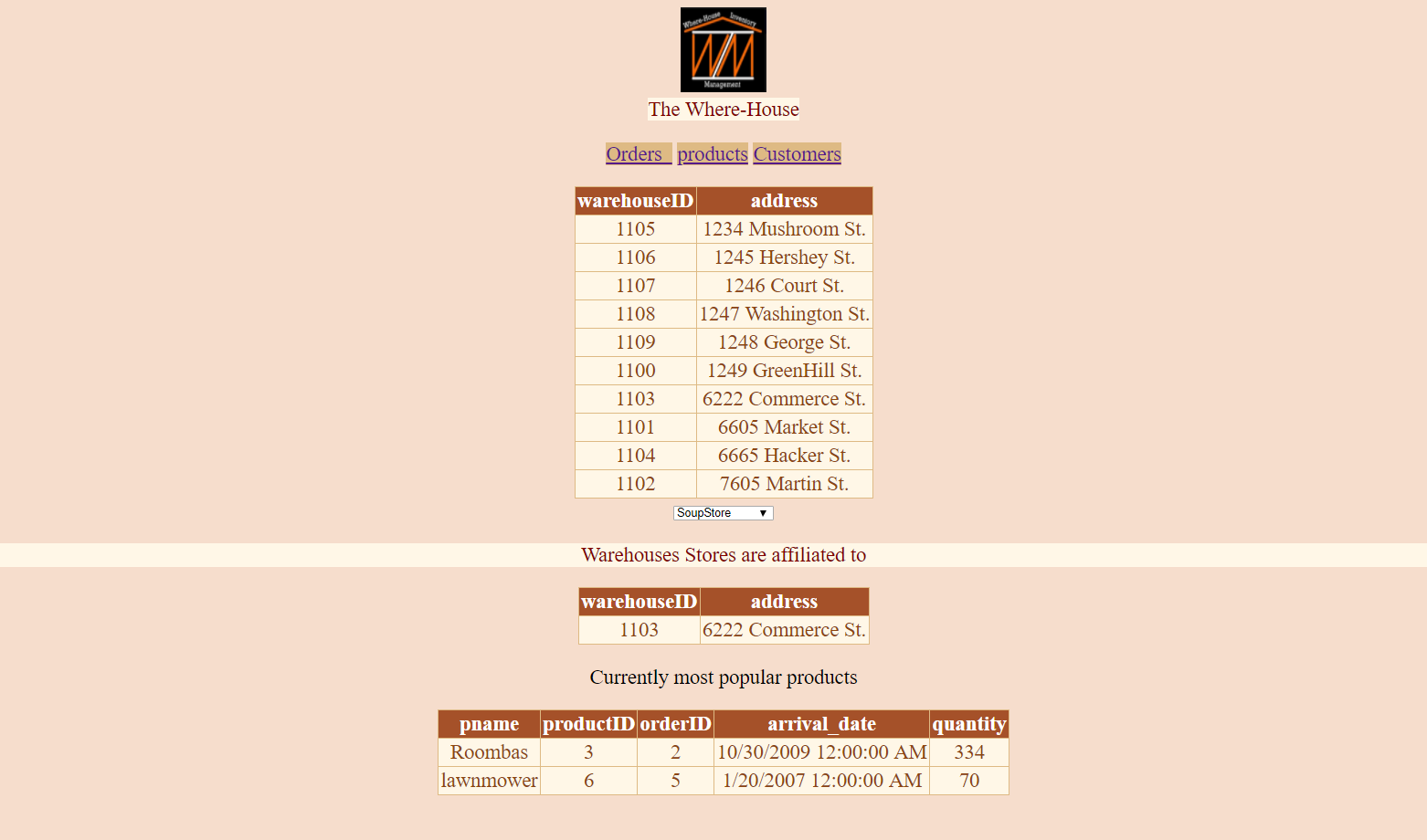
Purpose: Checks if there are no shipments

SELECT COUNT(\*) AS RowCount  
FROM shipments;

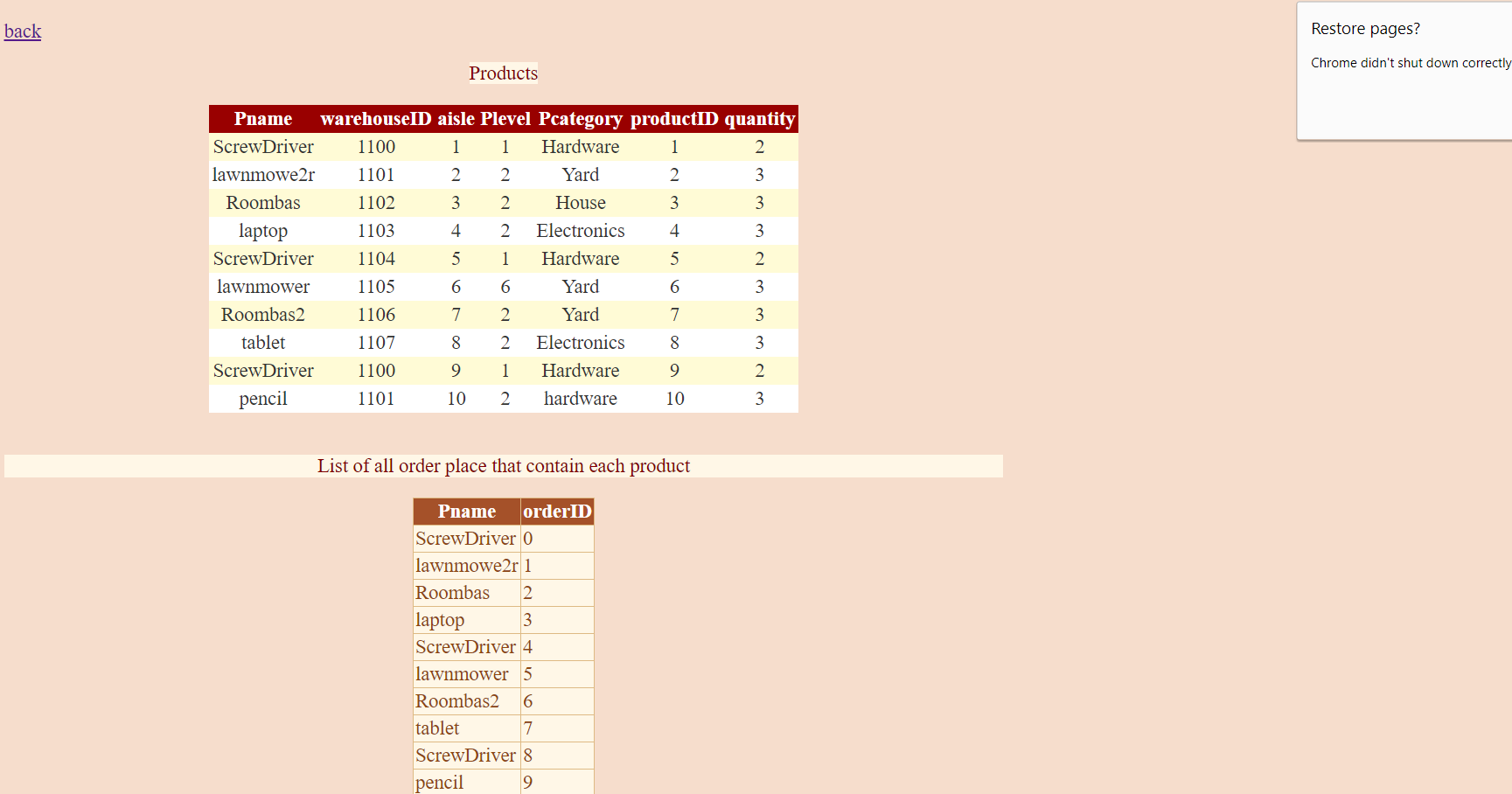


Appendix E: Screenshots of Web Interface:

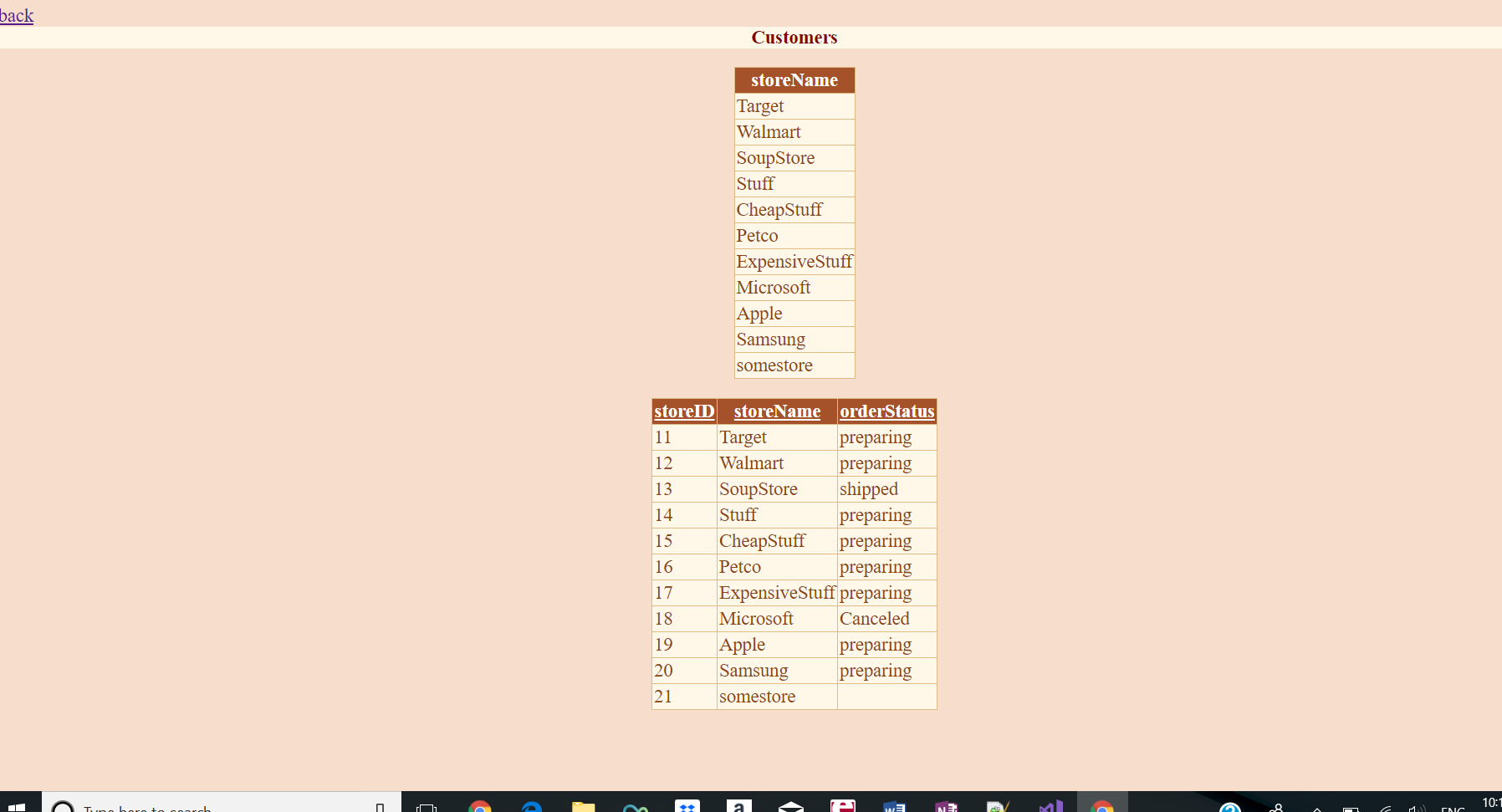
Home page for navigation and warehouse info



Product page for product info



Customer page for customer info



Order page for order info

