

## ABSTRACT

This project was used to develop a realistic experience of a fictional electronics vendor company. The electronics vendor company operates both a website and a chain of physical stores in which I was sought out to design the database which would serve as the underlying operations that would connect everything between the product, employees, website, stores, but more importantly to handle orders efficiently. The system that I designed offers a well-rounded suite for multiple layers of ordering, employee, and customer functionality as it relates to my database. It offers a beautifully crafted website that is customer facing where products and their prices can be viewed and ordered easily. Internally, websites were developed for call center staff to search for customers and enter phone orders quickly, stocking clerks to help them record incoming shipments and update inventory and customer service to check inventory for physical stores and warehouse locations. The database side of the business creates a smooth process where inventory is adjusted automatically, transfers are processed seamlessly, customer data is entered automatically, and everything is linked together by a haste free design. This report covers the problem's description, my database schema and design, implementation details, and an analysis of the running results. Key findings and successful queries will demonstrate that the system effectively streamlines order processing, providing a valuable tool for sales management.

## PROBLEM DESCRIPTION

The electronics vendor currently faces significant challenges due to a database system that is inadequate for the current business model. The manager assigned to solicit database design proposals is also not computer literate and unable to provide detailed specifications for the database design that is needed so I must design it to my own technical specifications. A flawed database can lead to inefficiencies and difficulties in meeting needs of both employees and customers but also disorganization, redundancy and inaccessibility.

There is a lack of many applications by current end users which includes both customers and employees. From an online presence, there is a lack of an elegant web interface for online customers which may restrict or help bring in a wide audience against it's competitors such as Best Buy. A beautifully crafted website that offers a seamless online shopping experience can help maximize profit.

Customer service representatives may struggle to check inventory availability across multiple store locations which may lead to delays in responding to customer inquiries given that they don't have a lookup application to check inventory at stores and warehouses. This could also lead to inaccuracies in inventory information. Connection to the database in a timely manner is essential so that inventory amounts are adjusted consistently.

The current systems limitations may hinder the ability to add phone orders quickly, so the call center staff need an application that allows for quick searching of customers with the ability to select those customers and then place a phone order for that customer with the products that they need and efficiently check that out. This should be connected to the database so that inventory is adjusted promptly.

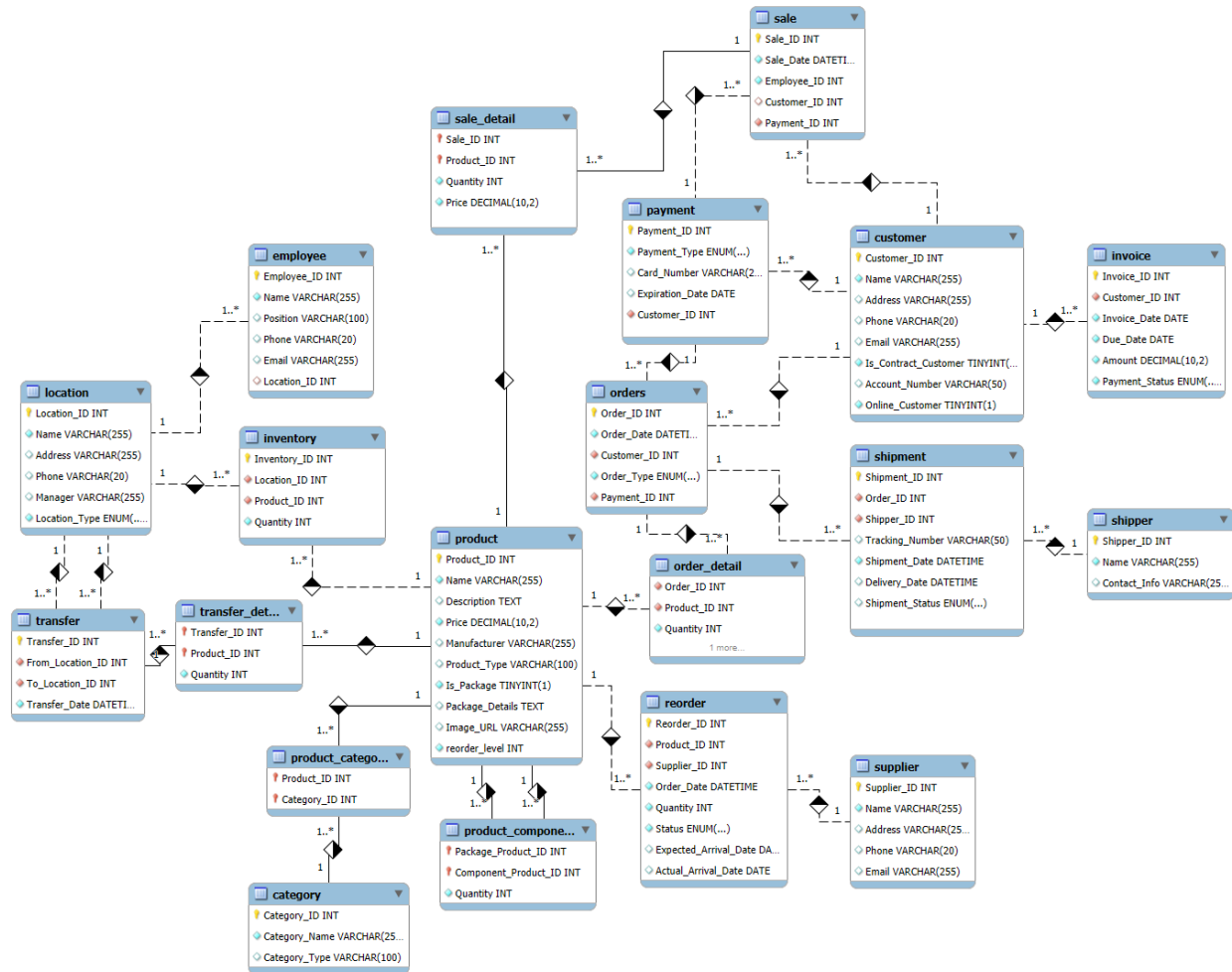
There is currently no warehouse management and stocking clerks at the warehouse need a streamlined application to record shipments, update inventory levels accurately and help ship out products to physical locations. The current system's inefficiencies can lead to discrepancies in inventory records and delays in processing shipments to other locations. A database that can handle incoming inventory adjustments and outgoing inventory adjustments is vital.

Sales reports and marketing reports are extremely vital to make sure a business is successful. Without knowing what areas need to be adjusted or where budgets need to be positioned at, money could be wasted. The current system's limitations hinder basic reporting and analysis needs so a database that can handle the needs listed above and generate successful reports is vital.

This project aims to address all these challenges by developing a comprehensive database and application solution that caters to the specific needs of each user group from customer to employee. This solution will focus on a centralized database that is well structured and can store and manage product information, inventory levels, customer data, orders, payments info, shipper information, and even transfer of inventory between locations. We will address the web applications such as the inventory lookup application for customer service representatives where they can quickly and accurately check inventory across all locations, the call center website where call center staff can search for customers and take phone orders efficiently, a website for stocking clerks to either record incoming shipments to the warehouse or transfer inventory out to physical stores, and successfully using the database for sales reports, special data mining and analysis

## DATABASE DESIGN

The original database design was a paper sketch that included only the basics which started primarily from product because I knew that we would be selling electronic products via our website and in store and everything else would be linked from there. This later evolved into a simple version in SQL Workbench via their ER Diagram. I started adding my original work in through SQL scripts which includes tables, columns, primary keys and foreign keys. I could then jump back and reverse engineer the ER Diagram and see what errors I had and what was connected properly. This later evolved into the final working product shown below.



Let us look at some of these tables and how they all link together. If you look at the customer table, you will see the fields that are associated with it. This allows us to store customer information such as Customer\_ID, Name, Address, etc. The table also includes fields like Is\_Contract Customer and Online\_Customer to help us differentiate between a customer that has a contract with the company and one that is an online customer. You can tell that this table is essential for linking orders and invoices specifically to customers by looking at the tables next to it labeled orders and invoices. There is a red diamond inside of it with Customer\_ID that lets us know it is a foreign key referencing the customer table.

Let us look at another table to see how important a database is. The stocking clerks at the warehouses needed an application to help them record incoming shipments and update inventory. When building the database, I looked at many of these examples to add individual tables before I started connecting everything together. When I first looked at how this application would work, I knew they would need something where they could record inventory or products coming in, but also ship those products out to other locations if they were at a warehouse. We will see later how I set this website up for them to give them two options. They can either receive inventory directly into the warehouse for incoming shipments from the supplier, or they can transfer

products to another warehouse or a physical location. The transfer table includes Transfer\_ID, From\_Location\_ID, To\_Location\_ID, and Transfer\_Date. The transfer\_detail table details the products and quantities that are involved in the transfer. This gives them everything they need to successfully record incoming inventory and transfer inventory from one location to the next, all while recording in real time what is happening.

Using the SQL Workbench I was able to generate effective tables and keys and processed the final relational schema with my foreign key constraints as shown below.

Relational schema:

product(Product\_ID, Name, Description, Price, Manufacturer, Product\_Type, Is\_Package, Package\_Details, Image\_URL, reorder\_level)  
employee(Employee\_ID, Name, Position, Phone, Email, Location\_ID)  
category(Category\_ID, Category\_Name, Category\_Type)  
supplier(Supplier\_ID, Name, Address, Phone, Email)  
customer(Customer\_ID, Name, Address, Phone, Email, Is\_Contract\_Customer, Account\_Number, Online\_Customer)  
payment(Payment\_ID, Payment\_Type, Card\_Number, Expiration\_Date, Customer\_ID)  
orders(Order\_ID, Order\_Date, Customer\_ID, Order\_Type, Payment\_ID)  
sale(Sale\_ID, Sale\_Date, Employee\_ID, Customer\_ID, Payment\_ID)  
location(Location\_ID, Name, Address, Phone, Manager, Location\_Type)  
transfer(Transfer\_ID, From\_Location\_ID, To\_Location\_ID, Transfer\_Date)  
inventory(Inventory\_ID, Location\_ID, Product\_ID, Quantity)  
reorder(Reorder\_ID, Product\_ID, Supplier\_ID, Order\_Date, Quantity, Status, Expected\_Arrival\_Date, Actual\_Arrival\_Date)  
shipper(Shipper\_ID, Name, Contact\_Info)  
shipment(shipment\_ID, Order\_ID, Shipper\_ID, Tracking\_Number, Shipment\_Date, Delivery\_Date, Shipment\_Status)

Foreign key constraints:

employee: Location\_ID references location(Location\_ID)  
payment: Customer\_ID references customer(Customer\_ID)  
orders: Customer\_ID references customer(Customer\_ID)  
orders: Payment\_ID references payment(Payment\_ID)  
sale: Employee\_ID references employee(Employee\_ID)  
sale: Customer\_ID references customer(Customer\_ID)  
sale: Payment\_ID references payment(Payment\_ID)  
transfer: From\_Location\_ID references location(Location\_ID)  
transfer: To\_Location\_ID references location(Location\_ID)

inventory: Location\_ID references location(Location\_ID)  
inventory: Product\_ID references product(Product\_ID)  
reorder: Product\_ID references product(Product\_ID)  
reorder: Supplier\_ID references supplier(Supplier\_ID)  
shipmen: Order\_ID references orders(Order\_ID)  
shipment: Shipment\_ID references shipment(Shipment\_ID)

As I stated previously, my original paper sketch of the ER Diagram started with the product, simply because I knew that everything would be linked from here given that we were an electronic vendor company and would be selling this product to customers. That is why when you look at the final ER diagram, the product table is more centered in the diagram with everything else linking to it in some manner. From there we needed those products in inventory. Every product is listed in inventory in the database across three physical stores and two warehouse locations. A customer can order from the website which saves information in the customer table of the database and what they ordered in the orders table of the database, records what product is selected from the products table and reduces that product from the quantity in the inventory table of the database. Information from the orders table is automatically sent to the order\_detail table which is then sent to the shipment table that automatically assigns a shipping company from the shipper table, a tracking number and a shipping date. If the quantity of that product is low in store it will automatically trigger a reorder from the reorder table and notify the supplier who will then ship to our warehouse where the shipping clerk can use their new website to record that incoming inventory and then transfer that inventory back to the store using the transfer table that will feed into the transfer\_detail table. This is a very intricate database design that all works in the background just from clicking the place order button from the customer.

## IMPLEMENTATION DETAILS

I did want to take this project a bit further than just the database side. I wanted a full featured website design that would be fully interactive with that database, so I implemented a full stack web design on top of the database for full interaction.

For the backend I used Node.js and Express.js since it does work well with high-volume requests and real-time data interactions. It was also ideal for multiple requests to the database simultaneously and high-level traffic. With Express.js I was able to use efficient routing in my code such as POST /submit-order, GET /products, and GET /customers and middleware for logging, error handling and parsing a ton of JSON requests. Probably one of the most important aspects of using this was the data validation and error handling. I had many issues with connecting everything together and reaching my database, so within my code you will see validation checks after pretty much each request to send data to the front end and vice versa. I

used the mysql2 library with Node.js to connect to my database. My app.js handles all my connection information to that database and serves all of my query executions.

The front-end interface for all of my websites for both the customers and the employees was built using HTML, CSS, and JavaScript to enable a rich, clean, and user-friendly experience.

I used HTML to create the structural components of the websites and any forms that you see such as searching for customers, selecting products, or filling out billing and shipping information. CSS was used to create the visually appealing layout for all the webpages but more so for the main customer facing website where all of the products are located. CSS customizations were added to ensure alignment with tables and important elements such as the products, the names of those products, drop down menus, banners, form sections and even the buttons on the checkout page. JavaScript was used to handle client-side interaction such as updates to product list, order totals, form validations and conditional displaying of fields such as the credit/debit card field on the cart page.

Everything else was communicated through JSON commands and REST APIs between the front end and back end. Again, my endpoints like GET /api/customers retrieves all the customer data and POST /submit-order is used for submitting the order. I also used Fetch API throughout my code to interact with the backend to pull data for customer searches and product selection among other things.

MySQL was selected as the database management mainly due to what was presented during class but also because it has a very structured approach to relational data, reliability, and ability to handle large datasets. The database schema was structured to incorporate primary and foreign keys to establish clear relationships between tables to allow for efficient data retrieval and ensure data integrity. Indexes were applied on frequently searched fields such as Customer\_ID and Product\_ID to reduce query response time. I used transactions for complex operations like placing an order so that all parts of the process either succeeded or failed together to preserve data consistency.

My biggest hurdle through everything was being able to click the “Place Order” button on the cart page and getting “Order Placed Successfully” and having all the information sent properly to the database. This is where most of my error logging came into place and I also resorted to using Postman to make sure all of my API endpoints were set up properly. With success it finally worked! When clicking that button, I was able to see my implementation in the works. The backend would validate and store the data into the relevant tables in MySQL such as customer, orders, order\_detail, and shipment. The backend would use transaction handling to ensure that all related records were committed together such as the order and inventory adjustments, and then on the frontend I was receiving a confirmation from the backend with a popup that showed “Order Placed Successfully”, the cart was cleared, and I was redirected to the homepage.

I had successful implementation of a powerful backend with a beautiful customer and employee driven front end interface across all my web pages with efficient data handling capabilities. MySQL works perfectly with this as it is popular, has great structure and reliable storage solution. With all of these together, it offers security, scalability and performance which is what I will need since I would like to pursue this project moving forward outside of this class.

## RUNNING RESULTS AND ANALYSIS

Let us run some tests to make sure our database has been adequately set up. The manager has asked us to run a few queries.

Assume the package shipped by USPS with tracking number A1WMOWVAZB8 is reported to have been destroyed in an accident. Find the contact information for the customer. Also, find the contents of that shipment and create a new shipment of replacement items.

Perfect, let us first find the contact information for the customer with the tracking number A1WMOWVAZB8.

```
1 • SELECT
2     s.Shipment_ID,
3     o.Customer_ID,
4     c.Name AS Customer_Name,
5     c.Address AS Customer_Address,
6     c.Phone AS Customer_Phone,
7     c.Email AS Customer_Email
8 FROM
9     shipment s
10 JOIN
11     orders o ON s.Order_ID = o.Order_ID
12 JOIN
13     customer c ON o.Customer_ID = c.Customer_ID
14 WHERE
15     s.Tracking_Number = 'A1WMOWVAZB8'
16     AND s.Shipper_ID = '4';
```

Result Grid					
Filter Rows: <input type="text"/> Export:  Wrap Cell Content:					
Shipment_ID	Customer_ID	Customer_Name	Customer_Address	Customer_Phone	Customer_Email
25	143	Ryan Hawkins	856 Ocean Springs Dr	7699993456	Ryan.hawkins@fairview.com

Result 2 x						
Output						
Action Output						
#	Time	Action	Message			
1	14:31:10	SELECT s.Shipment_ID, o.Customer_ID, c.Name AS Customer_Name, c.Address AS Customer_Address, c.Phone AS Customer_Ph...	1 row(s) returned			

All right, so we have our customer Ryan Hawkins with an order that was assigned to Shipment\_ID of 25. Let us look and see what contents were attached to Shipment\_ID 25.

```
1 • SELECT
2     od.Product_ID,
3     p.Name AS Product_Name,
4     od.Quantity,
5     od.Price
6 FROM
7     order_detail od
8 JOIN
9     product p ON od.Product_ID = p.Product_ID
10 WHERE
11     od.Order_ID = (
12         SELECT Order_ID FROM shipment WHERE Shipment_ID = '25'
13     );
```

Result Grid				
Filter Rows: <input type="text"/>   Export:    Wrap Cell Content:				
	Product_ID	Product_Name	Quantity	Price
▶	30	Whirlpool Side-by-Side Refrigerator	1	1299.00
	32	LG Counter Depth Max Refrigerator	1	1399.99

Result 1 ×							
Output							
Action Output							
#	Time	Action	Message				
✓ 1	14:31:10	SELECT s.Shipment_ID, o.Customer_ID, c.Name AS Customer_Name, c.Address AS Customer_Address, c.Phone AS Customer_Ph...	1 row(s) returned				
✓ 2	15:09:39	SELECT od.Product_ID, p.Name AS Product_Name, od.Quantity, od.Price FROM order_detail od JOIN product p ON od.Produ...	2 row(s) returned				



```
SQL Tables*   SQL File 7*   SQL File 9*   SQL File 10* x   SQL File 11*
[Icons] | Limit to 1000 rows | [Icons]
1  INSERT INTO shipment (Order_ID, Shipper_ID, Tracking_Number, Shipment_Date, Delivery_Date, Shipment_Status)
2  VALUES (
3      39,
4      4,
5      'D85HGK4WY7',
6      NOW(),
7      DATE_ADD(NOW(), INTERVAL 5 DAY),
8      'On-Time'
9  );
```

Output

Action Output

#	Time	Action	Message
✓ 1	15:52:33	INSERT INTO shipment (Order_ID, Shipper_ID, Tracking_Number, Shipment_Date, Delivery_Date, Shipment_Status) VALUES ( 39, 4, 'D...	1 row(s) affected

```
1 ● SELECT * FROM shipment;
```

Result Grid

Filter Rows:

Edit:

Export/Import:

Wrap Cell Content:

	Shipment_ID	Order_ID	Shipper_ID	Tracking_Number	Shipment_Date	Delivery_Date	Shipment_Status
17	31	1	E5HVOS53LQF	2024-10-23 11:43:17	2024-10-28 11:43:17	On-Time	
18	32	1	3UFL2DXHXC2	2024-10-23 11:50:48	2024-10-29 11:50:48	On-Time	
19	33	1	HW2UW5J301G	2024-10-23 11:55:45	2024-10-28 11:55:45	On-Time	
20	34	1	TFU6T6YL3CM	2024-10-23 12:07:24	2024-10-29 12:07:24	Late	
21	35	1	6RZ64MMRYRF	2024-10-23 12:16:01	2024-10-28 12:16:01	On-Time	
22	36	1	KIAVYT8WD	2024-10-23 12:28:54	2024-10-29 12:28:54	On-Time	
23	37	1	R6RI9ENICLQ	2024-10-23 12:33:32	2024-10-28 12:33:32	On-Time	
24	38	1	V00J2E2SODC	2024-10-27 17:43:27	2024-10-30 17:43:27	On-Time	
25	39	4	A1WMOWVAZB8	2024-10-27 17:43:30	2024-10-30 17:43:30	Destroyed	
26	40	1	TJ9P8FGRGCQ	2024-10-27 17:45:24	2024-11-01 17:45:24	On-Time	
27	41	1	MI4E7M7B69	2024-10-27 17:50:36	2024-10-31 17:50:36	On-Time	
28	42	1	5ED3C6SBK7E	2024-10-27 17:57:59	2024-10-30 17:57:59	On-Time	
29	43	1	SQKF3USTQRO	2024-10-27 18:05:59	2024-10-31 18:05:59	On-Time	
30	44	1	ACRDSTYIO6U	2024-10-27 18:26:55	2024-10-30 18:26:55	On-Time	
33	39	4	D85HGK4WY7	2024-10-30 15:52:33	2024-11-04 15:52:33	On-Time	
	NULL	NULL	NULL	NULL	NULL	NULL	

```
1 • SELECT
2     od.Product_ID,
3     p.Name AS Product_Name,
4     od.Quantity,
5     od.Price
6 FROM
7     shipment s
8 JOIN
9     orders o ON s.Order_ID = o.Order_ID
10 JOIN
11     order_detail od ON o.Order_ID = od.Order_ID
12 JOIN
13     product p ON od.Product_ID = p.Product_ID
14 WHERE
15     s.Shipment_ID = 33;
```

Result Grid				
		Filter Rows:	Export:	Wrap Cell Content:
	Product_ID	Product_Name	Quantity	Price
▶	30	Whirlpool Side-by-Side Refrigerator	1	1299.00
	32	LG Counter Depth Max Refrigerator	1	1399.99

Next, let us find the customer who has bought the most (by price) in the past year.

```
1 • SELECT
2     c.Customer_ID,
3     c.Name,
4     c.Email,
5     c.Phone,
6     SUM(od.Quantity * od.Price) AS Total_Spent
7 FROM
8     customer c
9 JOIN
10    orders o ON c.Customer_ID = o.Customer_ID
11 JOIN
12    order_detail od ON o.Order_ID = od.Order_ID
13 WHERE
14    o.Order_Date >= DATE_SUB(CURDATE(), INTERVAL 1 YEAR)
15 GROUP BY
16    c.Customer_ID, c.Name, c.Email
17 ORDER BY
18    Total_Spent DESC
19 LIMIT 1;
```

Result Grid

Filter Rows:

Export:

Wrap Cell Content:




Fetch rows:

	Customer_ID	Name	Email	Phone	Total_Spent
▶	141	Taylor Swift	taylor.swifty@records.biz	90090090000	129146.49

We can see that our biggest spender was none other than Taylor Swift with a Customer\_ID of 141.

Next, we need to find the top two products by dollar-amount sold in the past year.




```
1 • SELECT
2     p.Product_ID,
3     p.Name AS Product_Name,
4     SUM(od.Quantity * od.Price) AS Total_Revenue
5 FROM
6     product p
7 JOIN
8     order_detail od ON p.Product_ID = od.Product_ID
9 JOIN
10    orders o ON od.Order_ID = o.Order_ID
11 WHERE
12    o.Order_Date >= DATE_SUB(CURDATE(), INTERVAL 1 YEAR)
13 GROUP BY
14    p.Product_ID, p.Name
15 ORDER BY
16    Total_Revenue DESC
17 LIMIT 2;
```

Result Grid			
Filter Rows: <input type="text"/>			
Export: 			
Wrap Cell Content: 			
Fetch rows: 			
	Product_ID	Product_Name	Total_Revenue
▶	98	Apple 12.9 inch iPad Pro M2 Wi-Fi	64950.00
	156	Samsung 75 inch QLED 8K TV	29999.85

Looking at our query here, our top 2 products by dollar amount sold in the past year are the Apple 12.9-inch iPad Pro with the M2 chip and a Samsung 75-inch QLED 8K TV!

Let us find the top two products by unit sales in the past year.




```
1 • SELECT
2     p.Product_ID,
3     p.Name AS Product_Name,
4     SUM(od.Quantity) AS Total_Units_Sold
5 FROM
6     product p
7 JOIN
8     order_detail od ON p.Product_ID = od.Product_ID
9 JOIN
10    orders o ON od.Order_ID = o.Order_ID
11 WHERE
12     o.Order_Date >= DATE_SUB(CURDATE(), INTERVAL 1 YEAR)
13 GROUP BY
14     p.Product_ID, p.Name
15 ORDER BY
16     Total_Units_Sold DESC
17 LIMIT 2;
```





result Grid			
Filter Rows: <input type="text"/>			
Export: 			
Wrap Cell Content: 			
Fetch rows: 			
Product_ID	Product_Name	Total_Units_Sold	
51	Apple AirPods Pro	60	
97	Apple 10.2 inch iPad with Wi-Fi 64GB	50	

Apple Air Pods Pro and Apple 10.2-inch iPad with Wi-Fi were huge hits!

The manager also wants us to check if we have any locations that currently have products that are out of stock.

```
1 • SELECT
2     p.Product_ID,
3     p.Name AS Product_Name,
4     l.Name AS Store_Name
5 FROM
6     product p
7 JOIN
8     inventory i ON p.Product_ID = i.Product_ID
9 JOIN
10    location l ON i.Location_ID = l.Location_ID
11 WHERE
12    l.Location_Type = 'Store'
13 GROUP BY
14    p.Product_ID, p.Name, l.Name
15 HAVING
16    SUM(i.Quantity) = 0;
```

Result Grid    Filter Rows: <input type="text"/>   Export:    Wrap Cell Content: 			
	Product_ID	Product_Name	Store_Name
22	22	Samsung 6 Cu Ft Smart Duo Gas Range	Xplosive Electronics Madison
29	29	LG Top Freezer Refrigerator	Xplosive Electronics Gulfport

result 13 x 			
Output 			
 Action Output 			
#	Time	Action	Message
1	16:16:27	SELECT p.Product_ID, p.Name AS Product_Name, l.Name AS Store_Name FROM...	2 row(s) returned

This confirms that we do have two locations with inventory out of stock. Our Madison location has 0 inventory of Samsung Smart Duo gas ranges, and our Gulfport location has 0 inventory of LG Top Freezer refrigerators.

Our manager also wants us to be able to find those packages that were not delivered within the promised time.

```
1 • SELECT
2     s.Shipment_ID,
3     s.Tracking_Number,
4     s.Order_ID,
5     s.Shipment_Date,
6     s.Delivery_Date,
7     s.Shipment_Status,
8     c.Name AS Customer_Name,
9     c.Email AS Customer_Email
10  FROM
11     shipment s
12  JOIN
13     orders o ON s.Order_ID = o.Order_ID
14  JOIN
15     customer c ON o.Customer_ID = c.Customer_ID
16  WHERE
17     s.Shipment_Status = 'Late';
```

Result Grid | | Filter Rows:  | Export: | Wrap Cell Content:

	Shipment_ID	Tracking_Number	Order_ID	Shipment_Date	Delivery_Date	Shipment_Status	Customer_Name	Customer_Email
2	P8XNP0A5UZ	16	2024-10-20 14:06:10	2024-10-23 14:06:10	Late	Michael Jones	mjj121616@gmail.com	
8	MMZAM8BAG0N	22	2024-10-20 16:00:47	2024-10-24 16:00:47	Late	Michael Jones	mjj121616@gmail.com	
9	OWE8YCGRE8	23	2024-10-20 16:01:26	2024-10-23 16:01:26	Late	Michael Jones	mjj121616@gmail.com	
16	SV136KSOG	30	2024-10-23 10:23:03	2024-10-25 10:23:03	Late	HELEN HUNT	HELEN.HUNT@GMAIL.COM	
20	TFU6T6YL3CM	34	2024-10-23 12:07:24	2024-10-29 12:07:24	Late	Michael Jones	mjj121616@gmail.com	

result 14 x

Output

Action Output

#	Time	Action	Message
1	16:23:13	SELECT s.Shipment_ID, s.Tracking_Number, s.Order_ID, s.Shipment_Date, s...	5 row(s) returned

We can see below a list of shipments with their tracking numbers that were considered late. We could also dig further and find customer information, product information etc. Looks like Michael Jones needs to be reached out to!

Our final test query will be to generate a bill for each customer for the past month.

```
1 • SELECT
2     c.Customer_ID,
3     c.Name AS Customer_Name,
4     c.Email AS Customer_Email,
5     SUM(od.Quantity * od.Price) AS Total_Amount,
6     '2024-10-31' AS Invoice_Date,
7     DATE_ADD('2024-10-31', INTERVAL 90 DAY) AS Due_Date,
8     'Pending' AS Payment_Status
9 FROM
10    customer c
11 JOIN
12    orders o ON c.Customer_ID = o.Customer_ID
13 JOIN
14    order_detail od ON o.Order_ID = od.Order_ID
15 WHERE
16    o.Order_Date >= '2024-10-01' AND o.Order_Date <= '2024-10-31'
17    AND c.Is_Contract_Customer = 1
18 GROUP BY
19    c.Customer_ID, c.Name, c.Email;
```

Result Grid							
		Filter Rows:		Export:	Wrap Cell Content:		
	Customer_ID	Customer_Name	Customer_Email	Total_Amount	Invoice_Date	Due_Date	Payment_Status
▶	2	Jane Smith	jane.smith@gmail.com	129.99	2024-10-31	2025-01-29	Pending
	17	Zach Smith	zach.smith@gmail.com	44674.80	2024-10-31	2025-01-29	Pending

Result 9 ×			
Output			
Action Output			
#	Time	Action	Message
✓ 1	16:39:19	SELECT c.Customer_ID, c.Name AS Customer_Name, c.Email AS Customer_Email,...	2 row(s) returned

It looks like currently Zach Smith and Jane Smith are the only ones that have placed an order that was billed to their account that we need to invoice!





The marketing department also needs to be able to do sales reports and may also want to do special data mining and analysis. The database we have set up will allow them to easily do this. Let's look at some examples. The first example shows total sales so far for the year 2024 while also showing the total number of orders for the year.

```
1 • SELECT
2     MONTH(Order_Date) AS Month,
3     SUM(od.Quantity * od.Price) AS Total_Sales_Amount,
4     COUNT(DISTINCT o.Order_ID) AS Total_Orders
5 FROM
6     orders o
7 JOIN
8     order_detail od ON o.Order_ID = od.Order_ID
9 WHERE
10    YEAR(Order_Date) = 2024
11 GROUP BY
12    MONTH(Order_Date);
```

Result Grid			
Filter Rows: <input type="text"/>			
Export: <input type="button" value="Export"/>			
Wrap Cell Content: <input type="button" value="Wrap"/>			
	Month	Total_Sales_Amount	Total_Orders
+	10	234342.50	52



There are also many various data mining techniques that we can set up for marketing. This example is a very important one that can be used which simply identifies which products are often bought together and how many times that happens. This can help marketing with cross-selling strategies.

```
1 • SELECT
2     p1.Product_ID AS Product_A,
3     p2.Product_ID AS Product_B,
4     COUNT(*) AS Frequency
5 FROM
6     order_detail p1
7 JOIN
8     order_detail p2 ON p1.Order_ID = p2.Order_ID AND p1.Product_ID < p2.Product_ID
9 GROUP BY
10    Product_A, Product_B
11 HAVING
12    Frequency > 5;
```

Result Grid			
Filter Rows: <input type="text"/>			
Export:  Wrap Cell Content: 			
	Product_A	Product_B	Frequency
	126	156	22
▶	30	32	7

We could even find customers based on their purchase history, finding frequency or occasional buyers.

```
1 • SELECT
2     c.Customer_ID,
3     c.Name,
4     COUNT(o.Order_ID) AS Order_Count,
5     SUM(od.Quantity * od.Price) AS Total_Spent
6 FROM
7     customer c
8 JOIN
9     orders o ON c.Customer_ID = o.Customer_ID
10 JOIN
11     order_detail od ON o.Order_ID = od.Order_ID
12 GROUP BY
13     c.Customer_ID
14 ORDER BY
15     Total_Spent DESC;
```

Result Grid    Filter Rows: <input type="text"/>   Export:  Wrap Cell Content				
	Customer_ID	Name	Order_Count	Total_Spent
+	141	Taylor Swift	7	129146.49
	147	Thomas Evertson	3	4498.97
	125	Michael Jones	3	2259.97

We can also create “Stored Procedures” with most of these to generate recurring reports without needing to write queries.

The screenshot shows a database management interface. On the left, a 'SCHEMAS' pane lists the database structure, including tables like 'category', 'customer', 'employee', 'inventory', 'invoice', 'location', 'order\_detail', 'orders', 'payment', 'product', 'product\_category', 'product\_component', 'reorder', 'sale', 'sale\_detail', 'shipment', 'shipper', 'supplier', 'transfer', and 'transfer\_detail'. The 'Stored Procedures' section is expanded, showing 'GetMonthlySales' selected. The main query editor on the right contains the following SQL code:

```
1 • call xplosiveelectronics.GetMonthlySales(2024);
2
```

Below the query editor, a 'Result Grid' displays the output of the stored procedure. The grid has three columns: 'Month', 'Total\_Sales\_Amount', and 'Total\_Orders'. The data shown is for month 10, with a total sales amount of 234342.50 and 52 total orders.

Month	Total_Sales_Amount	Total_Orders
10	234342.50	52

## INTERFACES

First, we will look at the first webpage that we created which is for customer service. This allows them to lookup inventory at any physical locations or the two warehouses in real time.



### Inventory Lookup

Select Store Location:

Product Name	Description	Quantity
Samsung Dishwasher 440 Series	Samsung Dishwasher 440 Series	5
Bosch 550 Series Smart Control Dishwasher	Bosch 550 Series Smart Control Dishwasher	17
LG Dishwasher	LG Dishwasher	19
Bosch 300 Series Smart Control	Bosch 300 Series Smart Control	15
Kitchenaid Top Control Dishwasher	Kitchenaid Top Control Dishwasher	30
GE Top Control Fingerprint Resistant Dishwasher	GE Top Control Fingerprint Resistant Dishwasher	10
Frigidaire Front Control MaxDry Dishwasher	Frigidaire Front Control MaxDry Dishwasher	18
Samsung AutoRelease Smart Built-In Dishwasher	Samsung AutoRelease Dry Smart Built-In Stainless Steel Tub Dishwasher	13
Maytag 24 Front Control Built-In Dishwasher	Maytag 24 Front Control Built-In Dishwasher	22
GE 1.6 Cu. Ft. Over-the-Range Microwave	GE 1.6 Cu. Ft. Over-the-Range Microwave - Stainless Steel	20

The next interface was created for the call center staff. This website allows them quickly to search for customers, select that customer, then search for products and add them to an order. They can then place that order directly from the website!



## Customer Service - Order Entry

Search for Customer (Name or Phone):

Search Customer

Customer: Taylor Swift, Phone: 90090090000

Search for Product:

Search Product

Product: Bosch 550 Series Smart Control Dishwasher, Price: \$1169.99

Product: Bosch 300 Series Smart Control, Price: \$899.99

Select Payment Method:

### Card Information

Card Number:

Expiration Date:

Security Code:

Place Order

Loading up the page they will have two options. “Receive Inventory In” or “Transfer Inventory to Store”.



Let us look at the “Receive Inventory In” page. From this page they can start to search for a product, and it will retrieve those items from the names in the inventory table of the database. Once selected they can enter the quantity that they want to add. At the bottom they can click the “Receive Inventory” button for any number of products and their quantity, and it will update the inventory for that item for the North Warehouse in the database in real time.

## Receive Inventory into Xplosive Electronics (North Warehouse)

app ▼

Quantity		
Search product		
Quantity		
Search product		
Quantity		
Search product		
Quantity		

- Apple AirPods Pro
- Apple - AirPods Max - Space Gray
- Apple - AirPods Pro 2 - White
- Apple MacBook Air 13 inch Laptop
- Apple MacBook Pro 14 inch Laptop
- Apple 10.2 inch iPad with Wi-Fi 64GB
- Apple 12.9 inch iPad Pro M2 Wi-Fi
- Apple 11 inch iPad Pro M4 chip Wi-Fi 512GB
- Apple 13 inch iPad Air M2 chip Wi-Fi 256GB
- Apple Homepod Smart Speaker with Siri
- Apple 4K 64GB
- Apple - Siri Remote (3rd Generation)



On the “Transfer Inventory to Store” page they can select the invoice order information, then select the location they are shipping to. Based on the location they are shipping to, whatever product and quantity they enter, it will add to the transfer and transfer\_detail tables of the database while also updating the product quantities in the inventory table for that location.

## Inventory Transfer

Invoice Order Date:

mm/dd/yyyy



Invoice Received Date:

mm/dd/yyyy



Invoice Number:

Location:

Madison



Madison

Hattiesburg

Gulfport

South Warehouse

Quantity


Search products...

We saved the best for last which is our elegant web interface for our online customers which is fully integrated with our database. All customers are stored in the database by Customer\_ID and all products they order are grouped into the orders table and orders\_detail table. Shipping information is supplied through the shipment table. The shipper for that shipment is chosen from that order automatically which flows into the shipment table also. If a product is low, that is, at 5 or below, it will automatically notify the supplier and enter information into the supplier table. Let's look at the main page first. All categories at the top you can hover over to drop down into more categories.




Once underneath a category, you can see all the products that are available. We also have a view inventory button at the bottom of the page for each location where customers can see what is in stock by location.


### Laptops and Desktops




**Dell Inspiron 15 Touch Screen Laptop**  
Dell Inspiron 15 Touch Screen Laptop - Intel Core i5 - 8GB - 512GB SSD  
Price: \$379.99  
Manufacturer: Dell  
[Click "View Inventory" to see stock](#)  
[Add to Cart](#)




**Apple MacBook Air 13 inch Laptop**  
Apple - MacBook Air 13-inch Laptop - M3 chip - 8GB Memory - 256GB SSD  
Price: \$899.00  
Manufacturer: Apple  
[Click "View Inventory" to see stock](#)  
[Add to Cart](#)




**Apple MacBook Pro 14 inch Laptop**  
Apple - MacBook Pro 14" Laptop - M3 Pro Chip - 16GB Memory - 14 core GPU - 512GB SSD  
Price: \$1499.00  
Manufacturer: Apple  
[Click "View Inventory" to see stock](#)  
[Add to Cart](#)




**Lenovo Ideapad 11 15.6 FHD Touchscreen Laptop**  
Lenovo Ideapad 11 15.6" FHD Touchscreen Laptop Intel Core i3 - 8GB Memory - Intel UHD Graphics - 256GB SSD  
Price: \$279.99  
Manufacturer: Lenovo  
[Click "View Inventory" to see stock](#)  
[Add to Cart](#)




**Acer Chromebook 315**  
Acer Chromebook 315 - 15.6" HD Display Laptop - Intel Celeron - 4GB LPDDR4 - 64GB eMMC  
Price: \$149.00  
Manufacturer: Acer  
[Click "View Inventory" to see stock](#)  
[Add to Cart](#)




**HP Envy 2-in-1 14 Wide Ultra XGA Touch Screen Laptop**  
HP - Envy 2-in-1 14" Wide Ultra XGA Touch Screen Laptop - AMD Ryzen 7 - 16GB Memory - 1TB SSD  
Price: \$599.99  
Manufacturer: HP  
[Click "View Inventory" to see stock](#)




**Samsung Galaxy Book4**  
Samsung Galaxy Book4 - 15.6" FHD Laptop - Intel Core i7 - 16GB Memory - 512GB SSD  
Price: \$549.99  
Manufacturer: Samsung  
[Click "View Inventory" to see stock](#)  
[Add to Cart](#)



**Asus 14 Laptop**  
Asus 14" Laptop - Intel Celeron N4500 with 4GB Memory - 64GB eMMC  
Price: \$119.99  
Manufacturer: Asus  
[Click "View Inventory" to see stock](#)  
[Add to Cart](#)



**HP Envy Desktop**  
HP Envy Desktop - Intel Core i7 - 32GB Memory - NVIDIA GeForce RTX4060 - 1TB SSD  
Price: \$1439.99  
Manufacturer: HP  
[Click "View Inventory" to see stock](#)  
[Add to Cart](#)



**Dell Inspiron 27 Touch All In One Desktop**  
Dell Inspiron 27" Touch All In One Desktop - Intel Core i7 - 16GB Memory - 1TB SSD  
Price: \$1399.99  
Manufacturer: Dell  
[Click "View Inventory" to see stock](#)

[View Inventory](#)

Madison ▾

Once a customer adds their products to the cart they can simply click “View Cart” to go to the cart page. From here they can see the contents of the cart, enter their billing address and payment information and then click “Place Order”. It’s that easy!

## Your Cart



**Alienware Aurora R16 Desktop**

Price: \$1799.99



**CyberPowerPC Gaming Desktop**

Price: \$649.99

Total Price: \$2449.98  
Total with 7% Tax: \$2621.48  
[Back to Home](#)

## Billing Address

Name:  Address:  Phone:  Email:

## Payment Method

Select Payment Type:

## Card Information

Card Number:  Expiration Date:  Security Code (CVV):

Place Order

## CONCLUSION

I thoroughly enjoyed this project as it allowed me to invest a great deal of time into the main purpose of this class which was database management systems. It has been a while since I have taken a web development class since I finished my undergraduate degree back in 2009, so I had to brush up on that. I was already familiar with most of the front-end elements like HTML, JavaScript and CSS but had had to brush up on it some. The backend was more of the hurdle for me in connecting everything together. I ended up having to purchase several books to get familiar on where to use certain functions such as some of JavaScript's backend framework like Node.js and Express.js. Learning how to build app.js with Node.js with Express and learning how to use the REST APIs to communicate with the front end was monumental.

I started on the overall design of the initial customer facing website before anything and then moved over to MySQL. Discovering how to navigate and run MySQL both in the command line and MySQL Workbench seemed quite easy and enjoyable once I dove in. I added my tables in through the Workbench and then would see them in the Schemas panel and I was amazed when it all came together. Once I had finished most of my tables I struggled for a while trying to connect them together properly, but this was an enjoyable learning experience to overcome.

My biggest struggle that was also one of my most rewarding pieces of knowledge, was connecting the database through my code to my actual website. Again, I was trying to learn Node.js at this same time so I had to utilize several books to achieve this. I also made best friends with console error commands while also learning how to use an app called Postman API to see where I was running into errors after most of my `connection.query()` and `connection.execute()` commands. This also allowed me to learn what CRUD operations were in communicating with the MySQL database was. I had to change my code layout numerous times while copying and pasting the original one to a notepad and sometimes even wiping it and starting back over. Clicking the "Place Order" on my main page and getting it to populate information to the proper fields in the database often took the longest to achieve but was knowledge well spent.

The past couple of weeks have been spent just going back and double checking to make sure everything works properly and is connected properly. I still struggled with the code to properly run the queries needed for the assignment but once I got them right and it displayed the information I needed, it was almost like I was handed a briefcase with a million dollars in cash. My deep feeling of enjoyment and a deep sigh of relief from weeks upon weeks of hard work had finally paid off.

With that being said, it's hard to sum up in a couple of sentences what I actually learned from this assignment. When I first looked at this assignment, I was almost a little scared, almost like what did I get myself in to? After the completion of this project, I would tell anyone that is

thinking the same thing, this project is perfectly curated to give you the knowledge and experience you need for your future in real life situations. After doing this project, I feel comfortable very comfortable with database management systems and also website development to the extent that I would like to carry this project on further and even implement some of my own projects centered around relational databases.

## REFERENCES

Wexler, Jonathan. *Get Programming with Node.js*. First Edition. Manning Publications, 2019.

Brown, Ethan. *Web Development with Node and Express: Leveraging the JavaScript Stack*. 2<sup>nd</sup> Edition. O'Reilly Media, 2020.

DuRocher, David. *HTML and CSS QuickStart Guide: The Simplified Beginner's Guide to Developing a Strong Coding Foundation, Building Responsive Websites, and Mastering the Fundamentals of Modern Web Design*. January 22, 2021

Nixon, Robin. *Learning PHP, MySQL & JavaScript: A Step-by-Step Guide to Creating Dynamic Websites*. 6th Edition. O'Reilly Media.

Silberschatz, Abraham, Korth, Henry F., & Sudarshan, S. *Database System Concepts*. 7th Edition. McGraw-Hill Education.