Dylan Brooks
Eric Tran

Floatality Hut

URL: http://flip4.engr.oregonstate.edu:11385/products_in_sales

Executive Summary:

The project outline encapsulates the underlying drive behind the conception of a website-driven database tailored for a series of beachfront stores under the name “Floatality Hut.” Throughout the duration of the term, this outline has undergone iterative refinements, integrating both numerical and qualitative data to furnish the audience with a brief overview of the issue at hand and its corresponding solution, alongside an in-depth summary of the technical intricacies of the database's functionality.
Moreover, meticulous revisions have been made to enrich the database outline, encompassing comprehensive descriptions for each entity that defines their respective roles and purpose, while ensuring consistent naming conventions and applications of NULL and UNIQUE constraints. Furthermore, descriptions detailing the intricate network of relationships and interactions among entities have been included, along with provisions for intersection tables to facilitate many-to-many relationships.

Throughout the drafting processes, the entity-relationship diagram has undergone significant transformations. Initially, the ERD included a comprehensive visual of entity and intersection tables, each detailing primary keys, foreign keys, and assorted attributes. However, upon recognizing the disparities between an ERD and a schema, adjustments were made. The refactored ERD now exclusively features entity tables, each defined by its name and primary key, which are interconnected among each other via relationship and participation lines. This revised approach offers a panoramic depiction of the database relationships, contrasting with the more intricate visual representation provided by the schema. Furthermore, several attributes and data types underwent modifications to accommodate realistic input sizes and adhere to normal forms. For example, varchar sizes were expanded from 50 to 100, and primary keys were introduced to intersection tables to ensure compliance with the second normal form (2NF).

DDL went through several iterations, mostly focused on small tweaks like finding a properly unique identifier for locations (their address) and customers (their email address). There are things that can’t be taught, only learned: a person’s name is hardly unique. DMQ had fewer revisions as the queries there were merely rough drafts, and the final queries really took shape when developing the backend code in app.js. Queries were tested through the classmysql portal before being imported into app.js, and after testing, ported to DMQ.sql. The HTML pages drafted during Step 4 have been modified to utilize Node.js and the Handlebars framework to host the website on OSU’s flip servers and connect to the database. Additionally, a CSS file was added for unique styling and visual appeal. CRUD functionalities were successfully implemented for all entities with little to no backend bugs. Dropdown menus for entities with foreign keys, NULLable relationships and UPDATE functionality for M:N relationships, and code citations were implemented in the final product.

Project Outline:

Floatality Hut operates 5 beachfront stores which accrue a combined $1,500,000 net profit annually, each store averaging 750 monthly sale transactions and 1000 monthly customers. Specializing in the sales of knickknacks and towels, each product, although individually valued at no more than $80, possesses a unique touch, with designer towels standing out to be one-of-a-kind. The website-driven database is meticulously designed to track Products, Sales, Employees, and Customers for each of our beachfront Locations. This strategic approach will solve common problems local businesses consistently deal with, such as managing inventory space and storage in order to maximize profits. In other words, it will give insight regarding bestselling and least selling products, which would aid in the decision to increase or decrease inventory sizes for these specific products. Additionally, it will track all Sales made in each Location, as well as employee sales for performance reports on its 35 total employees.

Database Outline:

Locations:

- Purpose:
  - This table will store the information of the 5 beachfront stores operated by Floatality Hut. Each location is assigned a location ID, wares capacity, address, and phone number.

- Attributes and Constraints:
  - location_id: int, auto_increment, unique, not NULL, PK
  - wares_capacity: int, NOT NULL
  - address_line: varchar(100), NOT NULL
  - city: varchar(100), NOT NULL
  - postal_code: int, NOT NULL
  - site_phone: int, NOT NULL

- Relationships:
  - 1:M relationship between Locations and Sales.
    - A given Sale can happen at only one of the Locations, and one Location can have one or more Sales.
    - This is implemented by storing location_id as a FK inside Sales.
  - M:N relationship between Locations and Employees.
    - Each location will consist of at least 1 employed employee, while an Employee may work at multiple Locations (depending on their job).

        - This is implemented by an intersection table Location_has_Employees to facilitate M:N, storing location_id and employee_id FK pairs.
    - M:N relationship between Locations and Products.
        - Each location hosts several Products, and each Product can be found at multiple Locations.
        - This is implemented by an intersection table holding product_id and location_id FK pairs.

Customers:

- Purpose:
    - This table will store the information of the customers who have made at least one purchase in one of Floataility Hut's stores. Each customer is assigned a customer ID. The table will record the customer's contact information, store credit, and total purchases.

- Attributes and Constraints:
    - customer_id: int, auto_increment, unique, NOT NULL, PK
    - customer_name: varchar(100), NULL
    - email: varchar(100), unique, NULL
    - customer_phone: int, NULL
    - store_credit: int, NOT NULL
        - Initialize to 0
    - total_purchases: int, NOT NULL

- Relationships:
    - 1:M relationship between Customers and Sales.
        - Customers can indulge in multiple Sales while a Sale is only associated with a single Customer.
        - This is implemented by storing customer_id as a FK inside Sales.

Sales:

- Purpose:
    - The Sales table will track transactions and report on the customer, employee, location, and sale date. Each sale will be assigned a sale ID and will be related to Products via an intersection table.

- Attributes and Constraints:
    - sale_id: int, auto_increment, unique, NOT NULL, PK
    - customer_id: int, NOT NULL, FK

    - employee_id: int, NOT NULL, FK
    - location_id: int, NOT NULL, FK
    - sale_date: datetime, NOT NULL

- Relationships:
    - M:1 relationship between Sales and Locations.
        - A given Sale can happen at only one of the Locations, and one Location can have one or more Sales.
        - This is implemented by storing location_id as a FK inside Sales.
    - M:1 relationship between Sales and Customers.
        - Customers can indulge in multiple Sales while a Sale is only associated with a single Customer.
        - This is implemented by storing customer_id as a FK inside Sales
    - M:N relationship between Sales and Products.
        - Each Sale will include at least one Product, and a Product may appear one or more multiple sales.
        - This is implemented using an intersection table holding product_id and sale_id FK pairs.
            - An additional attribute “quantity” will be included to represent the total quantity of a specific product being purchased.
    - M:1 relationship between Sales and Employees.
        - Only one Employee can authorize a Sale, and multiple Sales can be authorized by the same Employee.
        - This is implemented by storing employee_id as a FK in Sales. This represents which employee processed the transaction.

Products:

- Purpose:
    - Products refer to types of products being sold in various Locations for a price, some of which have an artist or designer associated with them. For instance, a towel would be a Product.

- Attributes and Constraints:
    - product_id: int, auto_increment, unique, NOT NULL, PK
    - price: DECIMAL(10,2), NOT NULL
    - label: varchar(200), NOT NULL
    - designer: varchar, NULL

- Relationships:
    - M:N relationship between Locations and Products.

        - Each location hosts several Products, and each Product can be found at multiple Locations.
        - This is implemented by an intersection table holding product_id and location_id FK pairs.
    - M:N relationship between Sales and Products.
        - Each Sale will include at least one Product, and a Product may appear one or more multiple sales.
        - This is implemented using an intersection table holding product_id and sale_id FK pairs.
            - An additional attribute “quantity” will be included to represent the total quantity of a specific product being purchased.

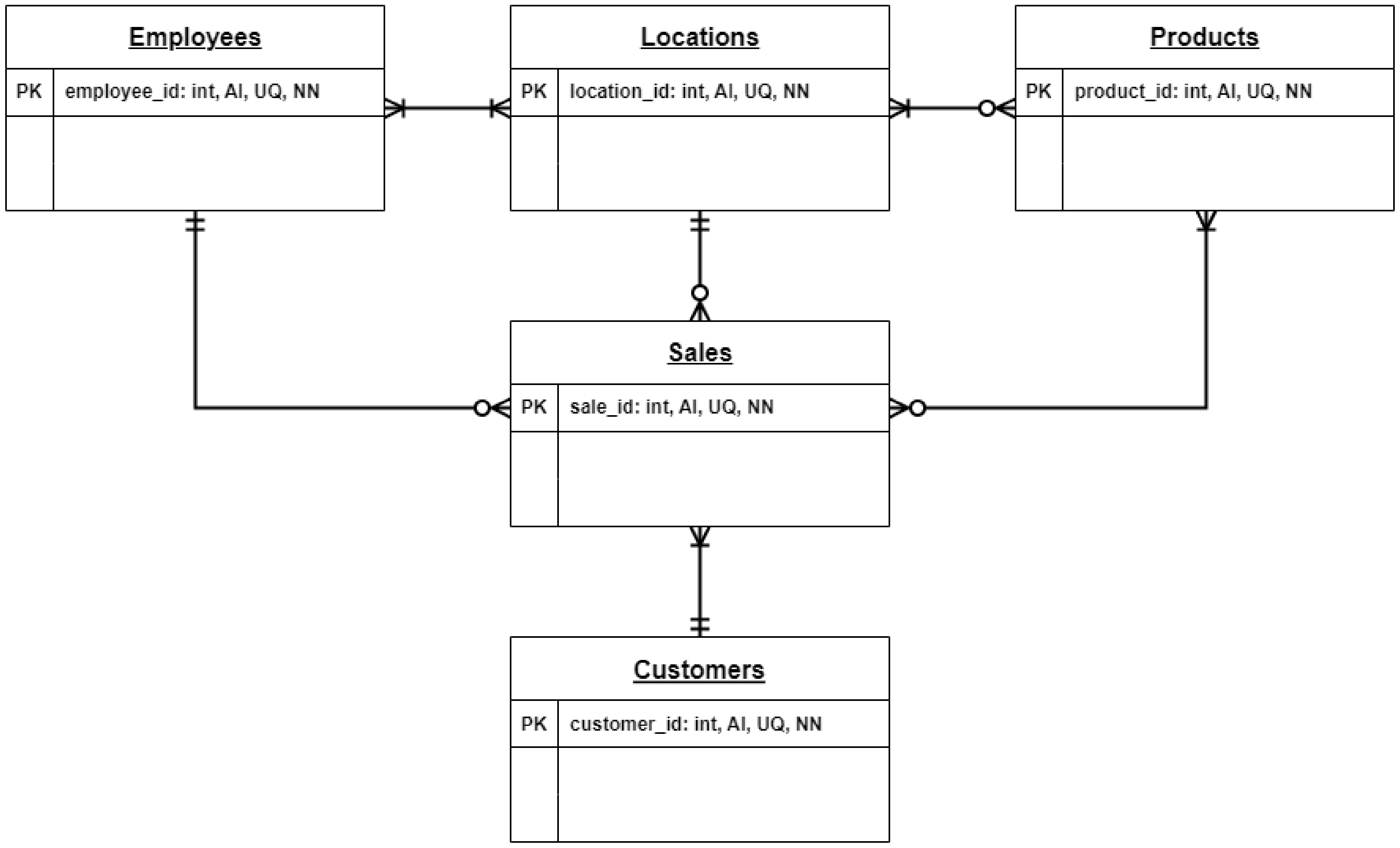
Employees:

- Purpose:
    - The Employees table stores each employee individually. Each employee is assigned an employee ID. The employee’s name and contact information is recorded.

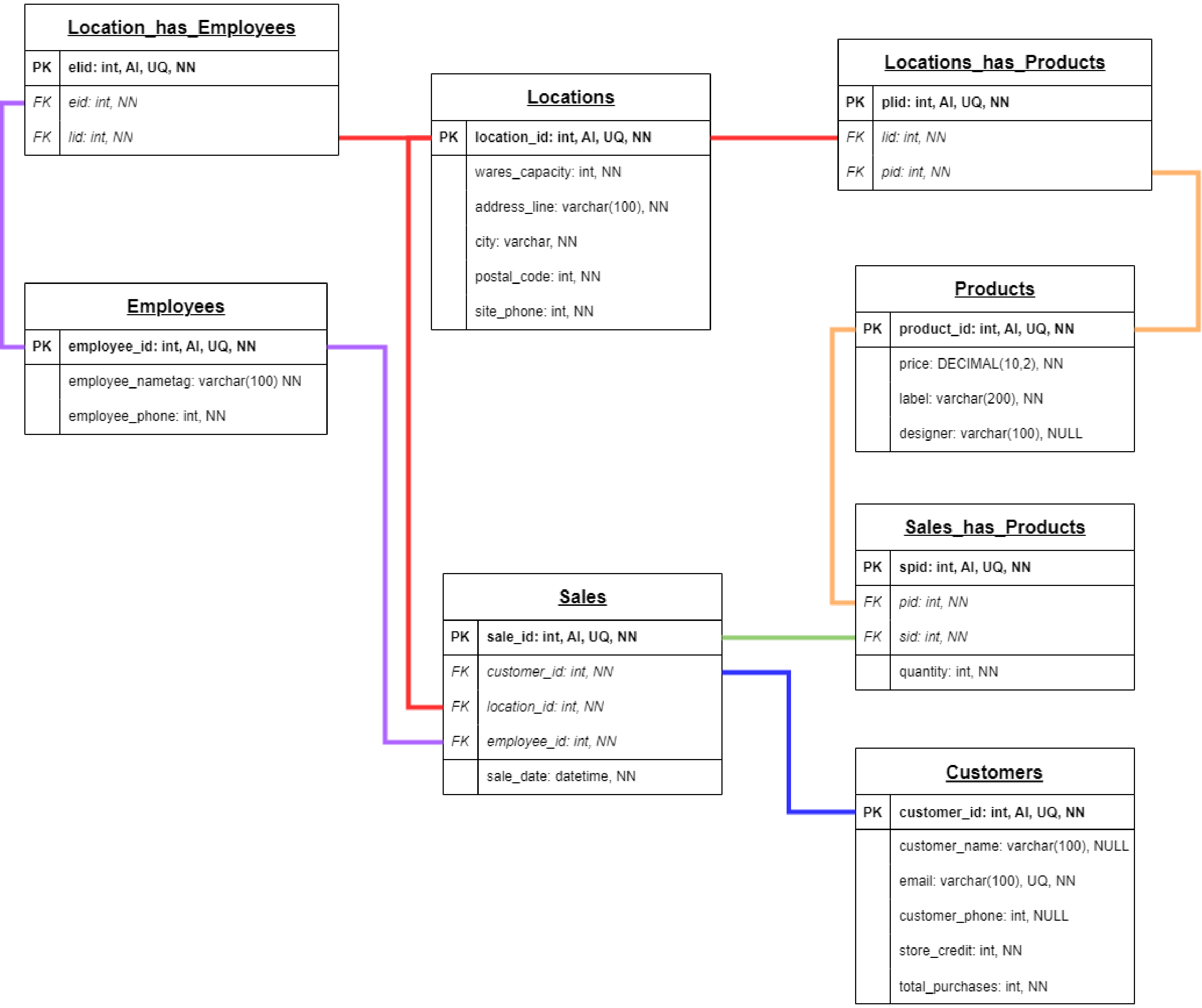
- Attributes and Constraints:
    - employee_id: int, auto_increment, unique, NOT NULL, PK
    - employee_nametag: varchar, NOT NULL
        - Expected format: “Jane Doe (she/them)”
    - employee_phone: int, NOT NULL

- Relationships:
    - 1:M relationship between Employees and Sales.
        - Only one Employee can authorize a Sale, and multiple Sales can be authorized by the same Employee.
        - This is implemented by storing employee_id as a FK in Sales. This represents which employee processed the transaction.
    - M:N relationship between Employees and Locations.
        - Each location will consist of at least 1 employed employee, while an Employee may work at multiple Locations (depending on their job).
        - This is implemented by an intersection table Location_has_Employees to facilitate M:N, storing location_id and employee_id FK pairs.

Entity-Relationship Diagram:


Employees
PK
employee_id: int, AI, UQ, NN
Locations
PK
location_id: int, AI, UQ, NN
Products
PK
product_id: int, AI, UQ, NN
Sales
PK
sale_id: int, AI, UQ, NN
Customers
PK
customer_id: int, AI, UQ, NN

Schema:



Location_has_Employees

PK	**elid: int, AI, UQ, NN**
FK	*eid: int, NN*
FK	*lid: int, NN*

Locations

PK	**location_id: int, AI, UQ, NN**
	wares_capacity: int, NN
	address_line: varchar(100), NN
	city: varchar, NN
	postal_code: int, NN
	site_phone: int, NN

Locations_has_Products

PK	**plid: int, AI, UQ, NN**
FK	*lid: int, NN*
FK	*pid: int, NN*

Employees

PK	**employee_id: int, AI, UQ, NN**
	employee_nametag: varchar(100) NN
	employee_phone: int, NN

Products

PK	**product_id: int, AI, UQ, NN**
	price: DECIMAL(10,2), NN
	label: varchar(200), NN
	designer: varchar(100), NULL

Sales_has_Products

PK	**spid: int, AI, UQ, NN**
FK	*pid: int, NN*
FK	*sid: int, NN*
	quantity: int, NN

Sales

PK	**sale_id: int, AI, UQ, NN**
FK	*customer_id: int, NN*
FK	*location_id: int, NN*
FK	*employee_id: int, NN*
	sale_date: datetime, NN

Customers

PK	**customer_id: int, AI, UQ, NN**
	customer_name: varchar(100), NULL
	email: varchar(100), UQ, NN
	customer_phone: int, NULL
	store_credit: int, NN
	total_purchases: int, NN

Example Data:

Locations

```
+-------------+----------------+----------------------+--------------+-------------+------------+
| location_id | wares_capacity | address_line         | city         | postal_code | site_phone |
+-------------+----------------+----------------------+--------------+-------------+------------+
|           1 |            100 |       810 SW 10th St |  Lincoln City |      97367 | 5419948888 |
|           2 |             45 |   56771 Breakers Blvd |     Neskowin  |      97149 | 5413302954 |
|           3 |             30 |    48405 Corvallis Ave |     Neskowin |      97149 | 5419691221 |
|           4 |             70 | 1234 Beach Loop Rd   |       Bandon |      97411 | 5413473770 |
+-------------+----------------+----------------------+--------------+-------------+------------+
```

Products

```
+------------+---------+------------------------------+----------------+
| product_id |  price  |            label             |    designer    |
+------------+---------+------------------------------+----------------+
|         11 |  589.99 |  Summer Catalogue Swimwear   |  Louis Vuitton |
|         91 |   54.03 |           20x10 Beach Towel  |  Anthropologie |
|        222 |   33.75 |               Dream Catcher  |           NULL |
|        606 |   85.00 |  Designer Towel Blue on Black  |    Dusen Dusen |
+------------+---------+------------------------------+----------------+
```

Employees

```
+-------------+-----------------------------+----------------+
| employee_id | employee_nametag            | employee_phone |
+-------------+-----------------------------+----------------+
|           3 |        Janet Brown (she/her) |     5419795992 |
|           4 | Donovan Gresham (he/him)    |     5415952108 |
|           6 | Malenia Mondut (She/them)   |     5410882431 |
|          10 |  Johnathan Joestar (he/him) |     9019938247 |
+-------------+-----------------------------+----------------+
```

Customers

```
+-------------+------------------+----------------------------+----------------+--------------+-----------------+
| customer_id | customer_name    | email                      | customer_phone | store_credit | total_purchases |
+-------------+------------------+----------------------------+----------------+--------------+-----------------+
|           1 | Eddie Franklin   | eddiethereaddy87@hotmail.com |   5418704112 |            0 |               1 |
|           2 |             NULL | boomhauermd41@gmail.com    |           NULL |           10 |               1 |
|           3 |             NULL | chasemiyagi@yahoo.com      |           NULL |            0 |               1 |
|           4 |             NULL | ygmarkosinc@gmail.com      |     5413078699 |          150 |               3 |
+-------------+------------------+----------------------------+----------------+--------------+-----------------+
```

Sales

```
+---------+-------------+-------------+-------------+---------------------+
| sale_id | location_id | employee_id | customer_id | sale_date           |
+---------+-------------+-------------+-------------+---------------------+
|    4032 |           3 |           3 |           4 | 2024-02-01 11:30:22 |
|    5555 |           2 |           6 |           3 | 2024-01-06 18:01:59 |
|    8765 |           1 |          10 |           1 | 2024-01-06 07:44:06 |
|   10114 |           1 |           3 |           1 | 2024-01-01 07:27:29 |
+---------+-------------+-------------+-------------+---------------------+
```

Location has Employees

```
+-----------------------+-------------+-------------+
| location_employees_id | location_id | employee_id |
+-----------------------+-------------+-------------+
|                     1 |           1 |           3 |
|                     2 |           1 |          10 |
|                     3 |           2 |           6 |
|                     4 |           3 |           3 |
+-----------------------+-------------+-------------+
```

Location has Products

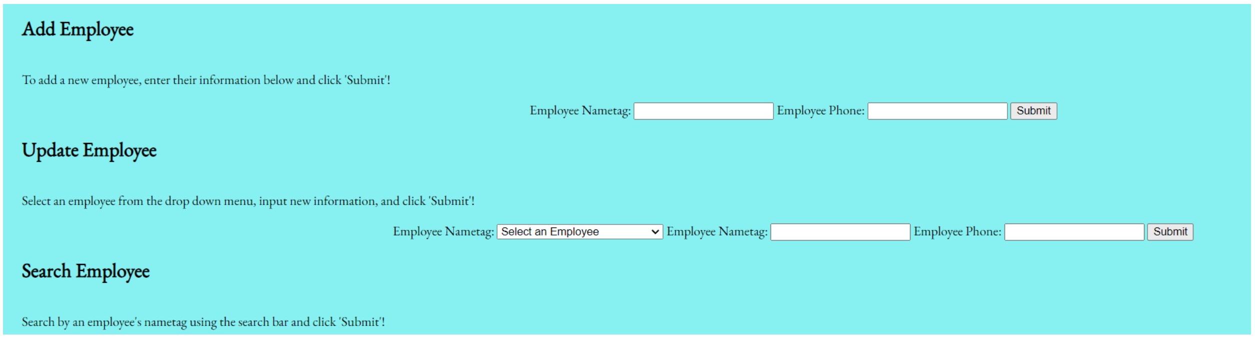
```
+----------------------+-------------+------------+
| location_products_id | location_id | product_id |
+----------------------+-------------+------------+
|                    1 |           1 |         11 |
|                    2 |           2 |         11 |
|                    3 |           2 |        606 |
|                    4 |           3 |         91 |
+----------------------+-------------+------------+
```

Sale has Products

```
+------------------------+-------------+-----------------+-------------+
| sale_products_id  |    sale_id  |   product_id  |   quantity |
+------------------------+-------------+-----------------+-------------+
|                            1 |      10114 |               11 |              2 |
|                            2 |        8765 |             606 |              1 |
|                            3 |      10114 |               91 |              1 |
|                            4 |        5555 |             222 |            21 |
+------------------------+-------------+-----------------+-------------+
```

UI Screenshots:

ADD, UPDATE, and SEARCH Employees



Add Employee

To add a new employee, enter their information below and click 'Submit'!

Employee Nametag: ______ Employee Phone: ______ Submit

Update Employee

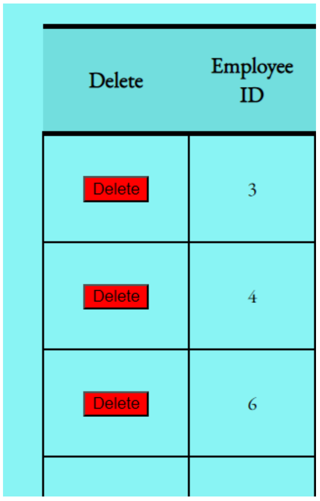
Select an employee from the drop down menu, input new information, and click 'Submit'!

Employee Nametag: Select an Employee Employee Nametag: ______ Employee Phone: ______ Submit

Search Employee

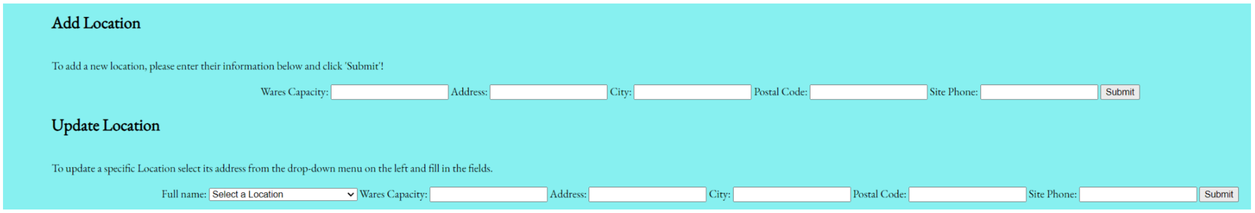
Search by an employee's nametag using the search bar and click 'Submit'!

DELETE Employees



Delete	Employee ID
Delete	3
Delete	4
Delete	6

ADD and UPDATE Locations


Add Location

To add a new location, please enter their information below and click 'Submit'!

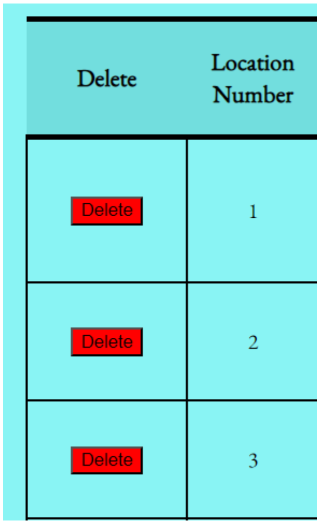
Wares Capacity: Address: City: Postal Code: Site Phone: Submit

Update Location

To update a specific Location select its address from the drop-down menu on the left and fill in the fields.

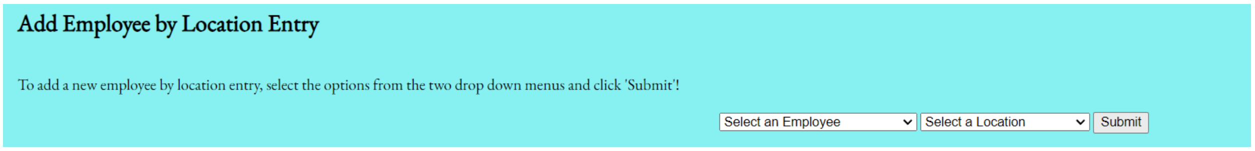
Full name: Select a Location Wares Capacity: Address: City: Postal Code: Site Phone: Submit

DELETE Locations



Delete	Location Number
Delete	1
Delete	2
Delete	3

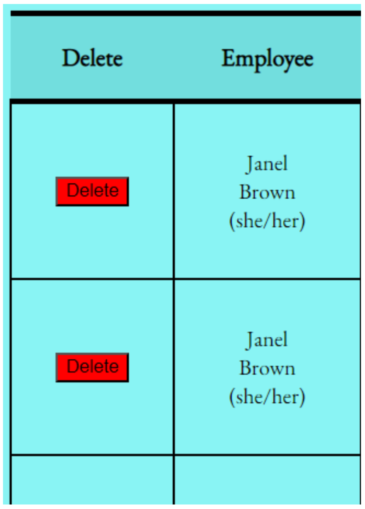
ADD Employees by Location


Add Employee by Location Entry

To add a new employee by location entry, select the options from the two drop down menus and click 'Submit'!

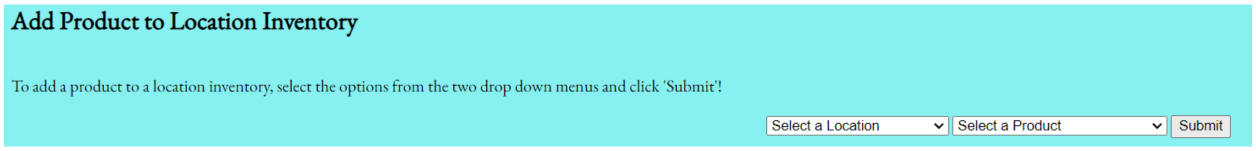
Select an Employee Select a Location Submit

DELETE Employees by Location



Delete	Employee
Delete	Janel Brown (she/her)
Delete	Janel Brown (she/her)

ADD Product by Location

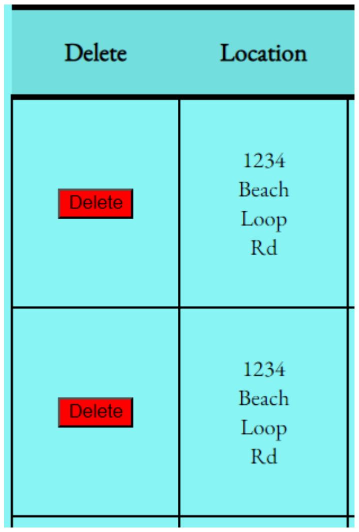


Add Product to Location Inventory

To add a product to a location inventory, select the options from the two drop down menus and click 'Submit'!

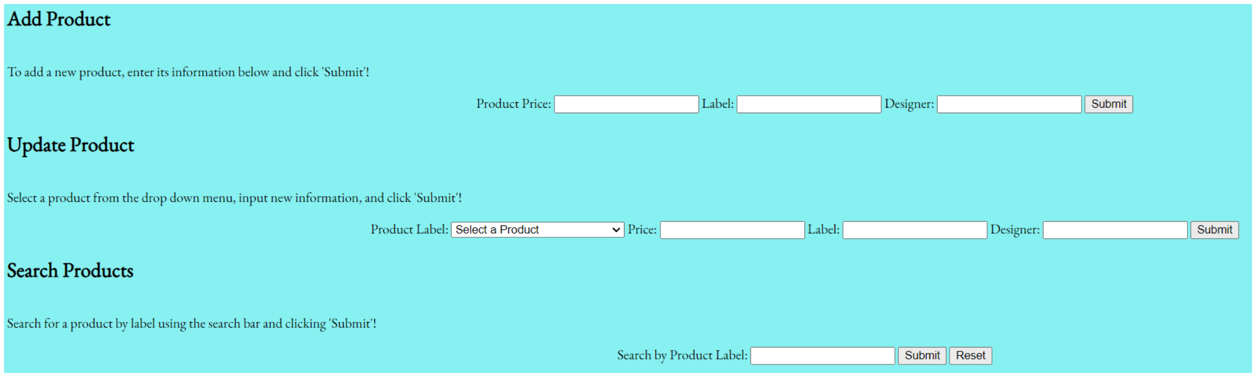
Select a Location | Select a Product | Submit

DELETE Product by Location



Delete	Location
Delete	1234 Beach Loop Rd
Delete	1234 Beach Loop Rd

ADD, UPDATE, and SEARCH Products



Add Product

To add a new product, enter its information below and click 'Submit'!

Product Price: ______ Label: ______ Designer: ______ Submit

Update Product

Select a product from the drop down menu, input new information, and click 'Submit'!

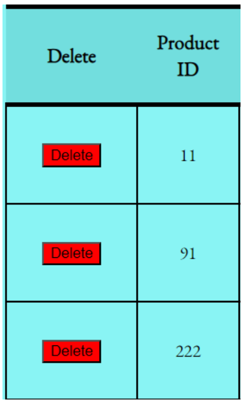
Product Label: Select a Product Price: ______ Label: ______ Designer: ______ Submit

Search Products

Search for a product by label using the search bar and clicking 'Submit'!

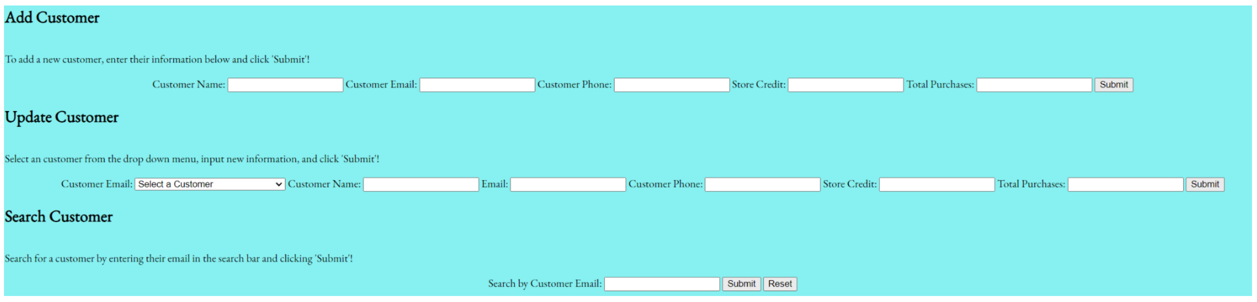
Search by Product Label: ______ Submit Reset

DELETE Products



Delete	Product ID
Delete	11
Delete	91
Delete	222

ADD, UPDATE, and SEARCH Customers



Add Customer

To add a new customer, enter their information below and click 'Submit'!

Customer Name: Customer Email: Customer Phone: Store Credit: Total Purchases: Submit

Update Customer

Select an customer from the drop down menu, input new information, and click 'Submit'!

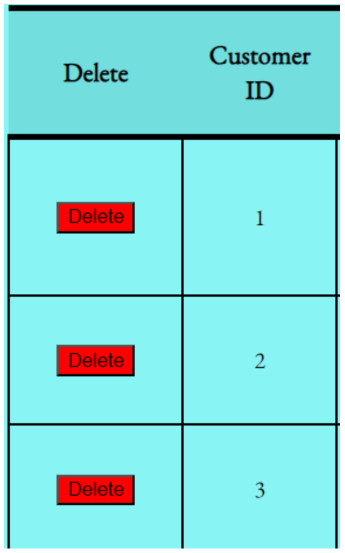
Customer Email: Select a Customer Customer Name: Email: Customer Phone: Store Credit: Total Purchases: Submit

Search Customer

Search for a customer by entering their email in the search bar and clicking 'Submit'!

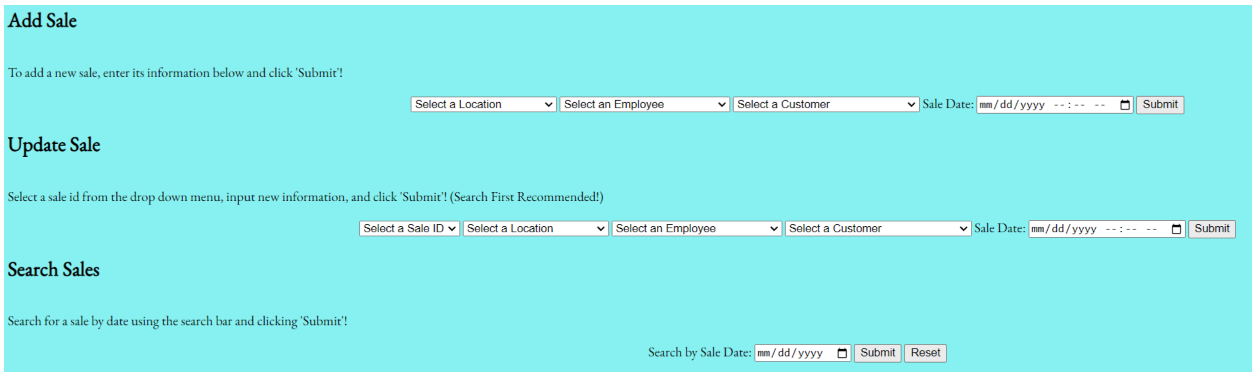
Search by Customer Email: Submit Reset

DELETE Customers



Delete	Customer ID
Delete	1
Delete	2
Delete	3

ADD, UPDATE, and SEARCH Sales



Add Sale

To add a new sale, enter its information below and click 'Submit'!

Select a Location Select an Employee Select a Customer Sale Date: mm/dd/yyyy --:-- -- Submit

Update Sale

Select a sale id from the drop down menu, input new information, and click 'Submit'! (Search First Recommended!)

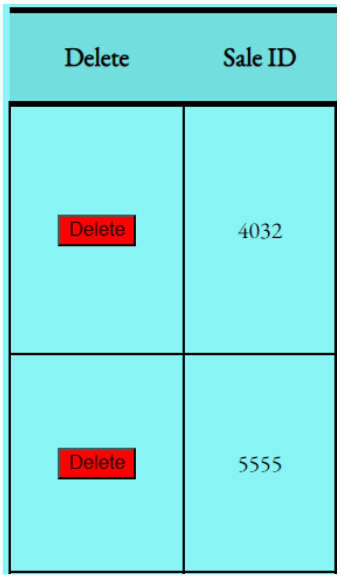
Select a Sale ID Select a Location Select an Employee Select a Customer Sale Date: mm/dd/yyyy --:-- -- Submit

Search Sales

Search for a sale by date using the search bar and clicking 'Submit'!

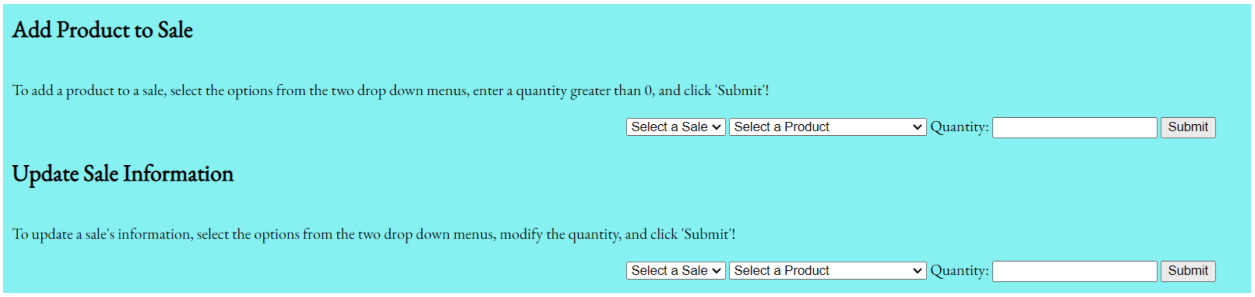
Search by Sale Date: mm/dd/yyyy Submit Reset

DELETE Sales



Delete	Sale ID
Delete	4032
Delete	5555

ADD and UPDATE Products in Sales



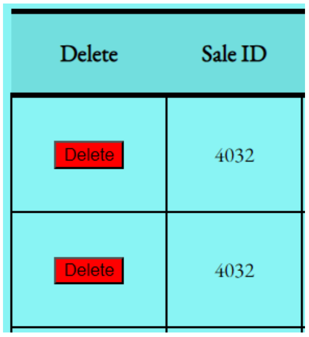
Add Product to Sale

To add a product to a sale, select the options from the two drop down menus, enter a quantity greater than 0, and click 'Submit'!

Update Sale Information

To update a sale's information, select the options from the two drop down menus, modify the quantity, and click 'Submit'!

DELETE Products in Sales



Delete	Sale ID
Delete	4032
Delete	4032