# Executive Summary
# Group 44
**Daniel Jones**
**Cameron O'Brien**

**URL TO WEBSITE: http://flip3.engr.oregonstate.edu:7777/**

A decent amount of things were changed from the beginning of the project until the end. One of the main things that we changed around was the data types for each of the attributes. Originally we weren't very familiar with the different data types allowed, but once we learned more, we had a lot of text or varchar data types, and the date data type as well. We also originally had all of the attributes as input fields when adding a new one, but later on changed it so the user does not need to add the ID for each row in the table, as that is auto-incrementing so it would make sense that it does not need to be manually added.

We also changed some attributes that were originally NOT NULL to be nullable, as it made more sense this way, for things like dateDelivered, as an order will be placed into that table when an order is placed, but there will only be a value there when the order ends up getting delivered. We also did not end up implementing the Store_Products table as we once intended because we feel it did not really add value, and we wanted to put more time into the other tables to make sure those operations were working well before we added too much more.

We changed the daysOpen from being a string of characters for which days the store was open EX) MWTWRS to Monday-Saturday. This made it a lot more simple, and since it is just a string, it was easy to allow this data type within the front end form and then query with a varchar(255) data type to insert this into the backend database.

The On Sale column changed from text saying "Yes" or "No" to a tinyint/bool, so it's a 1 if the product is on sale, and a 0 if the product is not on sale. Since this application will be used by the company, it does not need to be any more explanatory than that, as a simple 1 or 0 makes enough sense.

We took away the revenue column from the stores table because we feel it would just be a made up number from us anyways, so if we were to build upon this web application in the future, we could add it in to accurately keep track of the revenue from each of the stores, if we create more stores in the future.

We changed our Data Definition queries to work better, as before there were some issues since there was commenting within the file.

We made the UI easier to follow and simpler by changing the buttons and updating the Foreign Keys to what they should be

We changed a major bug which was displaying our customerID in our Order_Products table instead of the Product ID, which makes sense for our M:M relationship as this is a join table.

**LAX4U Proposal and Outline**
**By**
**Daniel Jones**
**Cameron O'Brien**

**URL TO WEBSITE: http://flip3.engr.oregonstate.edu:7777/**

****IMPORTANT****

To run the DDL queries in PHPMyAdmin, Deselect the "Enable Foreign Key Checks" before entering the query file as there are dependencies that don't allow the whole file to be run at once.

## Overview

LAX4U sells $8 million worth of lacrosse equipment and clothing annually. Our database driven website will help keep track of various things, included but not limited to: customer accounts (Customers entity), sales (Orders entity), Products (products entity) stores (stores entity), and more. By having a DB driven backend to our website, we will be able to make operations much more efficient by keeping track of everything in an organized manner that is easy to understand and will scale for a large volume of customers. We average around 200 customers per day in our retail location. We have 1 store currently and are hoping to expand more in the future. The DB backend will record our sales via the Orders entity and allow employees to easily look up customers, to find their account, change their info, and see all past orders from them.

The 5 tables that we are going to implement in our project are: Customers, Orders, Stores, and Products and Order_Products. Orders and Products have a M:M relationship that will be connected via the intersection table named Order_Products. We will have a total of 5 tables in our database. Cameron O'Brien will be responsible for the Customers and Orders tables, and Daniel Jones will be responsible for the Orders, Products, and Order_Products tables when it comes to code for the webpage.

## Outline for each table CRUD:

### Products

- INSERT by clicking on the ADD products button within the Products tab
- UPDATE by clicking on the edit button on the desired row to update
- DELETE by clicking on the delete button of desired row

### Orders

- INSERT by clicking on the Add Orders button within the Orders tab
- UPDATE by clicking on the edit button in whichever row you want
- DELETE by clicking on the delete button of desired row

### Customers

- INSERT a new customer by clicking on the Add Customer button within the Customers tab.
- UPDATE by clicking on the edit button in the row you want
- DELETE by clicking on the delete button of desired row

### Stores

- INSERT a new store by clicking the Add Store button within the Stores tab
- UPDATE a store information by clicking on the update button in the row of that store
- DELETE by clicking on the delete button of desired row
- Currently just have the one store, but it's set up for expansion in the future

### Order_Products

- This table represents the intersection table between Orders and Products
- UPDATE by pressing the update button in a given row.
- DELETE by clicking on the delete button of desired row
- SELECT by searching for an order ID in the search bar to bring that row up

Database Outline

- **Customers**: This holds information of each customer that buys from our website.
    - customerID*: int, unique, not NULL, auto-increment, Primary Key
    - email: varchar(255), not NULL
    - firstName: varchar(255), not NULL
    - lastName: varchar(255), not NULL
    - address: varchar(255), not NULL
    - dob: date
    - phone: varchar(255)
    - city: varchar(55)
    - state: varchar(55)
    - zipcode: varchar(9)
    - Relationship: A 1:M relationship between Customers and Orders, the customerID attribute is a FK inside Orders.

  - Purpose: The purpose of the Customers entity is to keep information stored about each customer which helps with things like returns, reordering past orders, rewards programs, etc.

- **Orders**: This table will hold the orders that are placed by customers. Used with the Join Table named Product_Orders, queries can be made to lookup specific orders by customerID or orderID to see the details from this table.
  - orderID*: int, not NULL, unique, auto-increment, PK
  - customerID*: int, unique, FK
  - productID*: int, not NULL, unique, FK
  - dateOrdered: DATE, not NULL
  - dateDelivered: DATE (Changed to NULLable since orders will be in system even if an item is not delivered yet.
  - totalPrice: decimal(10,2), not NULL
  - Purpose: Keep track of each order that is placed by a customer, and will be looked at if there are any issues with getting the products to the customer.
  - Relationship: There is a 1:M relationship between Customers and Orders. Each order can only be made by one customer (an order with the same items is still considered a separate order). A 1:M relationship between Orders and Stores. As a specific order can only be placed at one store, but a store will have many orders placed at it. (There is also a M:M relationship with Products. Which means there's a 1:M relationship between Orders and the Order_Products Join Table.

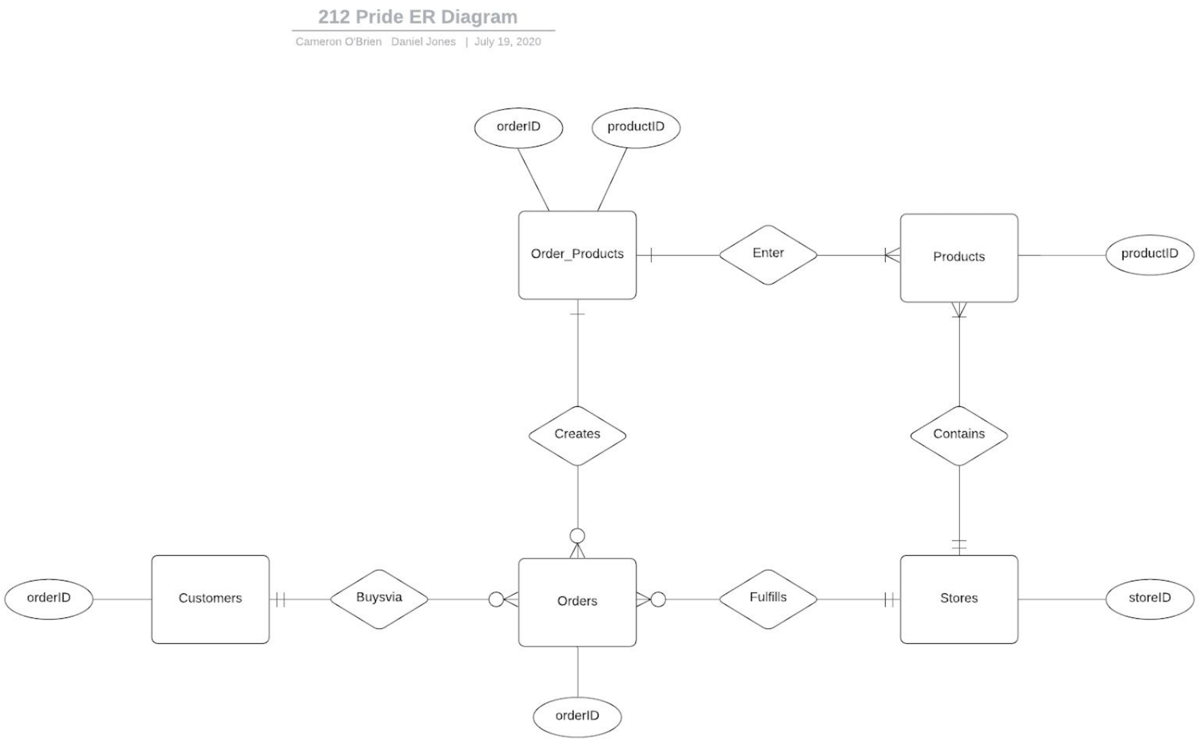
- **Products**: This is the table for attributes about each item that the store is selling.
  - productID*: int, not NULL, unique, auto-increment, PK
  - productName: varchar(255), not NULL
  - brandName: varchar(255), not NULL
  - price: decimal(10,2), not NULL
  - category: varchar(255)
  - sale: bool, not NULL
  - color: varchar(255), not NULL
  - Purpose: This table holds the specifications for each item the store sells. This had the product details such as ID, name, color, brand, and the price it is selling for. This will allow each order to contain this information.
  - Relationship: Products has a M:M relationship with Orders. This is because a single order can have many different products, and a single product can be in many different orders. We have a Join Table named Order_Products that is related to both Orders, and Products to make this relationship work. Therefore the direct relationship between Products and Order_Products is 1:M.

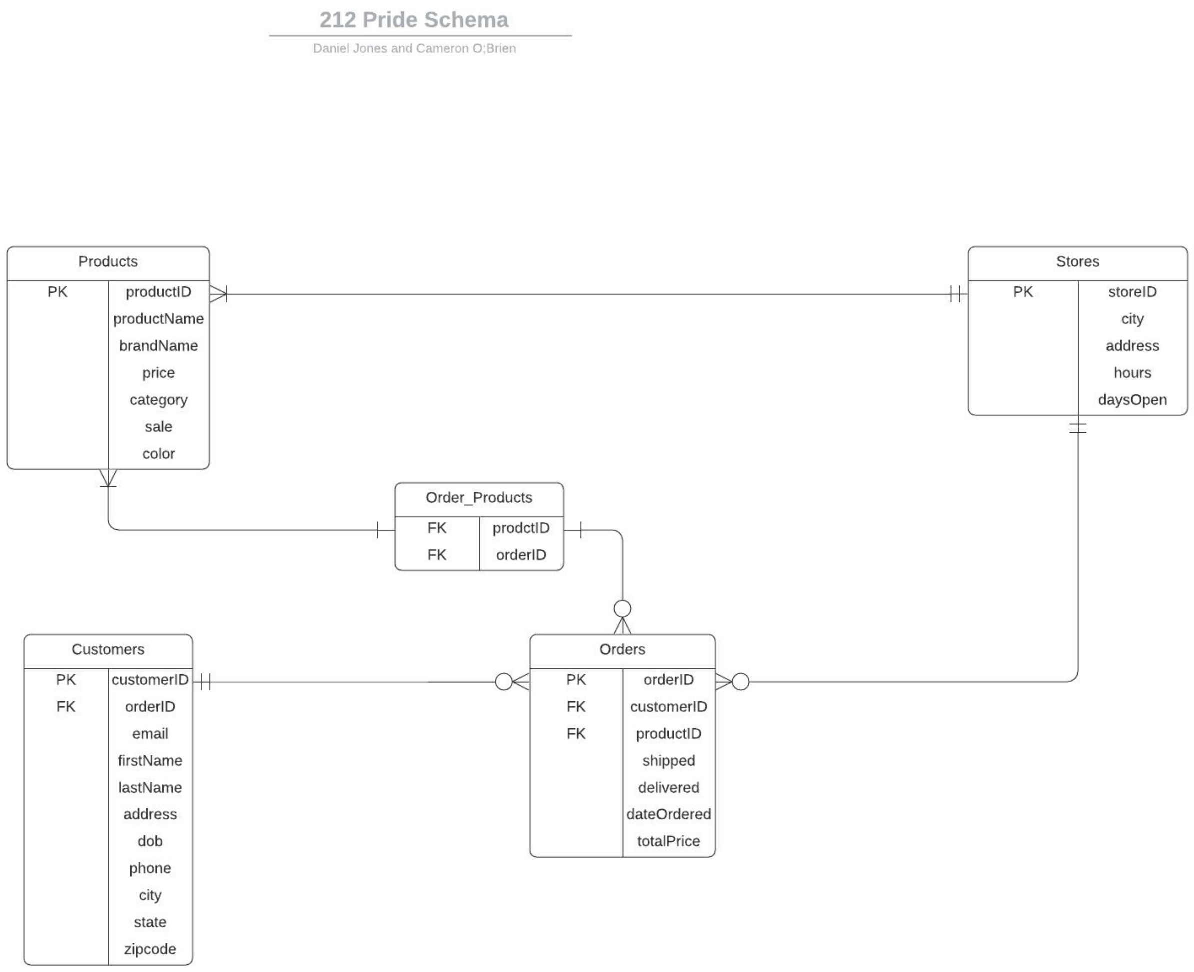
- **Order_Products**: This is a Join Table for the relationship between Orders and Products.
    - productID*: int, not NULL, unique, FK
    - orderID*: int, not NULL, unique, FK
    - Purpose: A Join Table to act as the intersection between Orders and Products to allow efficiency for the M:M relationship.
    - Relationship: Has a 1:M relationship with Orders, as well as Products.

- **Stores**: This is the table for the attributes regarding our store
    - storeID*: int, unique, not NULL, auto-increment, PK
    - city: varchar(255), not NULL
    - State: varchar(255), not NULL
    - address: varchar(255), not NULL
    - hours: varchar(255) (EX: 10-6)
    - daysOpen: varchar(255) (EX: Monday-Sunday)
    - Purpose: To keep track of the information regarding each store, so we can change our public website in the future if hours change, or even reconfigure the set up if we build new stores in the future.
    - Relationship: A 1:M relationship with Products as 1 store will have many products, and any product will only belong to one store, since we currently only have the single store.

- **Store_Products**: This is the Join Table for the information regarding which products we currently have in stock at different stores (whether it's in person or online)
    - productID*: int, not NULL, unique, FK
    - storeID*: int, unique, not NULL, FK
    - Quantity: int
    - Purpose: To keep track of the number of a certain item left at any given store. This will make reordering products easier and giving customers information on how to purchase the object if it is out of stock at a single store.
    - Relationship: There is a 1:M relationship between this table and Stores, and a 1:M between this table and Products. This will allow for the M:M relationship between Stores and Products to exist.
    - ***IMPORTANT*** - We are not implementing this table anyways, but it would only be used if we create more stores in the future, as that would shift to Stores and Products having a M:M relationship.

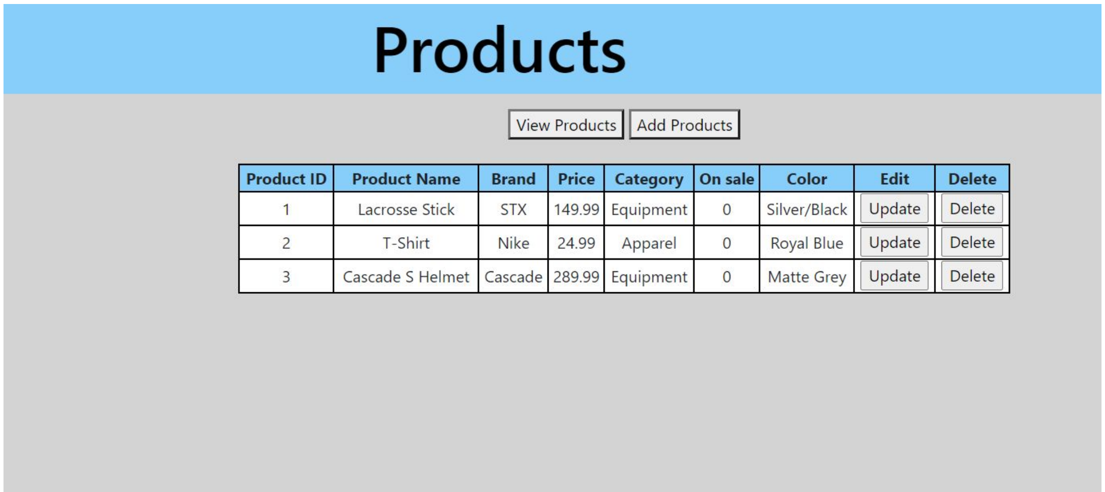
ERD Diagram here





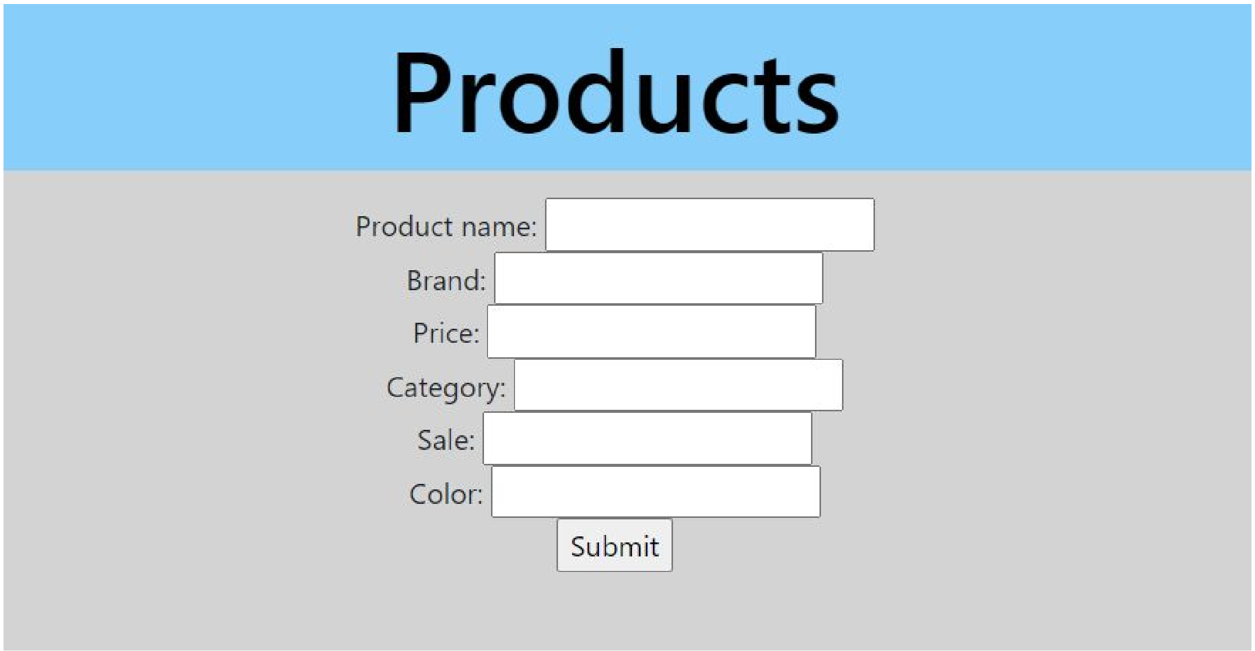
## Screenshots of the UI:

## READ/BROWSE/DISPLAY Products Page



| Product ID | Product Name | Brand | Price | Category | On sale | Color | Edit | Delete |
|---|---|---|---|---|---|---|---|---|
| 1 | Lacrosse Stick | STX | 149.99 | Equipment | 0 | Silver/Black | Update | Delete |
| 2 | T-Shirt | Nike | 24.99 | Apparel | 0 | Royal Blue | Update | Delete |
| 3 | Cascade S Helmet | Cascade | 289.99 | Equipment | 0 | Matte Grey | Update | Delete |

**CREATE/INSERT/ADD NEW Products page**



Products

Product name:

Brand:

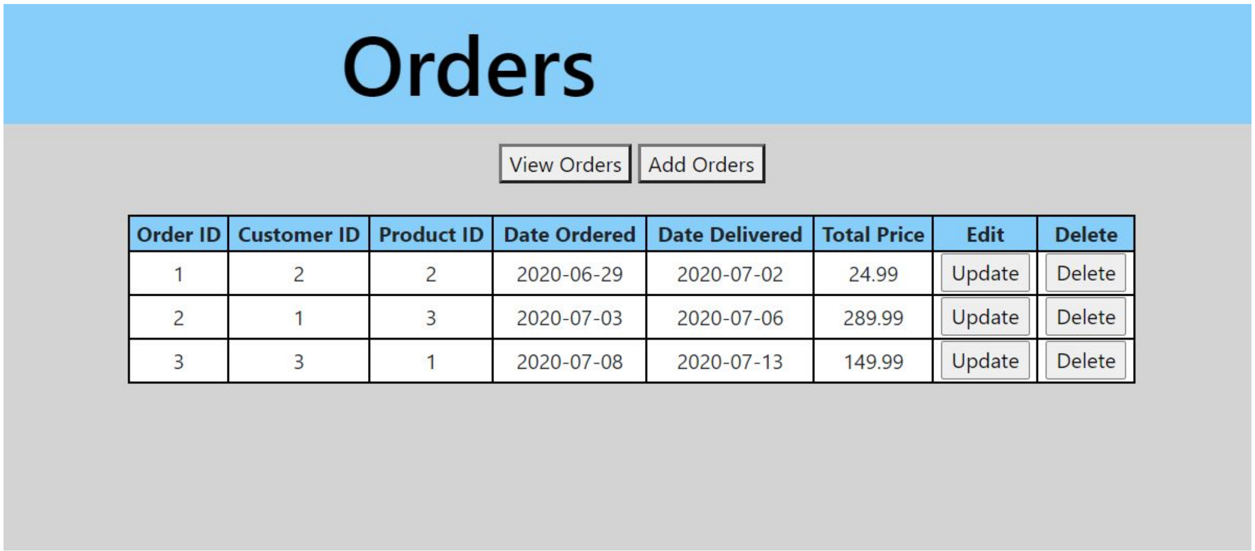
Price:

Category:

Sale:

Color:

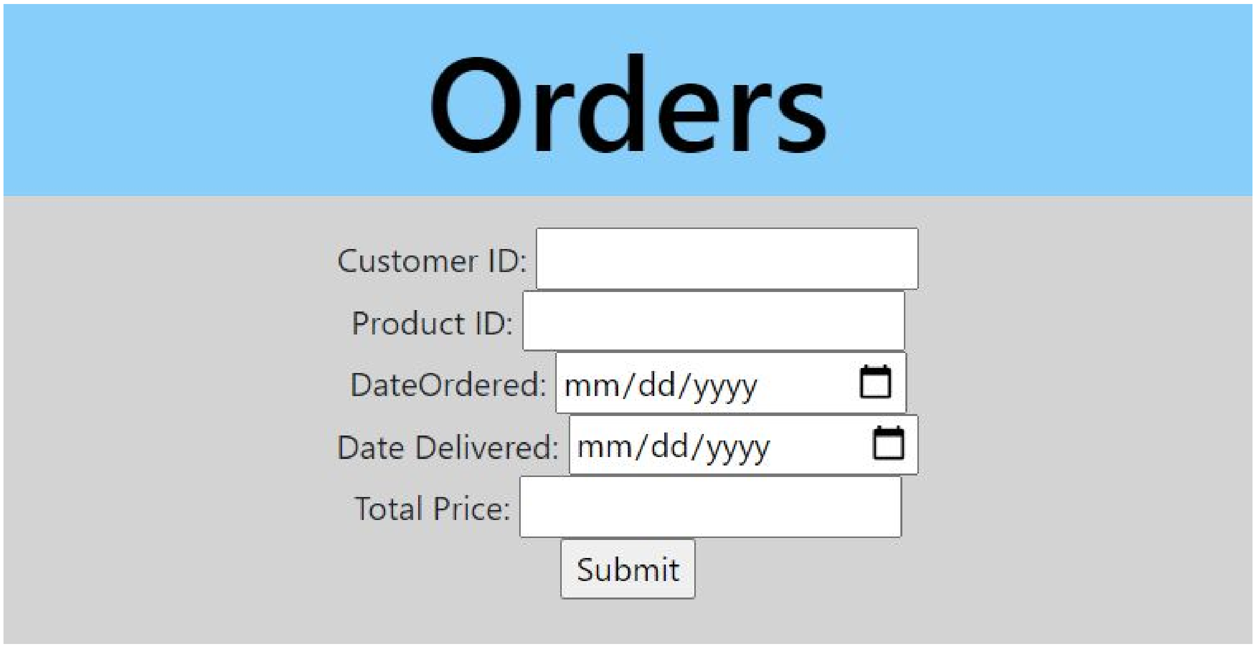
**READ/BROWSE/DISPLAY Orders Page**



Orders

| Order ID | Customer ID | Product ID | Date Ordered | Date Delivered | Total Price | Edit | Delete |
|---|---|---|---|---|---|---|---|
| 1 | 2 | 2 | 2020-06-29 | 2020-07-02 | 24.99 | Update | Delete |
| 2 | 1 | 3 | 2020-07-03 | 2020-07-06 | 289.99 | Update | Delete |
| 3 | 3 | 1 | 2020-07-08 | 2020-07-13 | 149.99 | Update | Delete |

**CREATE/INSERT/ADD NEW Orders page**



Orders

Customer ID:

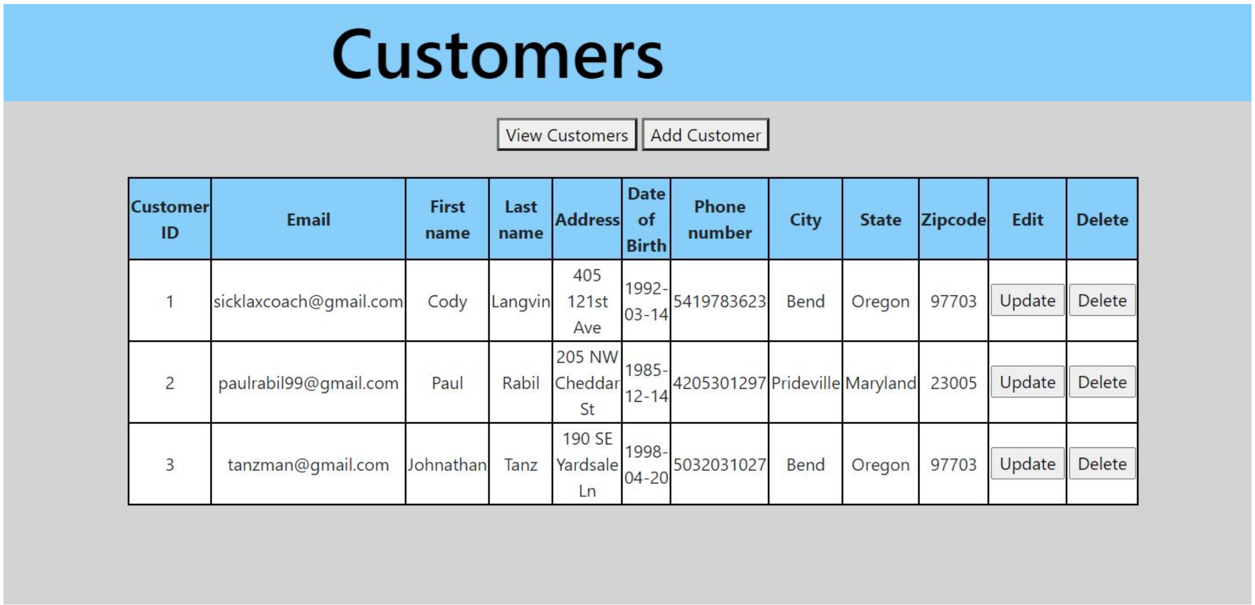
Product ID:

DateOrdered: mm/dd/yyyy

Date Delivered: mm/dd/yyyy

Total Price:

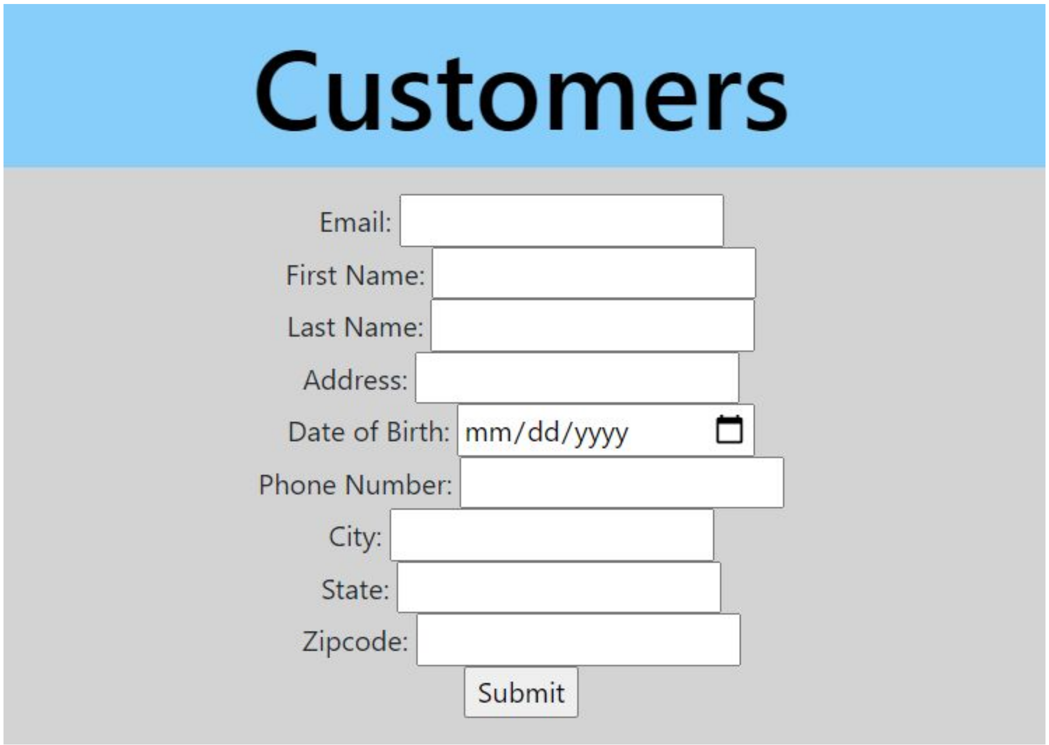
**READ/BROWSE/DISPLAY Customers  Page**



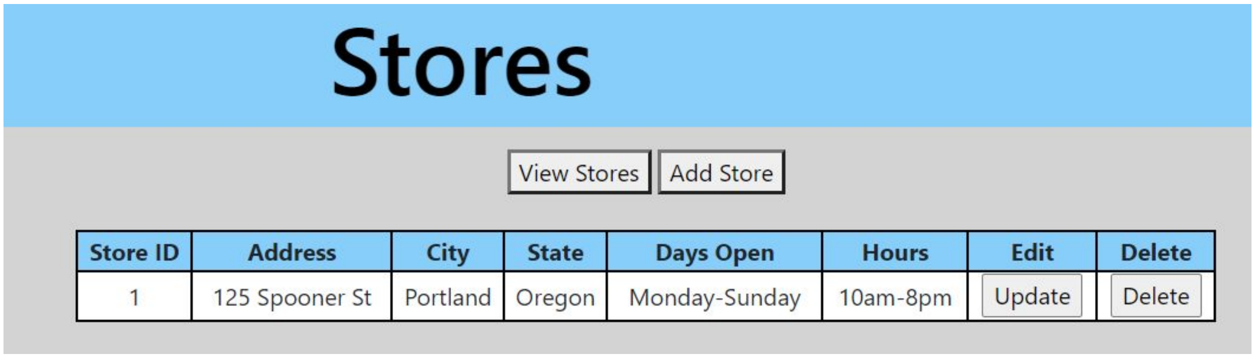
Customers

| Customer ID | Email | First name | Last name | Address | Date of Birth | Phone number | City | State | Zipcode | Edit | Delete |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | sicklaxcoach@gmail.com | Cody | Langvin | 405 121st Ave | 1992-03-14 | 5419783623 | Bend | Oregon | 97703 | Update | Delete |
| 2 | paulrabil99@gmail.com | Paul | Rabil | 205 NW Cheddar St | 1985-12-14 | 4205301297 | Prideville | Maryland | 23005 | Update | Delete |
| 3 | tanzman@gmail.com | Johnathan | Tanz | 190 SE Yardsale Ln | 1998-04-20 | 5032031027 | Bend | Oregon | 97703 | Update | Delete |

**CREATE/INSERT/ADD NEW Customers page**

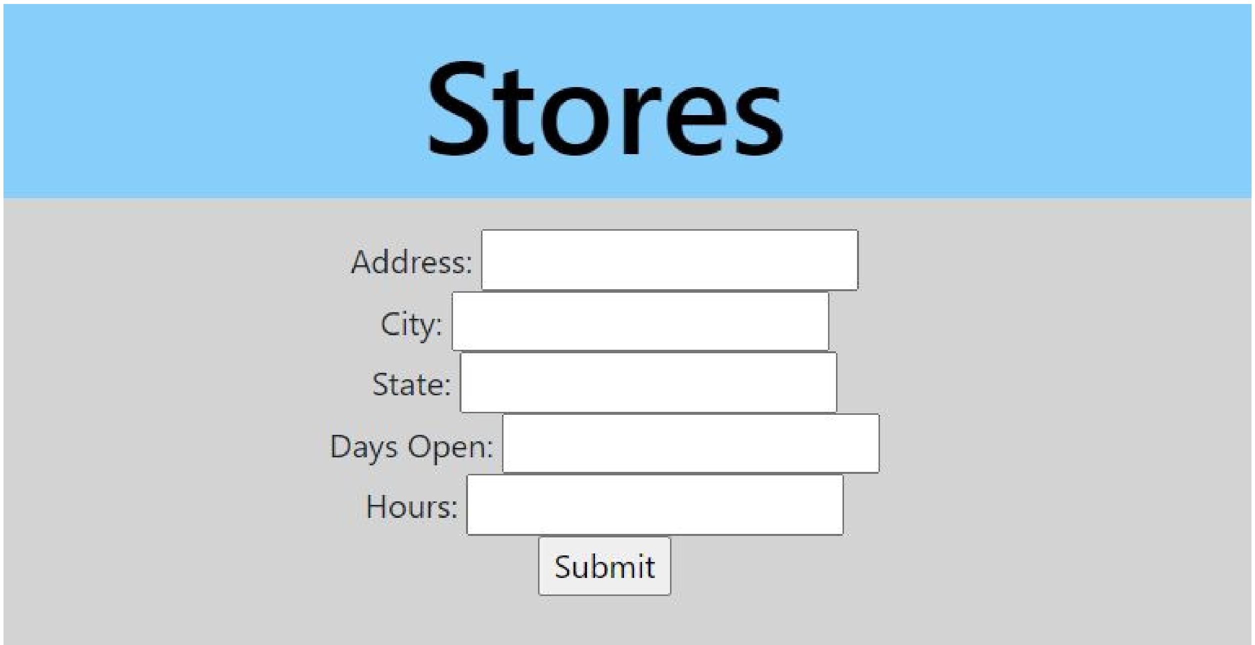


**READ/BROWSE/DISPLAY Stores Page**



| Store ID | Address | City | State | Days Open | Hours | Edit | Delete |
|---|---|---|---|---|---|---|---|
| 1 | 125 Spooner St | Portland | Oregon | Monday-Sunday | 10am-8pm | Update | Delete |

**CREATE/INSERT/ADD NEW Stores page**



Stores

Address:

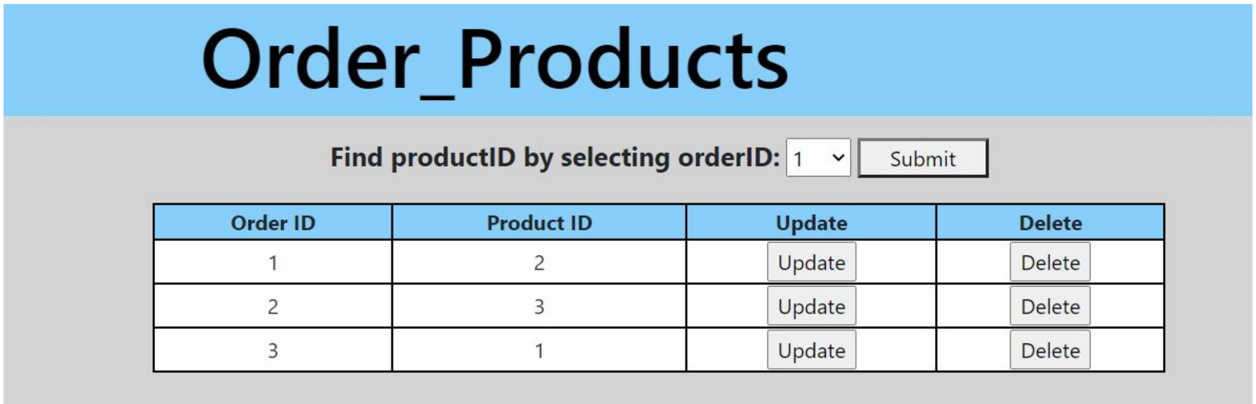
City:

State:

Days Open:

Hours:

**READ/BROWSE/DISPLAY Order_Products page**



Order_Products

**Find productID by selecting orderID:** 1

| Order ID | Product ID | Update | Delete |
|---|---|---|---|
| 1 | 2 | Update | Delete |
| 2 | 3 | Update | Delete |
| 3 | 1 | Update | Delete |