Project Report

Project Name: Beauty&Beast: Database Management System and Website Design for a Skin Care Products Company
Team Member: Song, C; Zhang, X; Zhu, Z
Table of Report Content:

1. Introduction

In this project, we design and implement a database management system and website for a skincare company. In our assumption, this company only sells products of its self-owned brand and has both E-commerce and in-store sales at the same time.

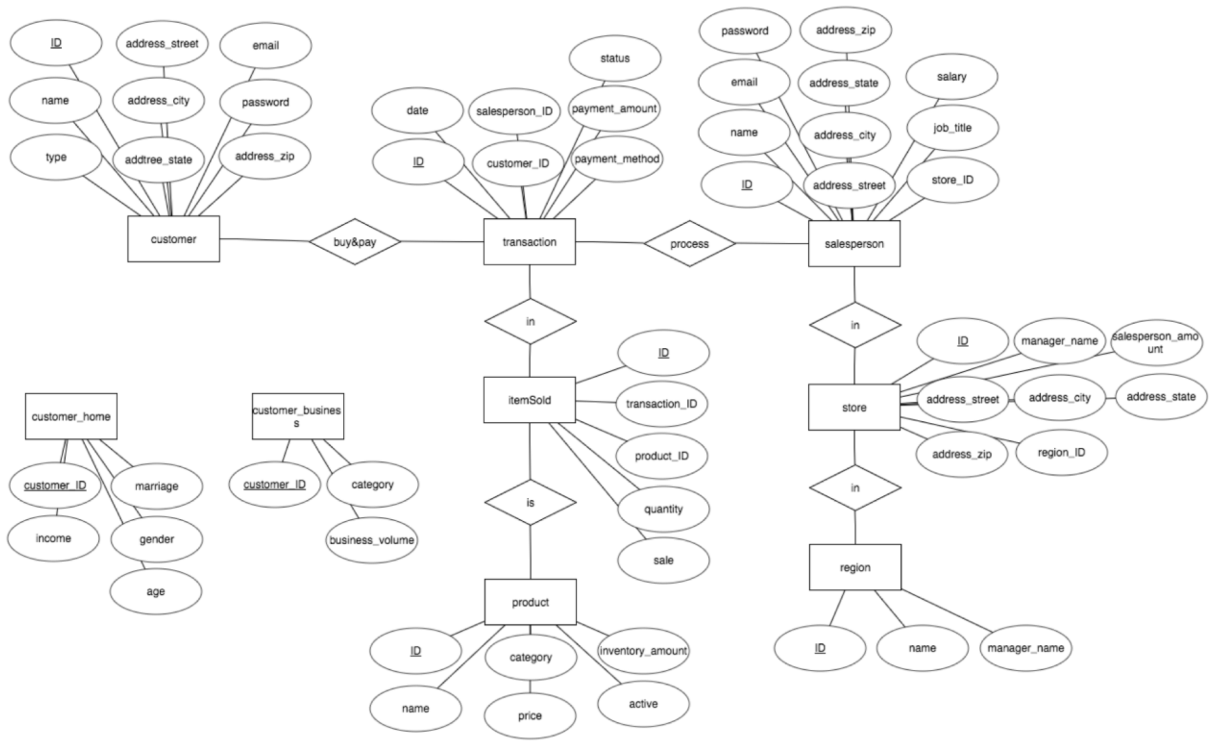
From the supply side, this company has one online center (located in New York City) and three stores (two in PA and one in OH), which form three regions. Each one of the four sites has a manager (they are salespeople at the same time), and each physical store has one more salesperson other than the store manager. The salesperson will be able to edit the information of each customer and log transactions for the customers. The managers will be able to edit the information of products, salespersons, and stores, check the status of the transactions and see aggregation information of sales.

From the demand side, there will be both home customers and business customers who will purchase products from our company and sell them in their own skincare services. The customers will be able to buy products online or in-store. If they are buying online, they will be able to browse the products, add products into a shopping cart, submit their orders and pay online through the website, and the sales will be count under the online center in NYC. If they are buying in-store, the salespersons will serve them and log the transactions into the database.

There are some other assumptions of the system: 1) Only registered customers can buy products, no matter online or in-store. New customers must be registered by themselves (online) or through a salesperson (in-store) first. 2) All customers are perfect ones since they will answer all questions asked to them, including gender, age, marriage, and income, and never return any product. 3) All stores will use the same inventory, which will not be true in the real world but can simplify our implementation. 4) This company only sells its own products, so we do not consider the company's purchasing, or in other words, increasing any inventory. 5) Correspondingly, we temporarily do not consider the situation that products may be out of stock to simplify the implementation. We only use "active" status to control each product to show on the website or not.

2. Database Design

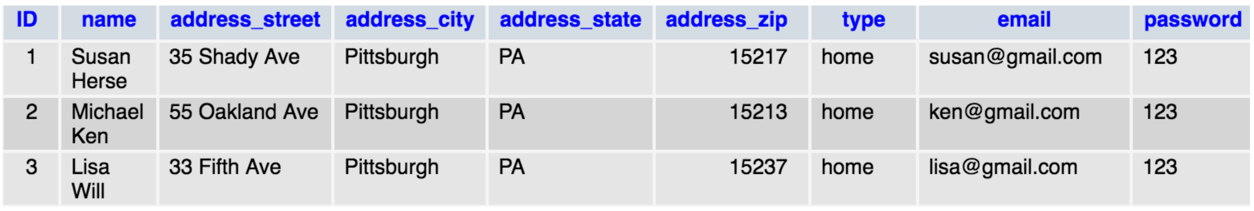
2.1 E-R Diagram



2.2 Description, Schemas, And DDL of Each Relation

As the E-R diagram shows, there will be 9 tables/relations in our database. Three of them (*customer, customer_home, customer_business*) manage the information of customers. The table of *customer* will store common information of all customers like name, address, email, and password. The other two tables will store the different information of different types of customers, so customer_ID in these two tables will be a primary key of themselves and a foreign key referencing *customer*. Another possible design here is keeping all information of customers in the same table and set the values to NA in not applicable schemas. We chose this 3-table design to avoid unnecessary NAs and assure BCNF (if using one-table design, there will be FDs not justifying BCNF, because marriage, gender, age and income of type “business,” or category and business_volume of type “home” will always be NA), but left more problems to the front end about how to avoid illegal insertion.

Table 1 customer:

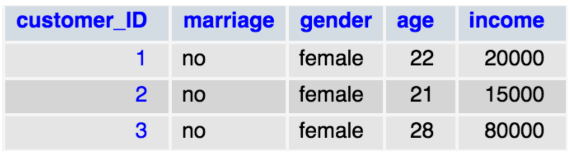


ID	name	address_street	address_city	address_state	address_zip	type	email	password
1	Susan Herse	35 Shady Ave	Pittsburgh	PA	15217	home	susan@gmail.com	123
2	Michael Ken	55 Oakland Ave	Pittsburgh	PA	15213	home	ken@gmail.com	123
3	Lisa Will	33 Fifth Ave	Pittsburgh	PA	15237	home	lisa@gmail.com	123

DDL of customer:

```
CREATE TABLE IF NOT EXISTS `Beauty&Beast`.`customer` (
  `ID` INT NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(45) NOT NULL,
  `address_street` VARCHAR(45) NOT NULL,
  `address_city` VARCHAR(45) NOT NULL,
  `address_state` VARCHAR(45) NOT NULL,
  `address_zip` INT NOT NULL,
  `type` VARCHAR(45) NOT NULL,
  `email` VARCHAR(45) NOT NULL,
  `password` VARCHAR(45) NOT NULL DEFAULT "123",
  PRIMARY KEY (`ID`))
```

Table 2 customer_home:

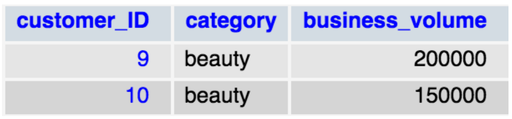


customer_ID	marriage	gender	age	income
1	no	female	22	20000
2	no	female	21	15000
3	no	female	28	80000

DDL of customer_home:

```
CREATE TABLE IF NOT EXISTS `Beauty&Beast`.`customer_home` (
  `customer_ID` INT NOT NULL,
  `marriage` VARCHAR(45) NOT NULL,
  `gender` VARCHAR(45) NOT NULL,
  `age` INT NOT NULL,
  `income` INT NOT NULL,
  PRIMARY KEY (`customer_ID`),
  CONSTRAINT `fk_customer_home`
    FOREIGN KEY (`customer_ID`)
    REFERENCES `Beauty&Beast`.`customer` (`ID`)
    ON DELETE NO ACTION
  ON UPDATE NO ACTION)
```

Table 3 customer_business:



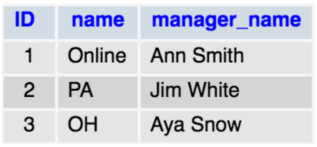
customer_ID	category	business_volume
9	beauty	200000
10	beauty	150000

DDL of customer_business:

```
CREATE TABLE IF NOT EXISTS `Beauty&Beast`.`product` (
  `ID` INT NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(45) NOT NULL,
  `inventory_amount` INT NOT NULL,
  `category` VARCHAR(45) NOT NULL,
  `price` DECIMAL(10,2) NOT NULL,
  `active` ENUM('1', '0') NOT NULL,
  PRIMARY KEY (`ID`))
```

There are also three relations manage the information of the company's structure: *salesperson, store,* and *region*. Each salesperson will be assigned to a store, and each store belongs to a region. The three tables connect with each other through foreign keys: store_ID in *salesperson*, and region_ID in *store*. Each store or region will have a manager who is also a salesperson. We did not record the managers' salesperson_ID in *store* or *region* but their names, which is a little problem in our database design because we did not consider about the same-name situation. Since this will not affect the functions of our database greatly, we keep it in an original way, but it can be fixed in a short time.

Table 4 region:

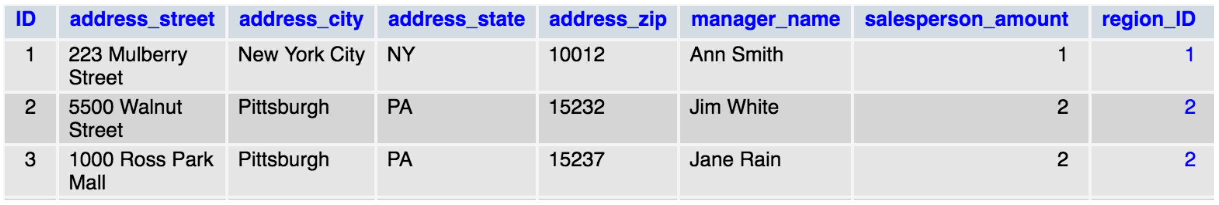


ID	name	manager_name
1	Online	Ann Smith
2	PA	Jim White
3	OH	Aya Snow

DDL of region:

```
CREATE TABLE IF NOT EXISTS `Beauty&Beast`.`region` (
  `ID` INT NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(45) NOT NULL,
  `manager_name` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`ID`))
```

Table 5 store:

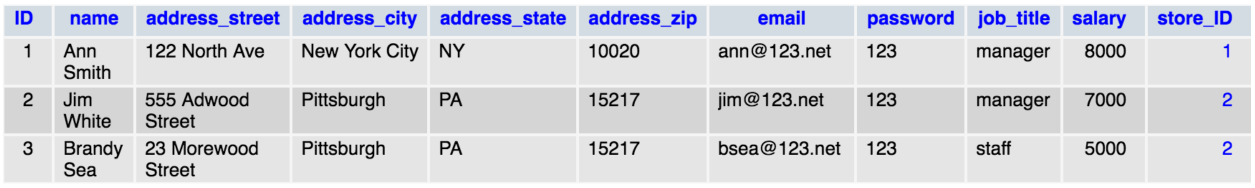


ID	address_street	address_city	address_state	address_zip	manager_name	salesperson_amount	region_ID
1	223 Mulberry Street	New York City	NY	10012	Ann Smith	1	1
2	5500 Walnut Street	Pittsburgh	PA	15232	Jim White	2	2
3	1000 Ross Park Mall	Pittsburgh	PA	15237	Jane Rain	2	2

DDL of store:

```
CREATE TABLE IF NOT EXISTS `Beauty&Beast`.`store` (
  `ID` INT NOT NULL AUTO_INCREMENT,
  `address_street` VARCHAR(45) NOT NULL,
  `address_city` VARCHAR(45) NOT NULL,
  `address_state` VARCHAR(45) NOT NULL,
  `address_zip` VARCHAR(45) NOT NULL,
  `manager_name` VARCHAR(45) NOT NULL,
  `salesperson_amount` INT NOT NULL,
  `region_ID` INT NOT NULL,
  PRIMARY KEY (`ID`),
  CONSTRAINT `fk_store_region`
    FOREIGN KEY (`region_ID`)
    REFERENCES `Beauty&Beast`.`region` (`ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
```

Table 6 salesperson:



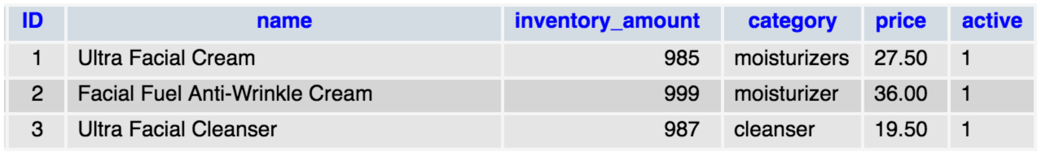
ID	name	address_street	address_city	address_state	address_zip	email	password	job_title	salary	store_ID
1	Ann Smith	122 North Ave	New York City	NY	10020	ann@123.net	123	manager	8000	1
2	Jim White	555 Adwood Street	Pittsburgh	PA	15217	jim@123.net	123	manager	7000	2
3	Brandy Sea	23 Morewood Street	Pittsburgh	PA	15217	bsea@123.net	123	staff	5000	2

DDL of salesperson:

```
CREATE TABLE IF NOT EXISTS `Beauty&Beast`.`salesperson` (
  `ID` INT NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(45) NOT NULL,
  `address_street` VARCHAR(45) NOT NULL,
  `address_city` VARCHAR(45) NOT NULL,
  `address_state` VARCHAR(45) NOT NULL,
  `address_zip` VARCHAR(45) NOT NULL,
  `email` VARCHAR(45) NOT NULL,
  `password` VARCHAR(45) NOT NULL DEFAULT "123",
  `job_title` VARCHAR(45) NOT NULL,
  `salary` INT NOT NULL,
  `store_ID` INT NOT NULL,
  PRIMARY KEY (`ID`),
  CONSTRAINT `fk_salesperson_store`
    FOREIGN KEY (`store_ID`)
    REFERENCES `Beauty&Beast`.`store` (`ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
```

The *product* table deals with the information about our products. As mentioned before, we assume that all stores use the same inventory. This could not be true for a company that has 4 sites and online and in-store sales at the same time but will simplify our implementation a lot.

Table 7 product:



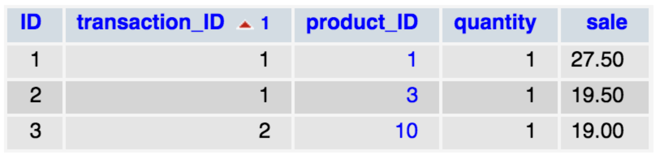
ID	name	inventory_amount	category	price	active
1	Ultra Facial Cream	985	moisturizers	27.50	1
2	Facial Fuel Anti-Wrinkle Cream	999	moisturizer	36.00	1
3	Ultra Facial Cleanser	987	cleanser	19.50	1

DDL of product:

```
CREATE TABLE IF NOT EXISTS `Beauty&Beast`.`product` (
  `ID` INT NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(45) NOT NULL,
  `inventory_amount` INT NOT NULL,
  `category` VARCHAR(45) NOT NULL,
  `price` DECIMAL(10,2) NOT NULL,
  `active` ENUM('1', '0') NOT NULL,
  PRIMARY KEY (`ID`))
```

ItemSold and *transaction* are the two tables that are in the center of our database and connect all relations together. *Transaction* record data, customer and salesperson ID (referencing *customer* and *salesperson*), and payment information of transactions. It connects with *itemSold* through using ID to reference transaction_ID in *itemSold*. *ItemSold* records product information in transactions, including product_ID, which references *product*, and quantity of each product in a transaction. We also add a trigger on *itemSold* so that once a tuple is inserted into *itemSold*, the trigger will update the inventory_amount in *product*.
The information of sale (of each product in a transaction) in *itemSold* and payment_amount in *transaction* will be calculated in the front end and inserted into the tables. We keep these two tables about transaction information because there could be different products in the same transaction. If we record all the information in one table, the information of date, customer, salesperson, and payment will have redundancy. Therefore, we keep two tables to assure BCNF: now, in the two tables, the only ID--all is an FD other than trivial FDs. Similarly, all other relations in our database are BCNF as well.

Table 8 itemSold

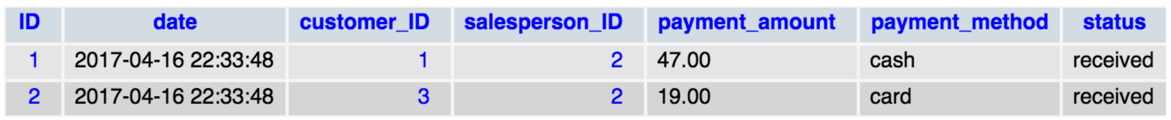


ID	transaction_ID ▲ 1	product_ID	quantity	sale
1	1	1	1	27.50
2	1	3	1	19.50
3	2	10	1	19.00

DDL of itemSold

```
CREATE TABLE IF NOT EXISTS `Beauty&Beast`.`itemSold` (
  `ID` int(11) NOT NULL AUTO_INCREMENT,
  `transaction_ID` INT NOT NULL,
  `product_ID` INT NOT NULL,
  `quantity` INT NOT NULL,
  `sale` DECIMAL(10,2),
  PRIMARY KEY (`ID`, `product_ID`),
  CONSTRAINT `fk_order_product`
    FOREIGN KEY (`product_ID`)
    REFERENCES `Beauty&Beast`.`product` (`ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
```

Table 9 transaction



ID	date	customer_ID	salesperson_ID	payment_amount	payment_method	status
1	2017-04-16 22:33:48	1	2	47.00	cash	received
2	2017-04-16 22:33:48	3	2	19.00	card	received

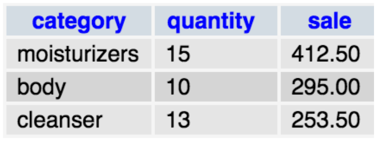
DDL of transaction

```
CREATE TABLE IF NOT EXISTS `Beauty&Beast`.`transaction` (
  `ID` INT NOT NULL AUTO_INCREMENT,
  `date` TIMESTAMP NOT NULL,
  `customer_ID` INT NOT NULL,
  `salesperson_ID` INT NOT NULL,
  `payment_amount` DECIMAL(10,2),
  `payment_method` VARCHAR(45) NOT NULL,
  `status` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`ID`),
  CONSTRAINT `fk_transaction_ID`
    FOREIGN KEY (`ID`)
    REFERENCES `itemSold` (`transaction_ID`)
    ON UPDATE CASCADE,
  CONSTRAINT `fk_transaction_customer`
    FOREIGN KEY (`customer_ID`)
    REFERENCES `Beauty&Beast`.`customer` (`ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_transaction_salesperson`
    FOREIGN KEY (`salesperson_ID`)
    REFERENCES `Beauty&Beast`.`salesperson` (`ID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
```

2.3 Aggregation Views

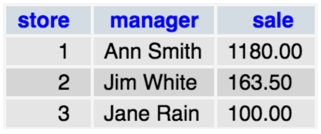
We also create some views in the backend to fulfill the aggregation functions. In our design, each time when an aggregation request is submitted, a corresponding view will be created or replaced, and the result will be displayed in the frontend. Here are some sample views:

Table 10 v_category_sale



category	quantity	sale
moisturizers	15	412.50
body	10	295.00
cleanser	13	253.50

Table 11 v_store_sale



store	manager	sale
1	Ann Smith	1180.00
2	Jim White	163.50
3	Jane Rain	100.00

3. Front End Design, Implementation, and Test

Our frontend page consists of two parts: customers and salespersons. Customers and salespersons use different login pages to get access to be granted independently of each other.

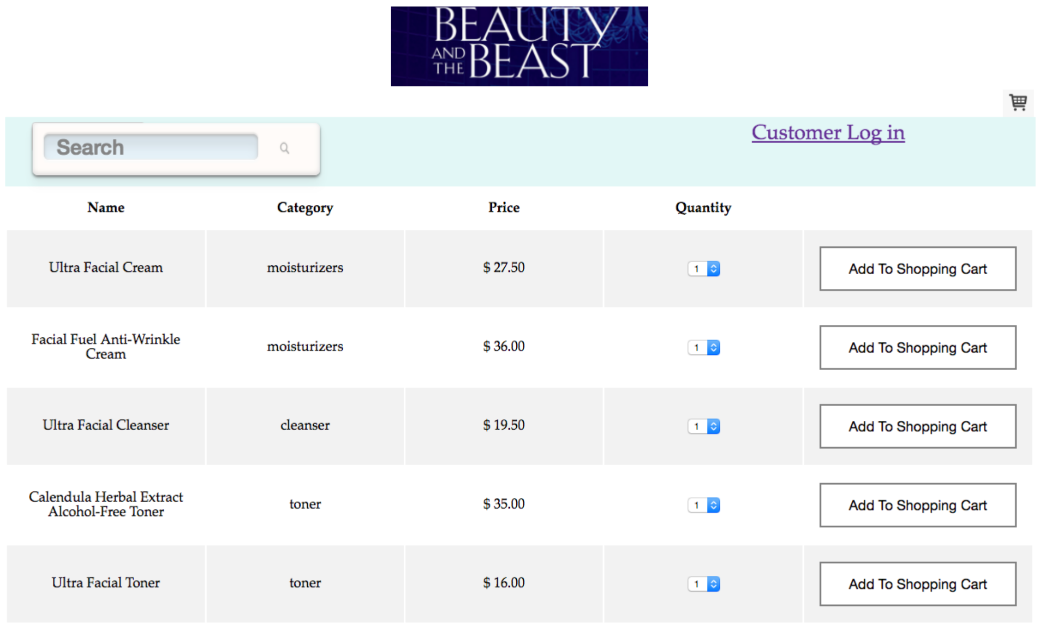
3.1 Customers

Front End Design

The frontend design of the two parts is based on several basic web languages: HTML, PHP, and JavaScript. The main functions of the frontend for an unregistered customer are browsing all the products and fuzzy searching for what they need using keywords. The main functions of the frontend for the customer who has logged in are more, including adding products to a shopping cart, buying products online, checking personal information, and viewing order history.

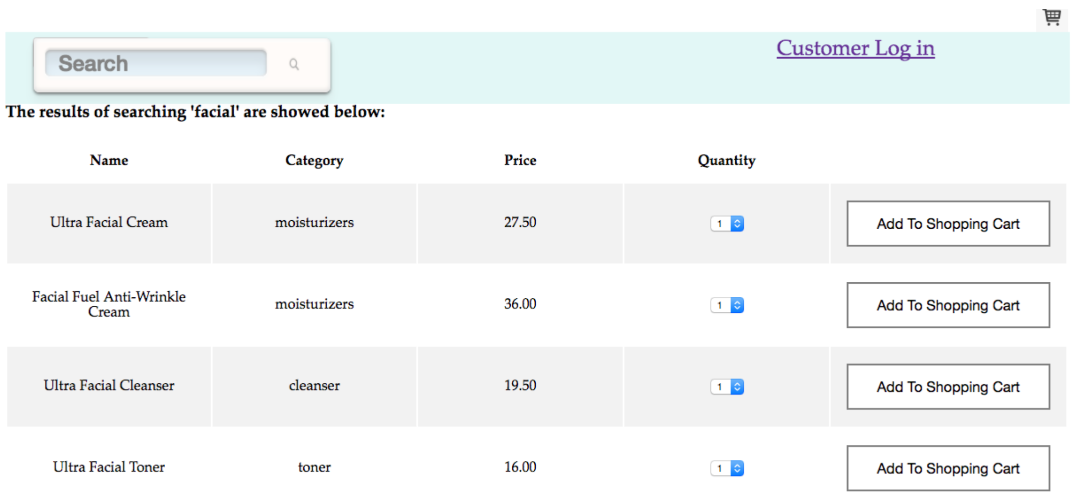
System Implementation and Testing

For an unregistered customer, he can ***1. Browse all the Product Information***:



Name	Category	Price	Quantity	
Ultra Facial Cream	moisturizers	$ 27.50	1	Add To Shopping Cart
Facial Fuel Anti-Wrinkle Cream	moisturizers	$ 36.00	1	Add To Shopping Cart
Ultra Facial Cleanser	cleanser	$ 19.50	1	Add To Shopping Cart
Calendula Herbal Extract Alcohol-Free Toner	toner	$ 35.00	1	Add To Shopping Cart
Ultra Facial Toner	toner	$ 16.00	1	Add To Shopping Cart

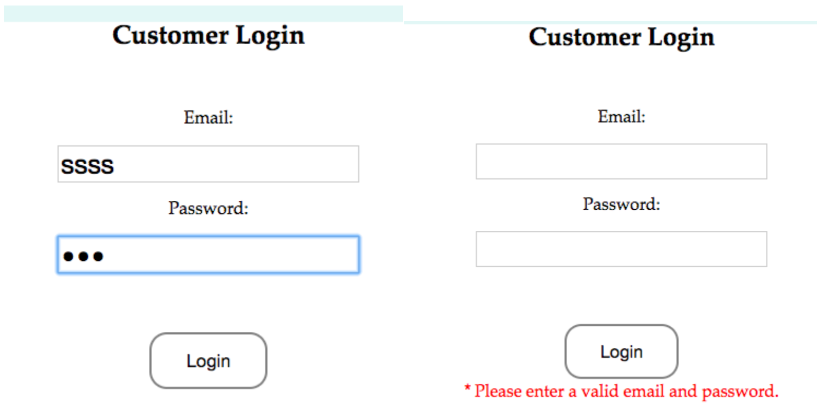
For an unregistered customer, he can ***2. Search for what he needs***:



The results of searching 'facial' are showed below:

Name	Category	Price	Quantity	
Ultra Facial Cream	moisturizers	27.50	1	Add To Shopping Cart
Facial Fuel Anti-Wrinkle Cream	moisturizers	36.00	1	Add To Shopping Cart
Ultra Facial Cleanser	cleanser	19.50	1	Add To Shopping Cart
Ultra Facial Toner	toner	16.00	1	Add To Shopping Cart

For an unregistered customer, he can ***3. Login****:*
And the email input will be **validate**d. Here’s a wrong example:


Customer Login

Email:

ssss

Password:

•••

Login

Customer Login

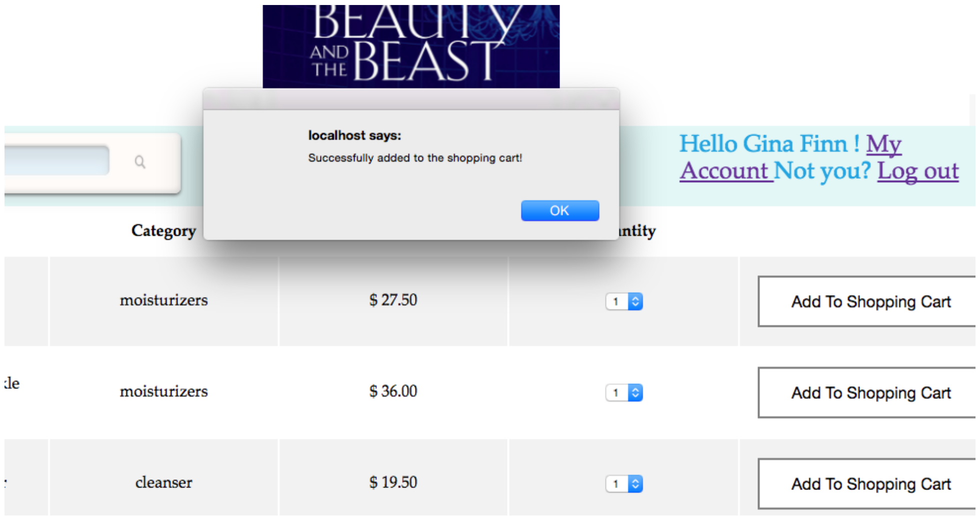
Email:

Password:

Login

* Please enter a valid email and password.

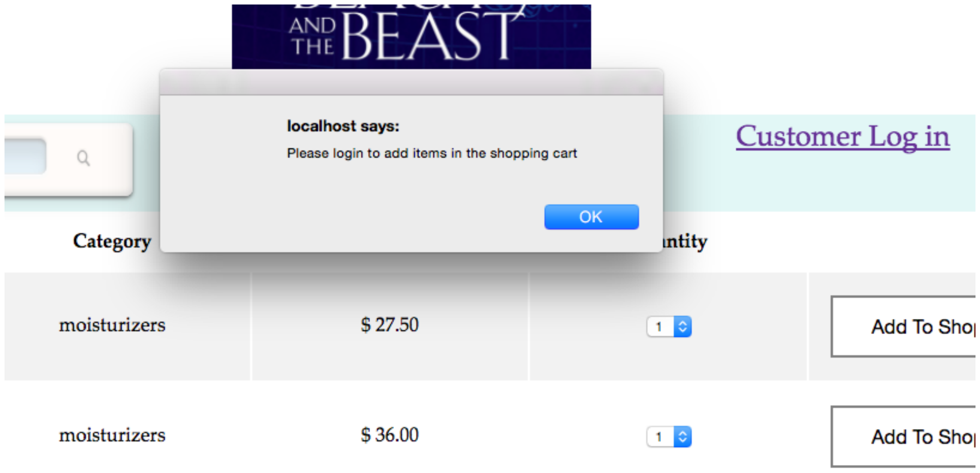
For a logged-in customer, he can ***4. Add Products into Shopping Cart***.:
If a customer does not log in, he cannot add products to the shopping cart.


localhost says:
Successfully added to the shopping cart!

OK

Hello Gina Finn ! My Account Not you? Log out

Category		ntity	
moisturizers	$ 27.50	1	Add To Shopping Cart
moisturizers	$ 36.00	1	Add To Shopping Cart
cleanser	$ 19.50	1	Add To Shopping Cart

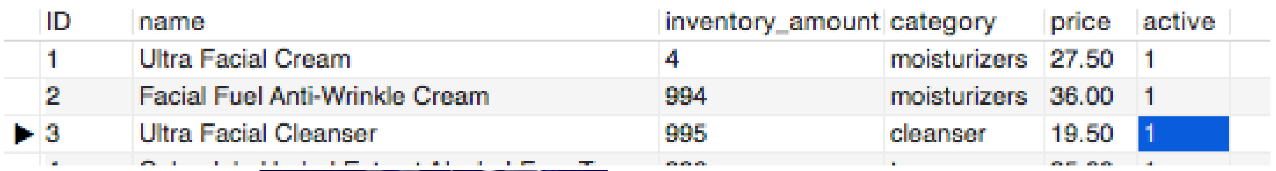

localhost says:
Please login to add items in the shopping cart

OK

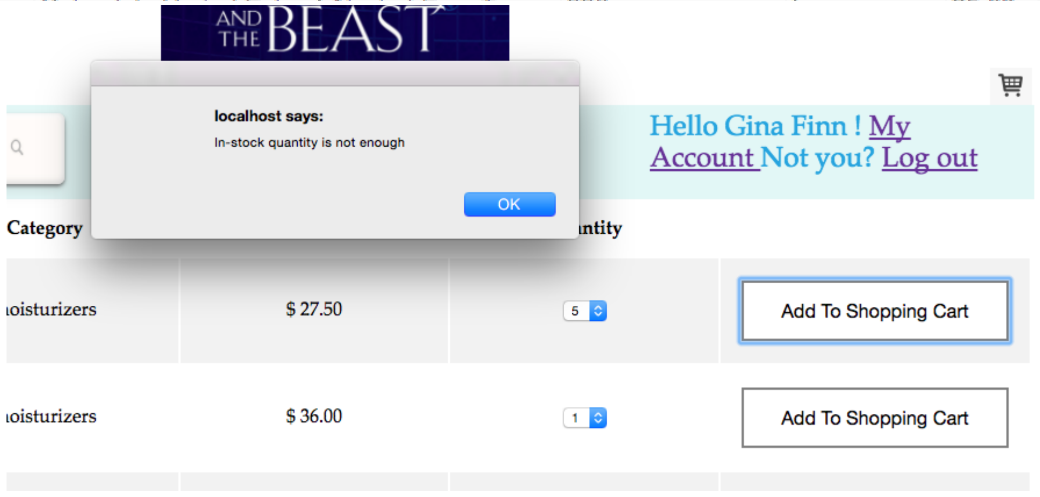
Customer Log in

Category		ntity	
moisturizers	$ 27.50	1	Add To Sho
moisturizers	$ 36.00	1	Add To Sho

The ***in-stock amount*** of the selected product will be checked when the ‘Add To Shopping Cart’ button is clicked. When the ‘inventory_amount’ of ‘product(ID=1)’ is 4, which is less than the selected quantity (which is 5), so the product will not be added to the shopping cart.



ID	name	inventory_amount	category	price	active
1	Ultra Facial Cream	4	moisturizers	27.50	1
2	Facial Fuel Anti-Wrinkle Cream	994	moisturizers	36.00	1
3	Ultra Facial Cleanser	995	cleanser	19.50	1

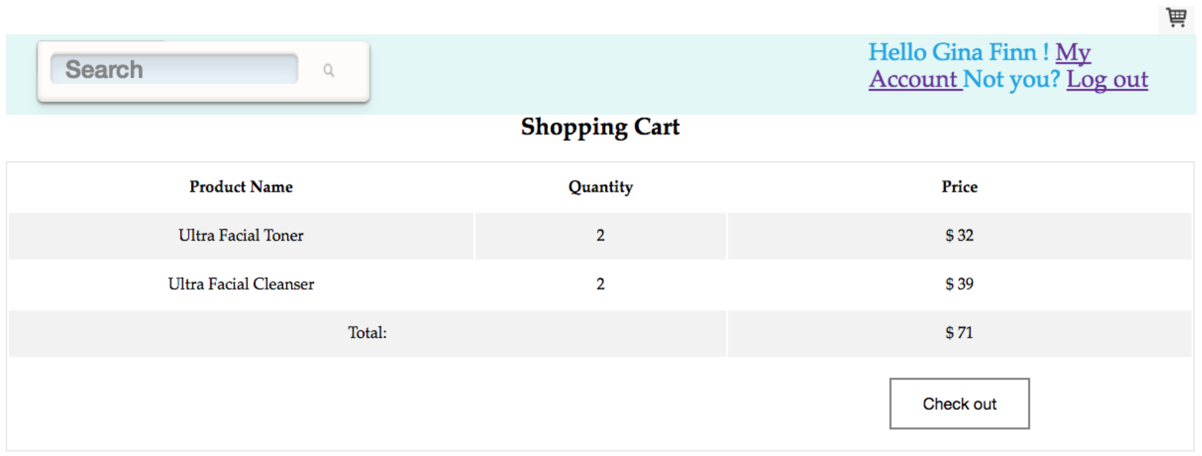


localhost says:
In-stock quantity is not enough

Hello Gina Finn ! My Account Not you? Log out

Category		ntity	
oisturizers	$ 27.50	5	Add To Shopping Cart
oisturizers	$ 36.00	1	Add To Shopping Cart

For a logged-in customer, he can ***5. View the Shopping Cart and Check out:***

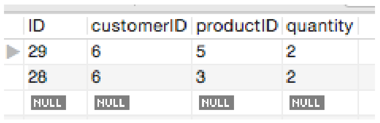


Hello Gina Finn ! My Account Not you? Log out

Shopping Cart

Product Name	Quantity	Price
Ultra Facial Toner	2	$ 32
Ultra Facial Cleanser	2	$ 39
Total:		$ 71

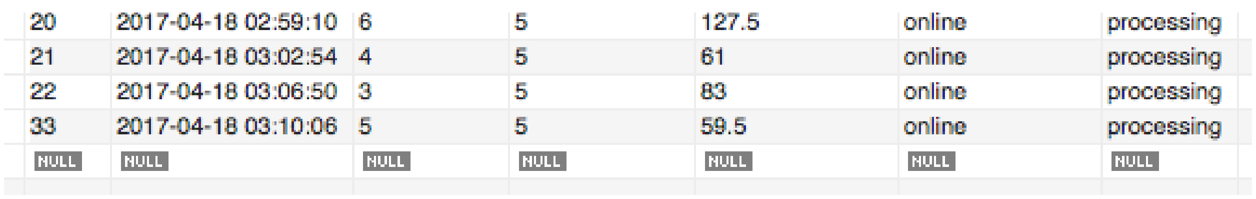
Also, the information will be recorded in the Table cart:



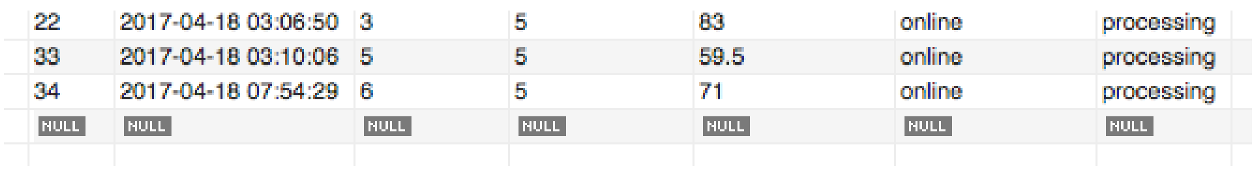
ID	customerID	productID	quantity
29	6	5	2
28	6	3	2
NULL	NULL	NULL	NULL

After clicking ‘Check out’, the information will be recorded in Table transaction, itemSold:

Table transaction:

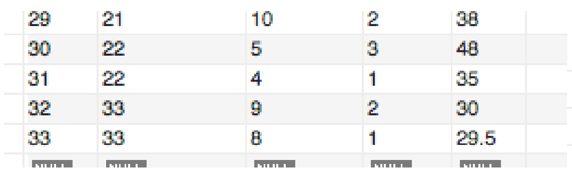


20	2017-04-18 02:59:10	6	5	127.5	online	processing
21	2017-04-18 03:02:54	4	5	61	online	processing
22	2017-04-18 03:06:50	3	5	83	online	processing
33	2017-04-18 03:10:06	5	5	59.5	online	processing
NULL	NULL	NULL	NULL	NULL	NULL	NULL

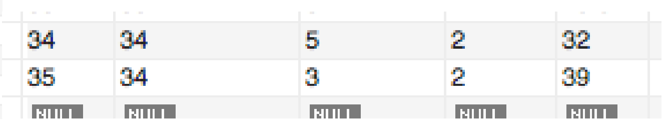


22	2017-04-18 03:06:50	3	5	83	online	processing
33	2017-04-18 03:10:06	5	5	59.5	online	processing
34	2017-04-18 07:54:29	6	5	71	online	processing
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Table itemSold:

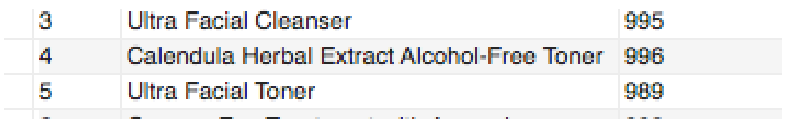


29	21	10	2	38
30	22	5	3	48
31	22	4	1	35
32	33	9	2	30
33	33	8	1	29.5

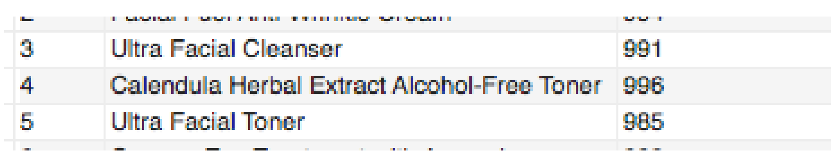


34	34	5	2	32
35	34	3	2	39
NULL	NULL	NULL	NULL	NULL

Table product (inventory_amount of product 3&5 will be updated):

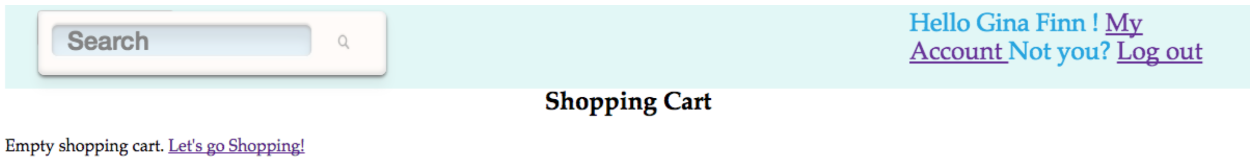


3	Ultra Facial Cleanser	995
4	Calendula Herbal Extract Alcohol-Free Toner	996
5	Ultra Facial Toner	989



3	Ultra Facial Cleanser	991
4	Calendula Herbal Extract Alcohol-Free Toner	996
5	Ultra Facial Toner	985

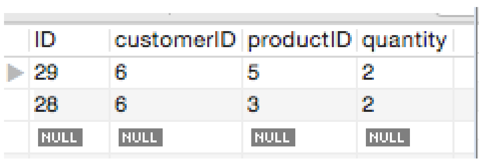
Data will be cleared both in the frontend shopping cart and backend Table cart:



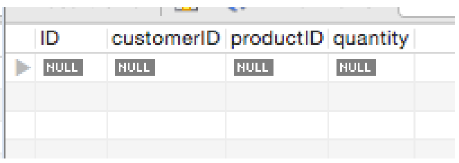
Hello Gina Finn ! My Account Not you? Log out

Shopping Cart

Empty shopping cart. Let's go Shopping!

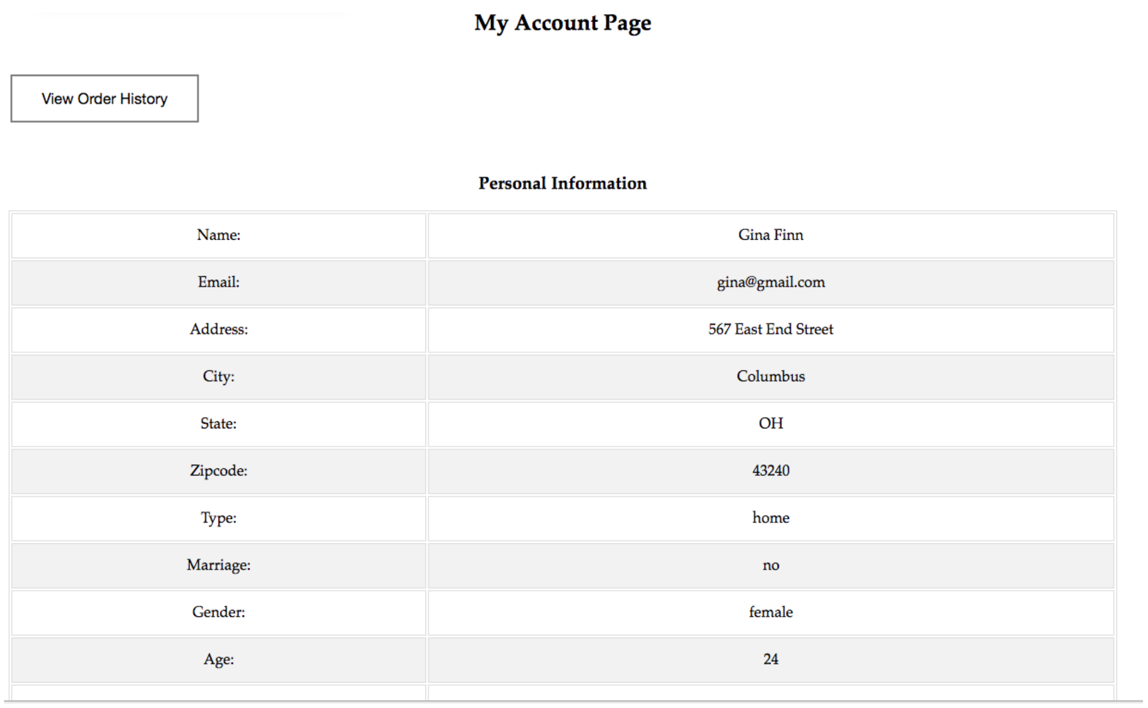


ID	customerID	productID	quantity
29	6	5	2
28	6	3	2
NULL	NULL	NULL	NULL



ID	customerID	productID	quantity
NULL	NULL	NULL	NULL

For a logged in customer, he can ***6. Viewing My Account Page:***



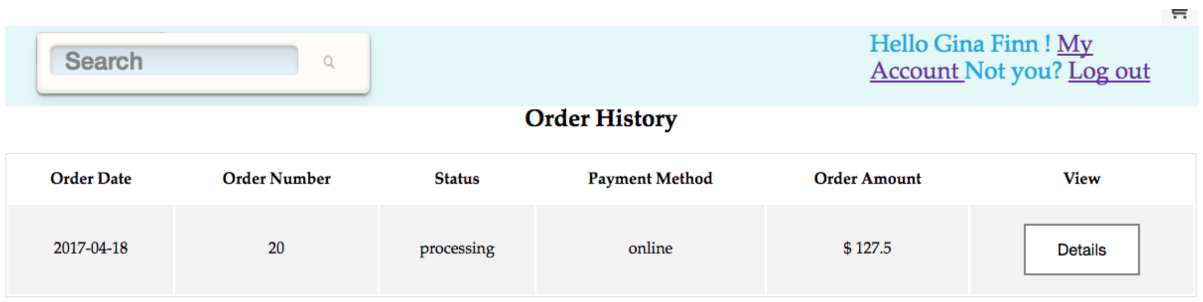
My Account Page

View Order History

Personal Information

Name:	Gina Finn
Email:	gina@gmail.com
Address:	567 East End Street
City:	Columbus
State:	OH
Zipcode:	43240
Type:	home
Marriage:	no
Gender:	female
Age:	24

For a logged in customer, he can ***7. Viewing Order History:***



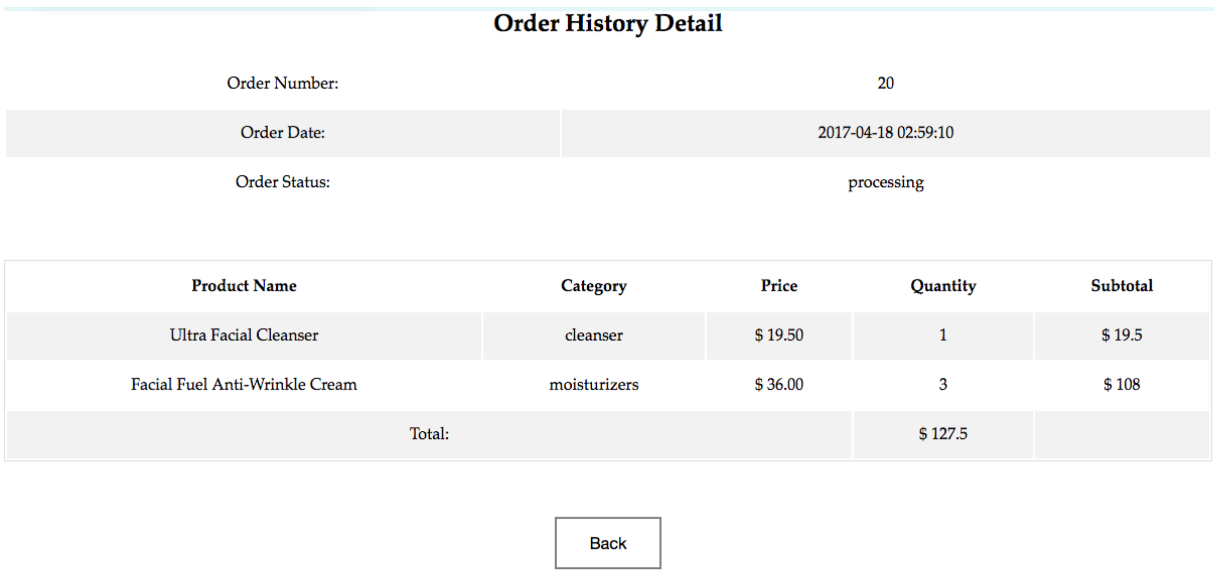
Search

Hello Gina Finn ! My Account Not you? Log out

Order History

Order Date	Order Number	Status	Payment Method	Order Amount	View
2017-04-18	20	processing	online	$ 127.5	Details

After clicking ‘Details’ button, customer can ***8. Check some more Detailed Order Information:*** The ‘Back’ button will *Link Back to the Order History Page*.

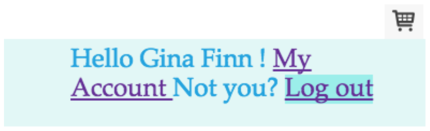


Order History Detail

Order Number:	20
Order Date:	2017-04-18 02:59:10
Order Status:	processing

Product Name	Category	Price	Quantity	Subtotal
Ultra Facial Cleanser	cleanser	$ 19.50	1	$ 19.5
Facial Fuel Anti-Wrinkle Cream	moisturizers	$ 36.00	3	$ 108
Total:			$ 127.5	

For a logged-in customer, he can ***9. Logout.*** After a customer logout, the original session will be destroyed.



3.2 Salespersons

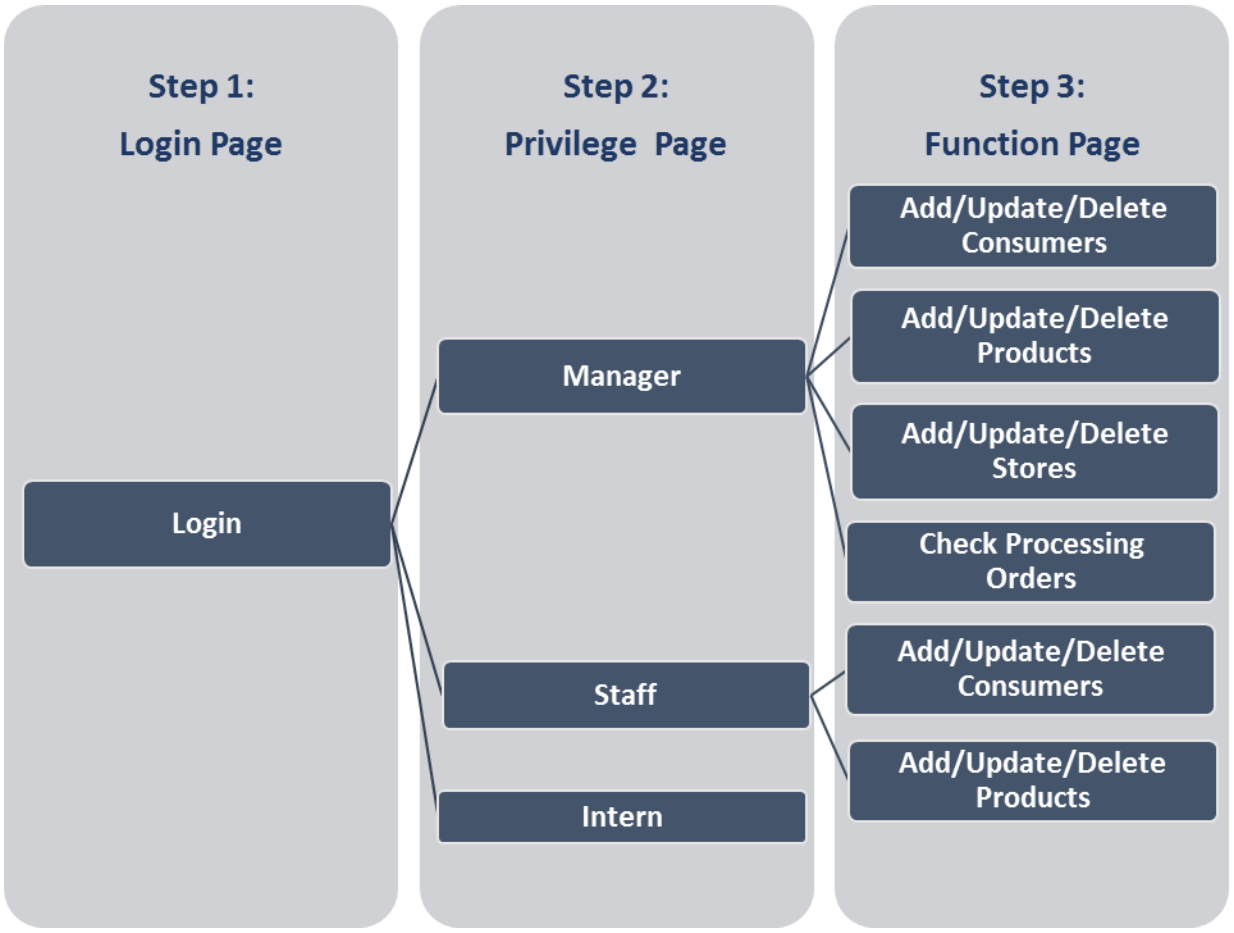
Front End Design

The frontend for salespersons is a comprehensive web application that includes a login page, a privileged page displaying what operations this person has been granted to conduct, and several function pages that navigate the salesperson to conduct different operations. The job title of the salesperson will be checking when he/she log in and different function pages will be exposed to him/her depending on his/her job title. A manager will be granted a higher privilege than a staff. Any staff can add, update and delete consumers and products, and a manager is also able to add, update and delete stores and check processing orders. Interns have no privilege to work on the database system from the frontend.

System Implementation And Testing

Figure 1 below shows the implementation flow of our system for salespersons at Beauty&Beast.

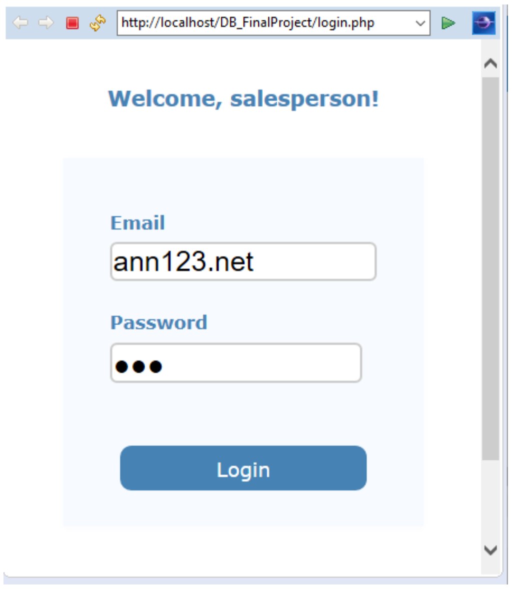
Figure 1 System Implementation Flow Chart



Step 1: Login Page

Input validation: To prevent the entry of unsafe data into the web application, input validation is always performed on our front end when salespersons are adding or updating data into the system.

Login input validation testing:


http://localhost/DB_FinalProject/login.php

Welcome, salesperson!

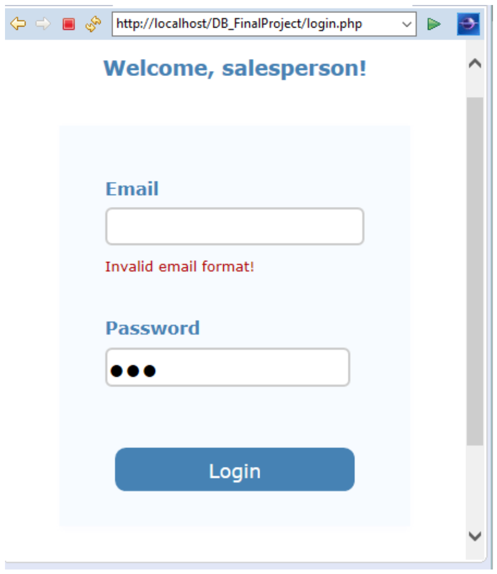
Email

ann123.net

Password

●●●

Login


http://localhost/DB_FinalProject/login.php

Welcome, salesperson!

Email

Invalid email format!

Password

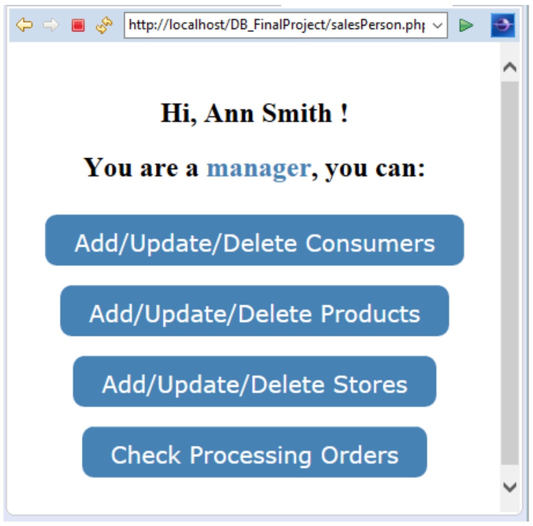
●●●

Login

Since input validation will be performing before the frontend connects to the backend, the invalid input of “ann123.net” will generate an error message in red color. Once the user enters an email address in a valid format, the frontend will connect to the backend and match this user with data in the database.

Step 2: Privilege Page

Different privilege pages will be displayed according to the job titles. Ann Smith using the email of “ann@123.net” is a manager, so she has been granted the highest privilege.


http://localhost/DB_FinalProject/salesPerson.php

Hi, Ann Smith !

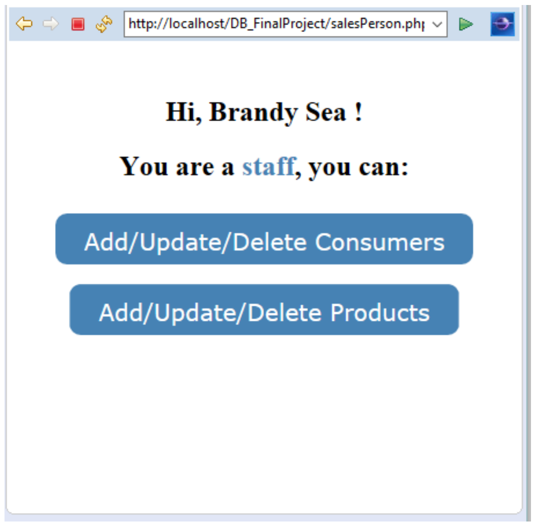
You are a manager, you can:

Add/Update/Delete Consumers

Add/Update/Delete Products

Add/Update/Delete Stores

Check Processing Orders

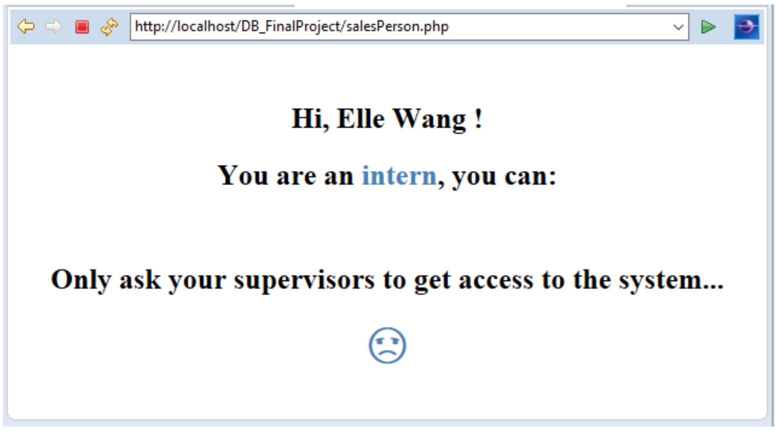

http://localhost/DB_FinalProject/salesPerson.php

Hi, Brandy Sea !

You are a staff, you can:

Add/Update/Delete Consumers

Add/Update/Delete Products


http://localhost/DB_FinalProject/salesPerson.php

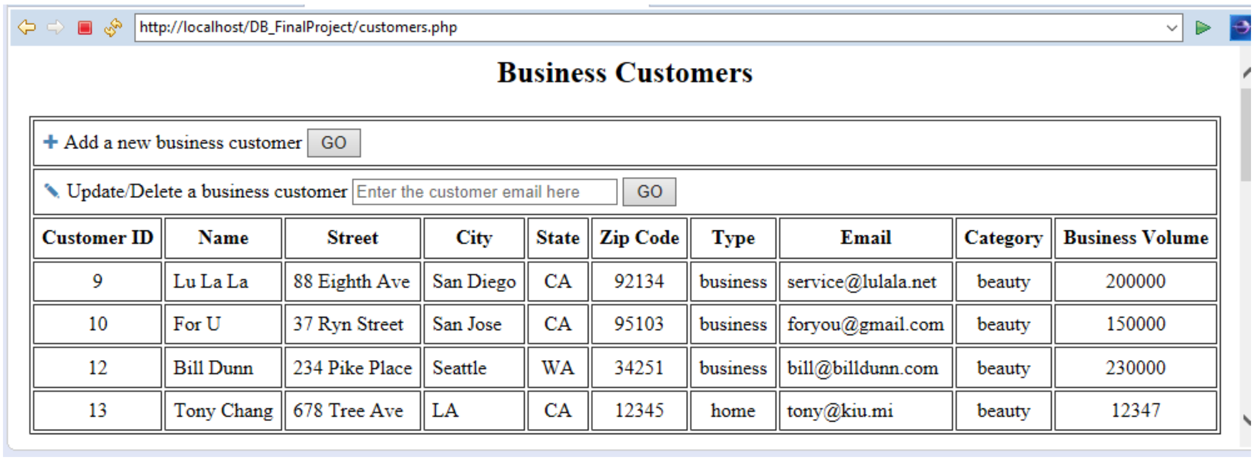
Hi, Elle Wang !

You are an intern, you can:

Only ask your supervisors to get access to the system...

Step 3: Function Pages

There are two types of customers in our system: home customers and business customers. Any salesperson can view all the customers of these two types.

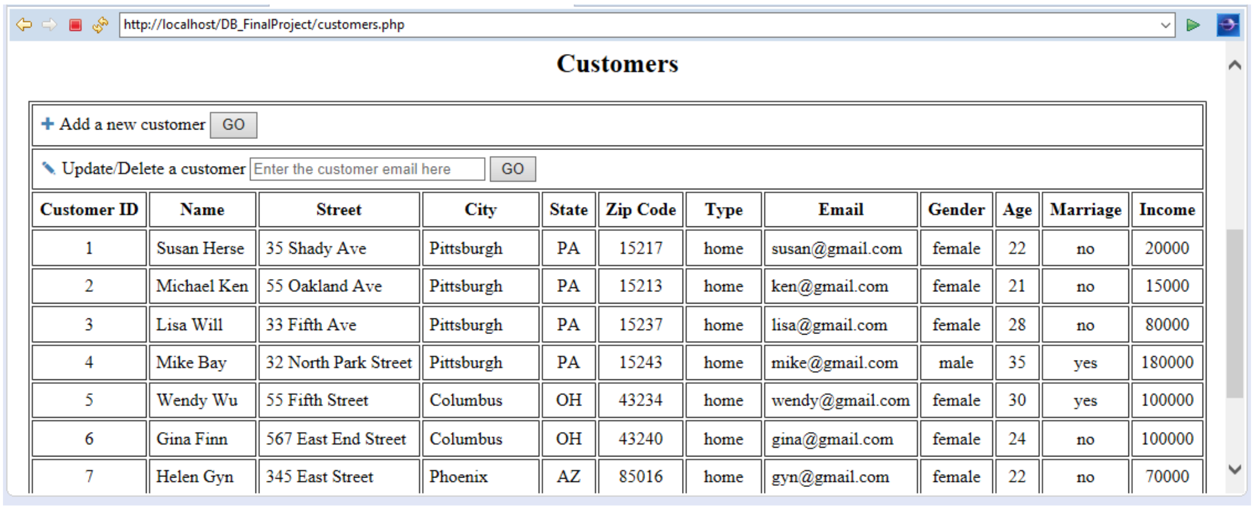

http://localhost/DB_FinalProject/customers.php

Business Customers

Add a new business customer GO

Update/Delete a business customer Enter the customer email here GO

Customer ID	Name	Street	City	State	Zip Code	Type	Email	Category	Business Volume
9	Lu La La	88 Eighth Ave	San Diego	CA	92134	business	service@lulala.net	beauty	200000
10	For U	37 Ryn Street	San Jose	CA	95103	business	foryou@gmail.com	beauty	150000
12	Bill Dunn	234 Pike Place	Seattle	WA	34251	business	bill@billdunn.com	beauty	230000
13	Tony Chang	678 Tree Ave	LA	CA	12345	home	tony@kiu.mi	beauty	12347


http://localhost/DB_FinalProject/customers.php

Customers

Add a new customer GO

Update/Delete a customer Enter the customer email here GO

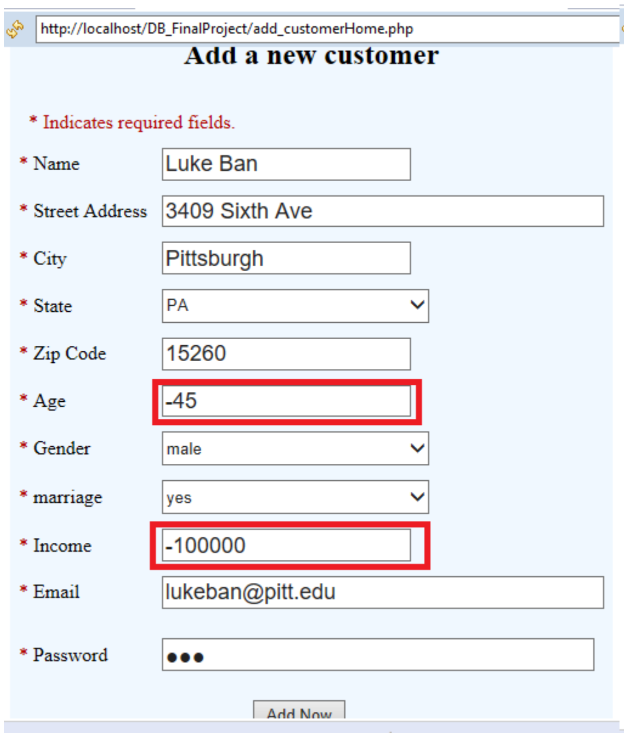
Customer ID	Name	Street	City	State	Zip Code	Type	Email	Gender	Age	Marriage	Income
1	Susan Herse	35 Shady Ave	Pittsburgh	PA	15217	home	susan@gmail.com	female	22	no	20000
2	Michael Ken	55 Oakland Ave	Pittsburgh	PA	15213	home	ken@gmail.com	female	21	no	15000
3	Lisa Will	33 Fifth Ave	Pittsburgh	PA	15237	home	lisa@gmail.com	female	28	no	80000
4	Mike Bay	32 North Park Street	Pittsburgh	PA	15243	home	mike@gmail.com	male	35	yes	180000
5	Wendy Wu	55 Fifth Street	Columbus	OH	43234	home	wendy@gmail.com	female	30	yes	100000
6	Gina Finn	567 East End Street	Columbus	OH	43240	home	gina@gmail.com	female	24	no	100000
7	Helen Gyn	345 East Street	Phoenix	AZ	85016	home	gyn@gmail.com	female	22	no	70000

The information of the customers is selected from the table of *customer* in the database join the table of customer_business or join the table of customer_home. Therefore, the salesperson can easily review the specific information of the customers rather than reading some attributes that refer to indexes, such as some foreign keys.

1) Add/Update/Delete Consumers

Add a new customer:

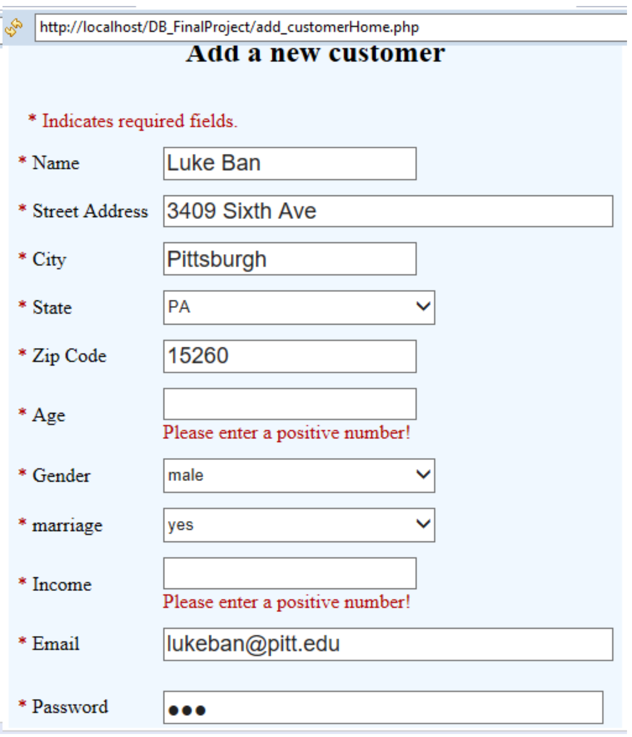
All our forms on the frontend pages are sticky; thus any page that contains input fields would keep the valid inputs when it requires the user to correct some invalid inputs. Just as the demonstration in the screenshots below, if we put a negative age and income, the page would get rid of the invalid data and keep the valid ones.


http://localhost/DB_FinalProject/add_customerHome.php

Add a new customer

* Indicates required fields.

* Name	Luke Ban
* Street Address	3409 Sixth Ave
* City	Pittsburgh
* State	PA
* Zip Code	15260
* Age	-45
* Gender	male
* marriage	yes
* Income	-100000
* Email	lukeban@pitt.edu
* Password	●●●

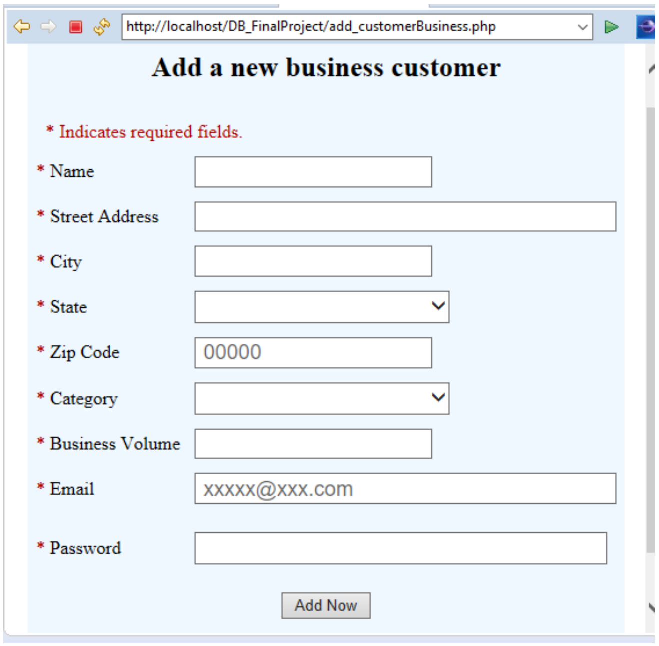

http://localhost/DB_FinalProject/add_customerHome.php

Add a new customer

* Indicates required fields.

* Name	Luke Ban
* Street Address	3409 Sixth Ave
* City	Pittsburgh
* State	PA
* Zip Code	15260
* Age	Please enter a positive number!
* Gender	male
* marriage	yes
* Income	Please enter a positive number!
* Email	lukeban@pitt.edu
* Password	●●●

When all the inputs are valid, the system will check by email if this customer has already existed in the database. If not, this new customer will be added. If the insertion succeeds, a success message will be displayed.

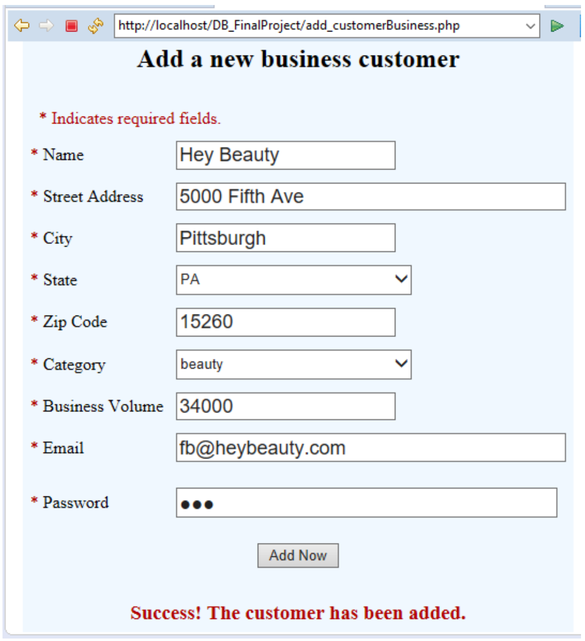

http://localhost/DB_FinalProject/add_customerBusiness.php

Add a new business customer

* Indicates required fields.

* Name	
* Street Address	
* City	
* State	
* Zip Code	00000
* Category	
* Business Volume	
* Email	xxxxx@xxx.com
* Password	

Add Now


http://localhost/DB_FinalProject/add_customerBusiness.php

Add a new business customer

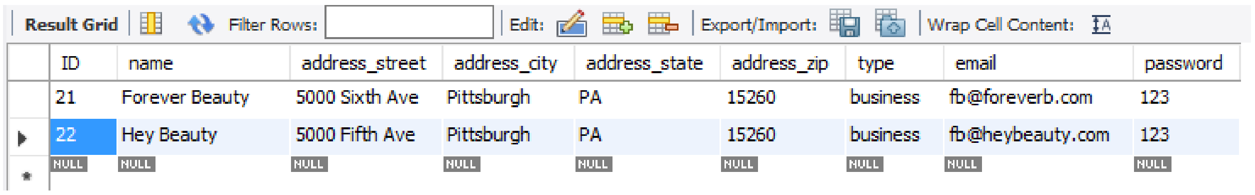
* Indicates required fields.

* Name	Hey Beauty
* Street Address	5000 Fifth Ave
* City	Pittsburgh
* State	PA
* Zip Code	15260
* Category	beauty
* Business Volume	34000
* Email	fb@heybeauty.com
* Password	●●●

Add Now

Success! The customer has been added.

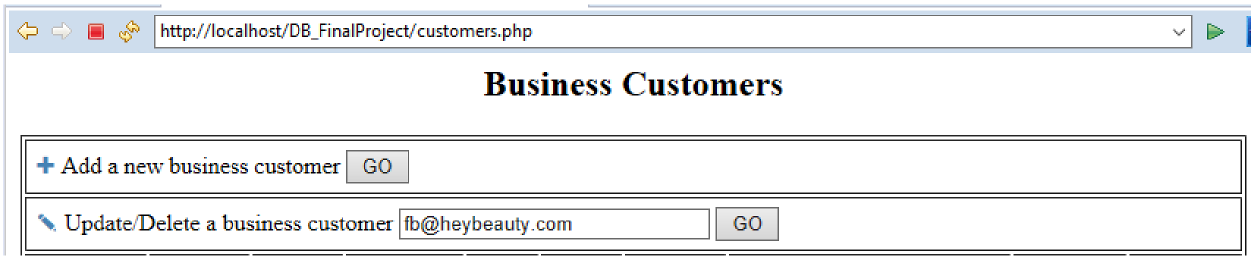
Check from the backend, the new business customer of “Hey Beauty” has been successfully added.


Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

ID	name	address_street	address_city	address_state	address_zip	type	email	password
21	Forever Beauty	5000 Sixth Ave	Pittsburgh	PA	15260	business	fb@foreverb.com	123
22	Hey Beauty	5000 Fifth Ave	Pittsburgh	PA	15260	business	fb@heybeauty.com	123
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Update a customer:

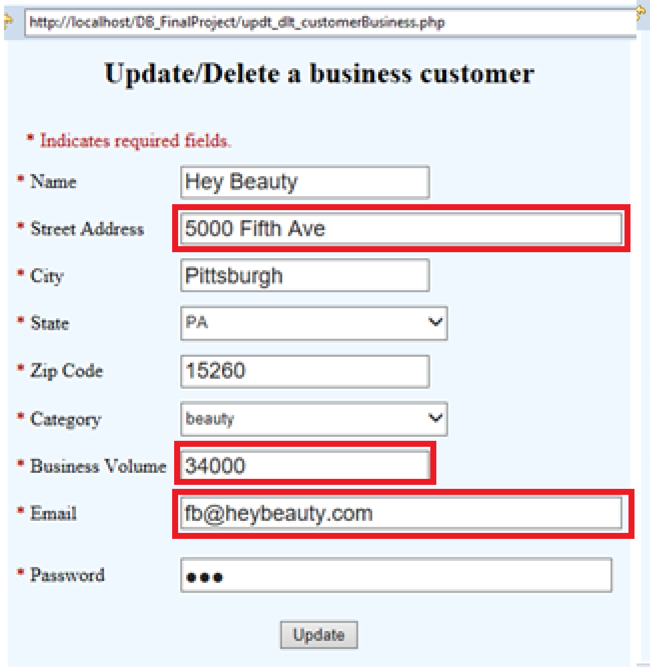
Different from the page of adding a customer, which starts with all input fields empty, the update/delete page will be fulfilled with the information in the input fields when the page is loaded. On the customer review page, the salesperson needs to input the email of the customer he/she wants to update or delete before the webheads to the update/delete page.


http://localhost/DB_FinalProject/customers.php

Business Customers

+ Add a new business customer	GO
Update/Delete a business customer fb@heybeauty.com	GO

The screenshot on the left side below is an update/delete page with fulfilled input fields when loaded. We can modify some information as the screenshot on the right sideshows.

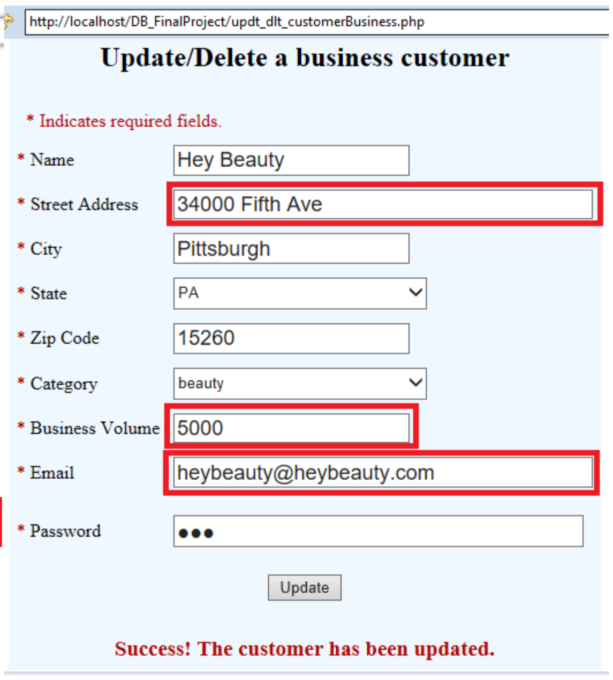


http://localhost/DB_FinalProject/updt_dlt_customerBusiness.php

Update/Delete a business customer

* Indicates required fields.

* Name	Hey Beauty
* Street Address	5000 Fifth Ave
* City	Pittsburgh
* State	PA
* Zip Code	15260
* Category	beauty
* Business Volume	34000
* Email	fb@heybeauty.com
* Password	●●●



http://localhost/DB_FinalProject/updt_dlt_customerBusiness.php

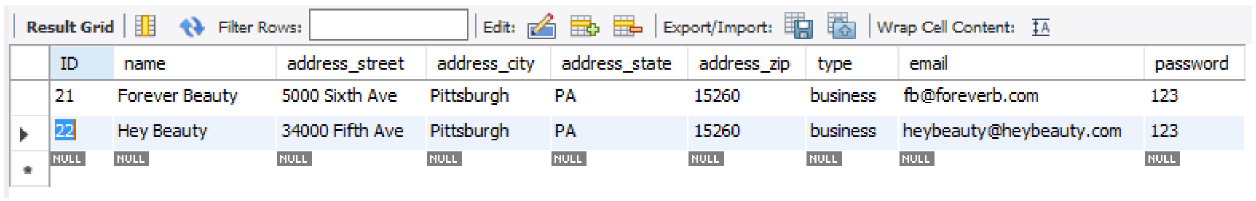
Update/Delete a business customer

* Indicates required fields.

* Name	Hey Beauty
* Street Address	34000 Fifth Ave
* City	Pittsburgh
* State	PA
* Zip Code	15260
* Category	beauty
* Business Volume	5000
* Email	heybeauty@heybeauty.com
* Password	●●●

Success! The customer has been updated.

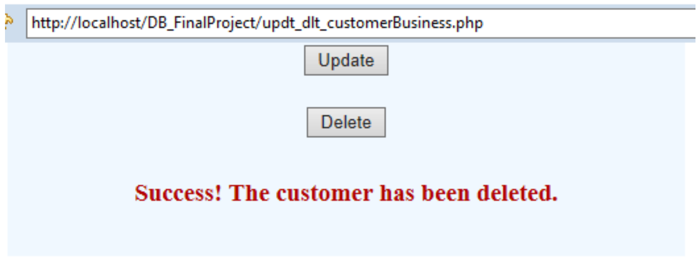
The information of this customer has been updated in the database when we check from the backend:



ID	name	address_street	address_city	address_state	address_zip	type	email	password
21	Forever Beauty	5000 Sixth Ave	Pittsburgh	PA	15260	business	fb@foreverb.com	123
22	Hey Beauty	34000 Fifth Ave	Pittsburgh	PA	15260	business	heybeauty@heybeauty.com	123
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Delete a customer:

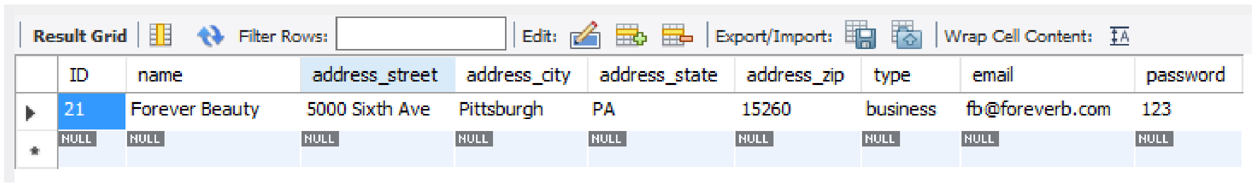
The salesperson can also delete the customer by click on the “Delete” button:



http://localhost/DB_FinalProject/updt_dlt_customerBusiness.php

Success! The customer has been deleted.

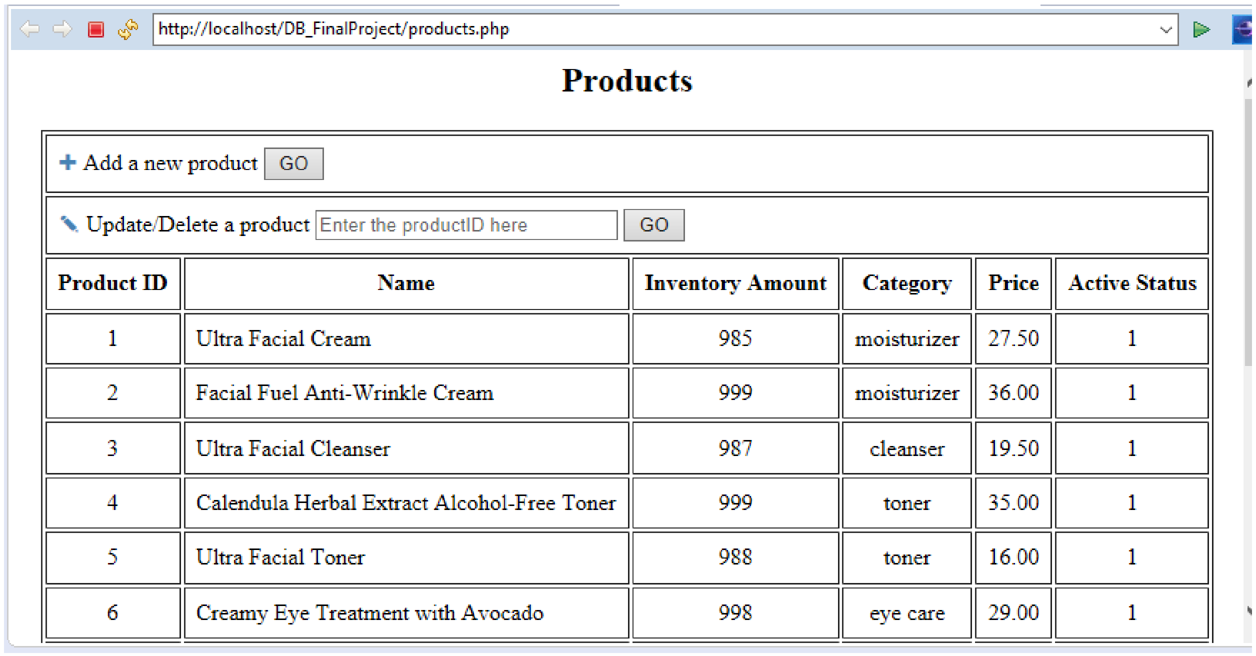
The customer of “Hey Beauty” does not exist anymore in the system:


Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

ID	name	address_street	address_city	address_state	address_zip	type	email	password
21	Forever Beauty	5000 Sixth Ave	Pittsburgh	PA	15260	business	fb@foreverb.com	123
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

2) Add/Update/Delete Products

Pages of products and pages of stores perform functions in a similar way as the pages of customers. Input is always validated, and forms are always sticky.


http://localhost/DB_FinalProject/products.php

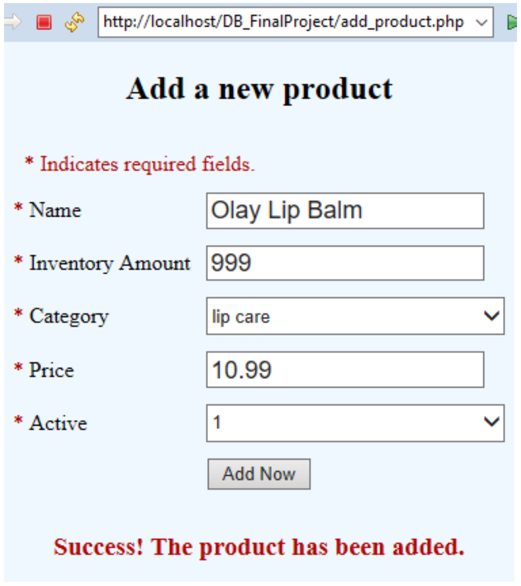
Products

+ Add a new product GO

Update/Delete a product Enter the productID here GO

Product ID	Name	Inventory Amount	Category	Price	Active Status
1	Ultra Facial Cream	985	moisturizer	27.50	1
2	Facial Fuel Anti-Wrinkle Cream	999	moisturizer	36.00	1
3	Ultra Facial Cleanser	987	cleanser	19.50	1
4	Calendula Herbal Extract Alcohol-Free Toner	999	toner	35.00	1
5	Ultra Facial Toner	988	toner	16.00	1
6	Creamy Eye Treatment with Avocado	998	eye care	29.00	1

Add a new product:


http://localhost/DB_FinalProject/add_product.php

Add a new product

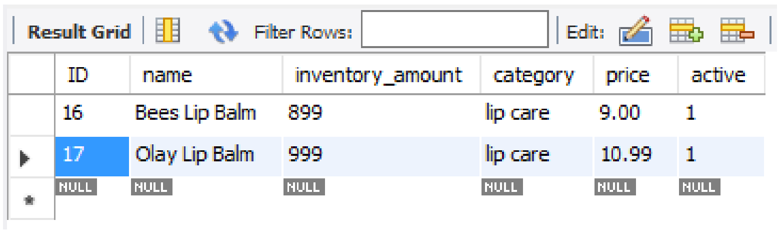
* Indicates required fields.

* Name	Olay Lip Balm
* Inventory Amount	999
* Category	lip care
* Price	10.99
* Active	1

Add Now

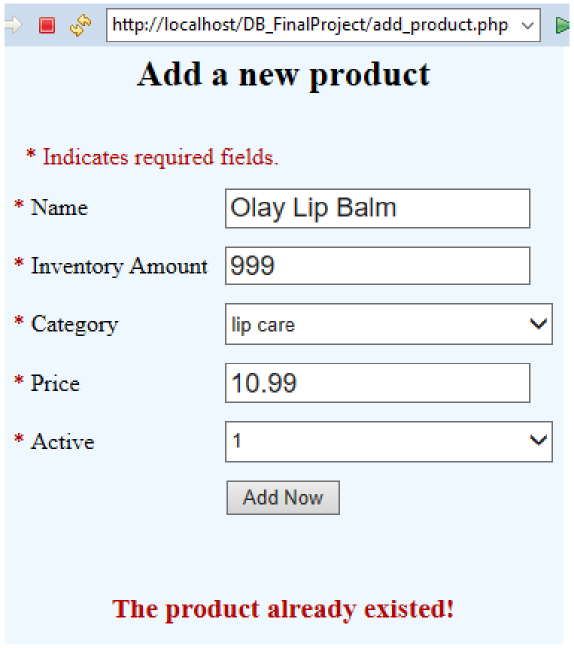
Success! The product has been added.

When all the inputs are valid, both the frontend and backend indicate that the new product of "Olay Lip Balm" has been added to the database.



ID	name	inventory_amount	category	price	active
16	Bees Lip Balm	899	lip care	9.00	1
17	Olay Lip Balm	999	lip care	10.99	1
NULL	NULL	NULL	NULL	NULL	NULL

The system disallows salespersons to add duplicated products. Each product will be matching by its name with the products in the database before it is added. The screenshot below shows that the system displays an error message and prevents salespersons from adding a new product that is already in the database.



Add a new product

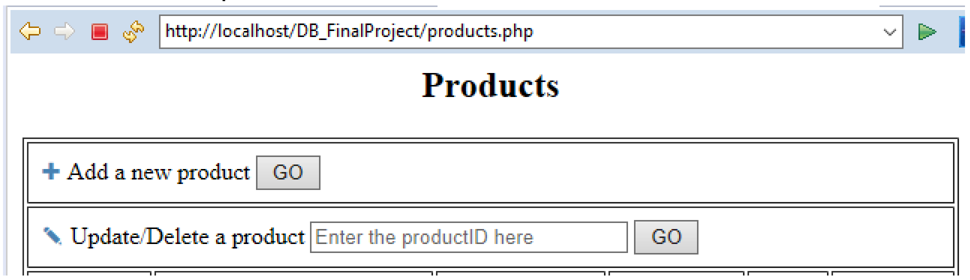
* Indicates required fields.

* Name	Olay Lip Balm
* Inventory Amount	999
* Category	lip care
* Price	10.99
* Active	1

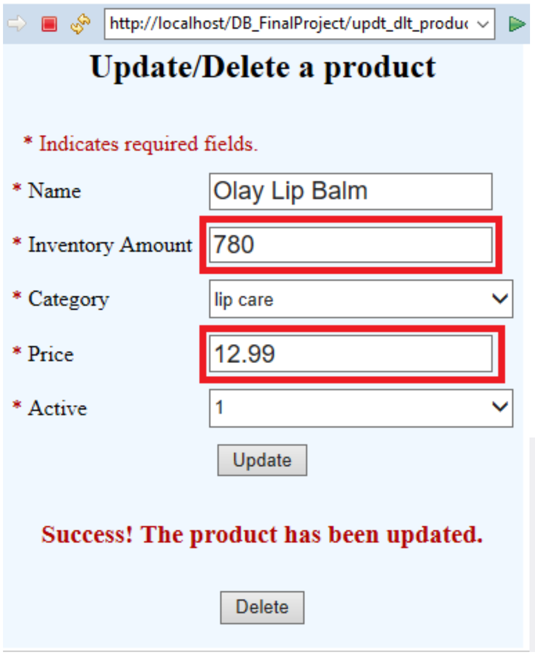
The product already existed!

Update a product:

To update or delete a product, the salesperson needs to get the information of the product by entering its product ID. But what if the person has no idea about the product ID? It can be easily found on the products review page by hitting CTRL + F keys and searching by any characters contained in the product name.



In the demonstration below, we changed the inventory amount and price of “Olay Lip Balm” from the backend, and information at the backend has been updated.



http://localhost/DB_FinalProject/updt_dlt_produc

Update/Delete a product

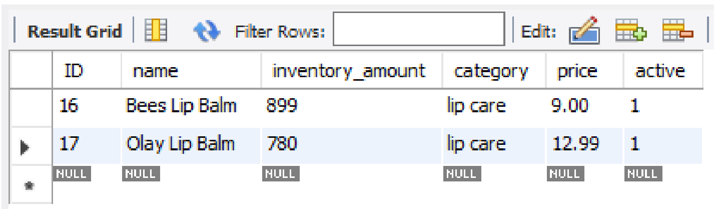
* Indicates required fields.

* Name	Olay Lip Balm
* Inventory Amount	780
* Category	lip care
* Price	12.99
* Active	1

Update

Success! The product has been updated.

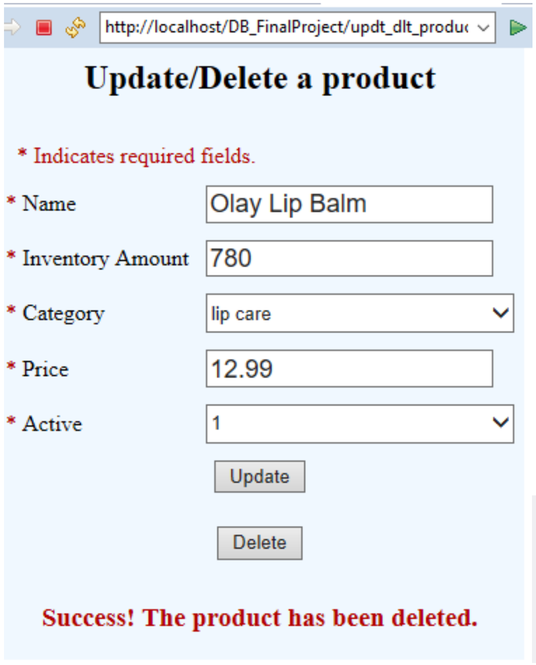
Delete



ID	name	inventory_amount	category	price	active
16	Bees Lip Balm	899	lip care	9.00	1
17	Olay Lip Balm	780	lip care	12.99	1
NULL	NULL	NULL	NULL	NULL	NULL

Delete a product:

Delete is easy with one click on the “Delete” button. The product of “Olay Lip Balm” has been removed from the table of product.



http://localhost/DB_FinalProject/updt_dlt_produc

Update/Delete a product

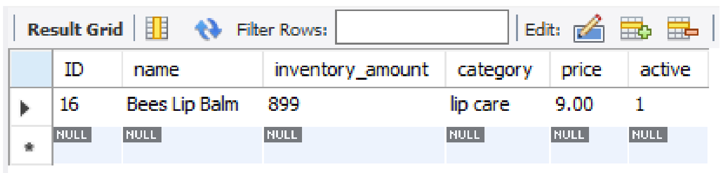
* Indicates required fields.

* Name	Olay Lip Balm
* Inventory Amount	780
* Category	lip care
* Price	12.99
* Active	1

Update

Delete

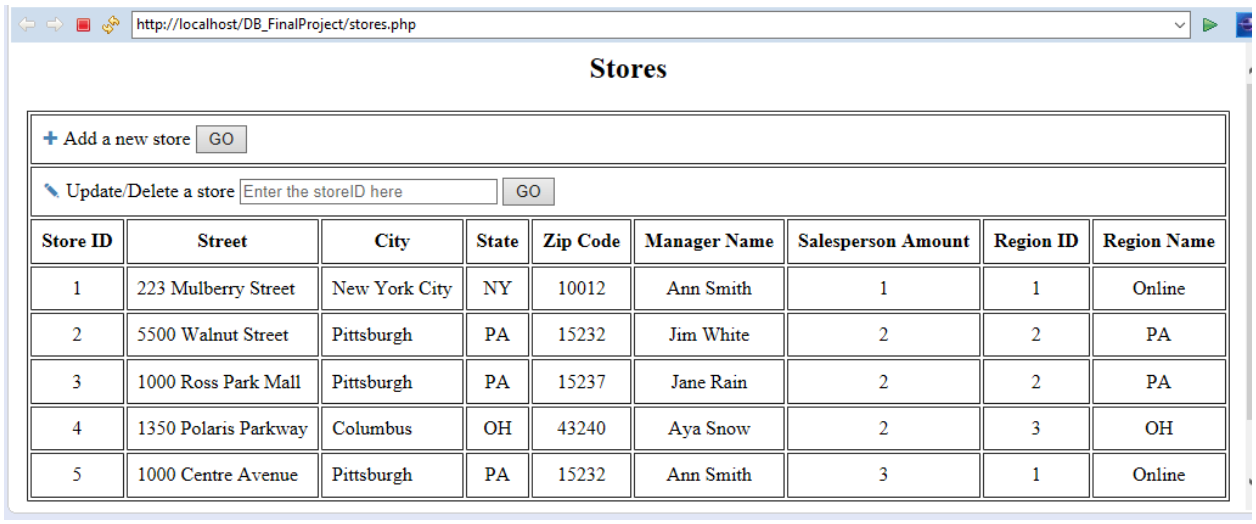
Success! The product has been deleted.



ID	name	inventory_amount	category	price	active
16	Bees Lip Balm	899	lip care	9.00	1
NULL	NULL	NULL	NULL	NULL	NULL

3) Add/Update/Delete Stores

Operations on store information are only allowed to be conducted by managers. A manager can view all the information of all the stores on the review page and can add, update or delete a store.



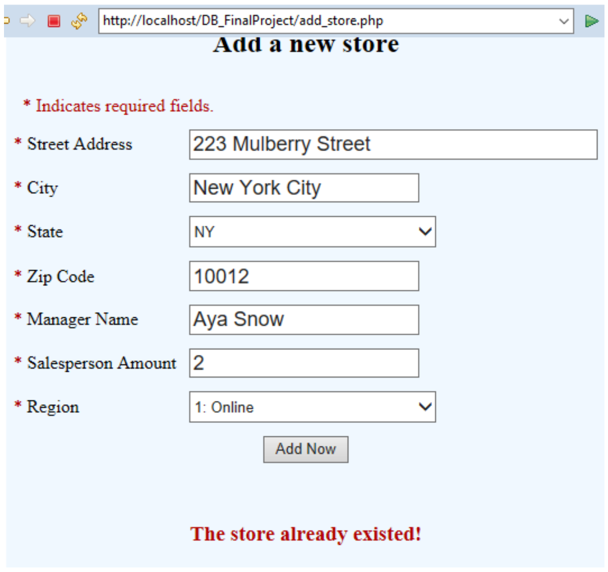
http://localhost/DB_FinalProject/stores.php

Stores

Store ID	Street	City	State	Zip Code	Manager Name	Salesperson Amount	Region ID	Region Name
1	223 Mulberry Street	New York City	NY	10012	Ann Smith	1	1	Online
2	5500 Walnut Street	Pittsburgh	PA	15232	Jim White	2	2	PA
3	1000 Ross Park Mall	Pittsburgh	PA	15237	Jane Rain	2	2	PA
4	1350 Polaris Parkway	Columbus	OH	43240	Aya Snow	2	3	OH
5	1000 Centre Avenue	Pittsburgh	PA	15232	Ann Smith	3	1	Online

Add a new store:

Any new store to be added will be matching by its address with the stores in the database. The store located in NYC in the screenshot on the left side below is already in the system, so an error message appears to inform the prevention of adding it. Once we change the address, the store is successfully added.



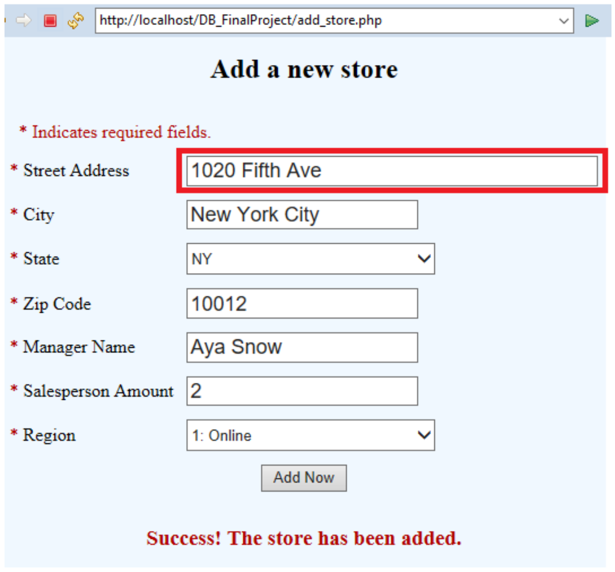
http://localhost/DB_FinalProject/add_store.php

Add a new store

* Indicates required fields.

* Street Address	223 Mulberry Street
* City	New York City
* State	NY
* Zip Code	10012
* Manager Name	Aya Snow
* Salesperson Amount	2
* Region	1: Online

The store already existed!



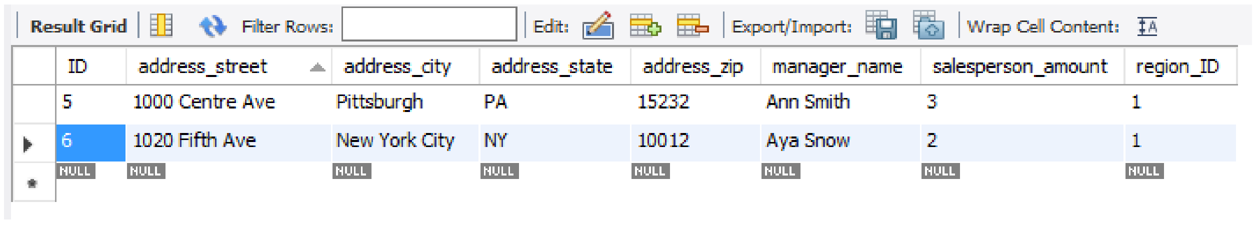
http://localhost/DB_FinalProject/add_store.php

Add a new store

* Indicates required fields.

* Street Address	1020 Fifth Ave
* City	New York City
* State	NY
* Zip Code	10012
* Manager Name	Aya Snow
* Salesperson Amount	2
* Region	1: Online

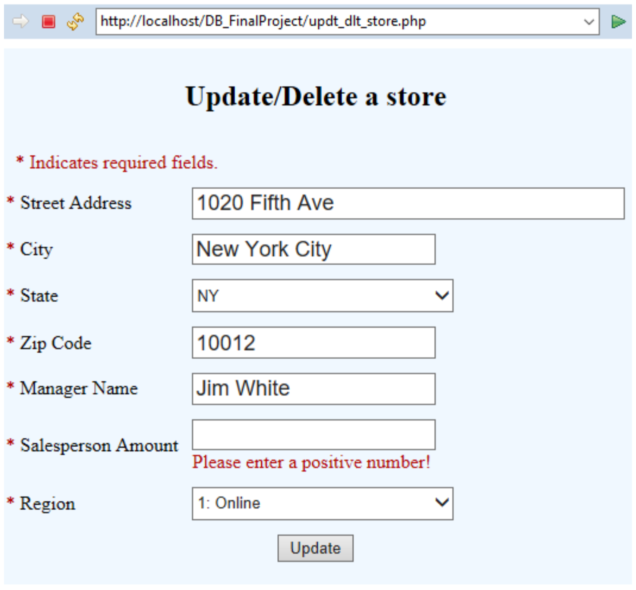
Success! The store has been added.


Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

	ID	address_street	address_city	address_state	address_zip	manager_name	salesperson_amount	region_ID
	5	1000 Centre Ave	Pittsburgh	PA	15232	Ann Smith	3	1
▶	6	1020 Fifth Ave	New York City	NY	10012	Aya Snow	2	1
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Update a store:

Update the manager and salesperson amount for a store. If the input of the salesperson amount is a negative number, the system will tell it is unsafe data and will prevent sending it to the backend. The only safe will be sent and updated in the system.

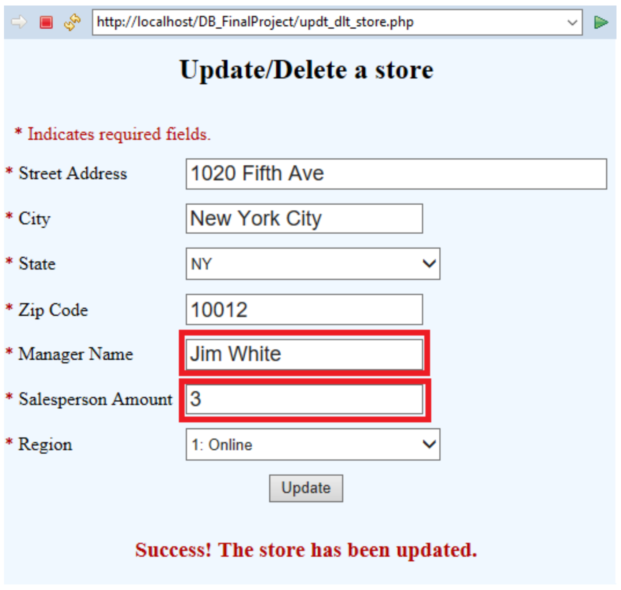

http://localhost/DB_FinalProject/updt_dlt_store.php

Update/Delete a store

* Indicates required fields.

* Street Address	1020 Fifth Ave
* City	New York City
* State	NY
* Zip Code	10012
* Manager Name	Jim White
* Salesperson Amount	Please enter a positive number!
* Region	1: Online

Update


http://localhost/DB_FinalProject/updt_dlt_store.php

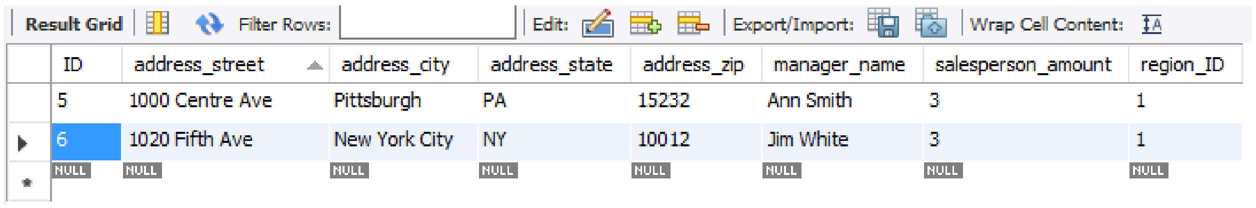
Update/Delete a store

* Indicates required fields.

* Street Address	1020 Fifth Ave
* City	New York City
* State	NY
* Zip Code	10012
* Manager Name	Jim White
* Salesperson Amount	3
* Region	1: Online

Update

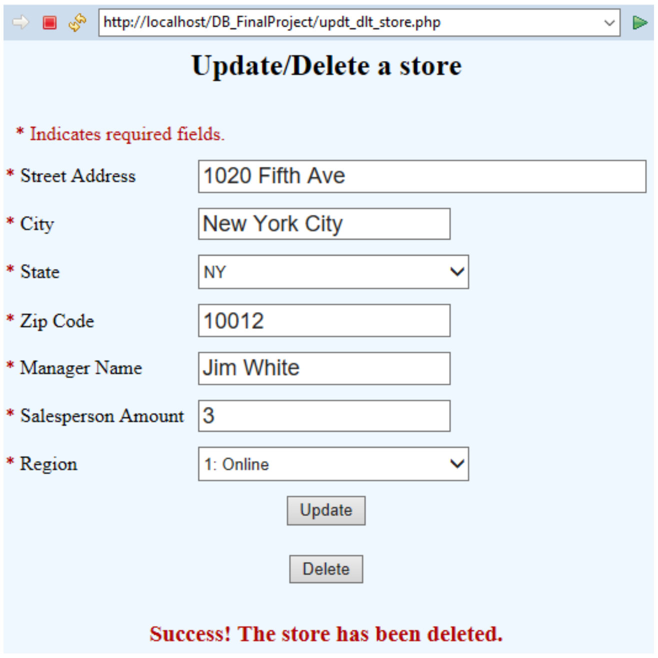
Success! The store has been updated.


Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

	ID	address_street	address_city	address_state	address_zip	manager_name	salesperson_amount	region_ID
	5	1000 Centre Ave	Pittsburgh	PA	15232	Ann Smith	3	1
▶	6	1020 Fifth Ave	New York City	NY	10012	Jim White	3	1
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Delete a store:

Click on the "Delete" button will remove the store from the table of the store. The store located at 1020 Fifth Ave, New York City, NY, 10012 is no longer in the system when we check from the backend.



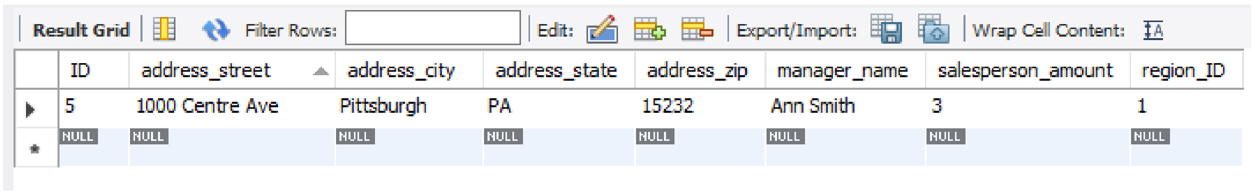
http://localhost/DB_FinalProject/updt_dlt_store.php

Update/Delete a store

* Indicates required fields.

* Street Address	1020 Fifth Ave
* City	New York City
* State	NY
* Zip Code	10012
* Manager Name	Jim White
* Salesperson Amount	3
* Region	1: Online

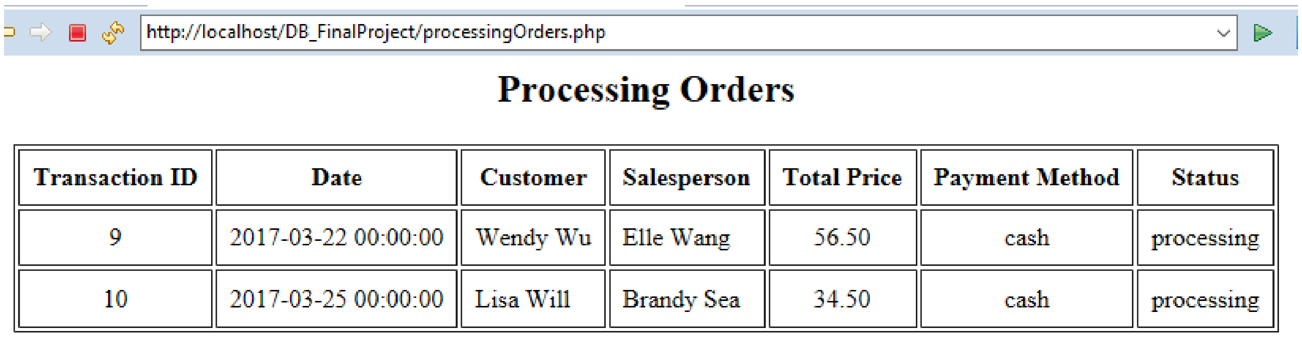
Success! The store has been deleted.



ID	address_street	address_city	address_state	address_zip	manager_name	salesperson_amount	region_ID
5	1000 Centre Ave	Pittsburgh	PA	15232	Ann Smith	3	1
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

4) Check Processing Orders

Only managers can check orders under processing status. The page of processing orders allows managers to review all the processing orders with all attributes in the table of *transaction*.



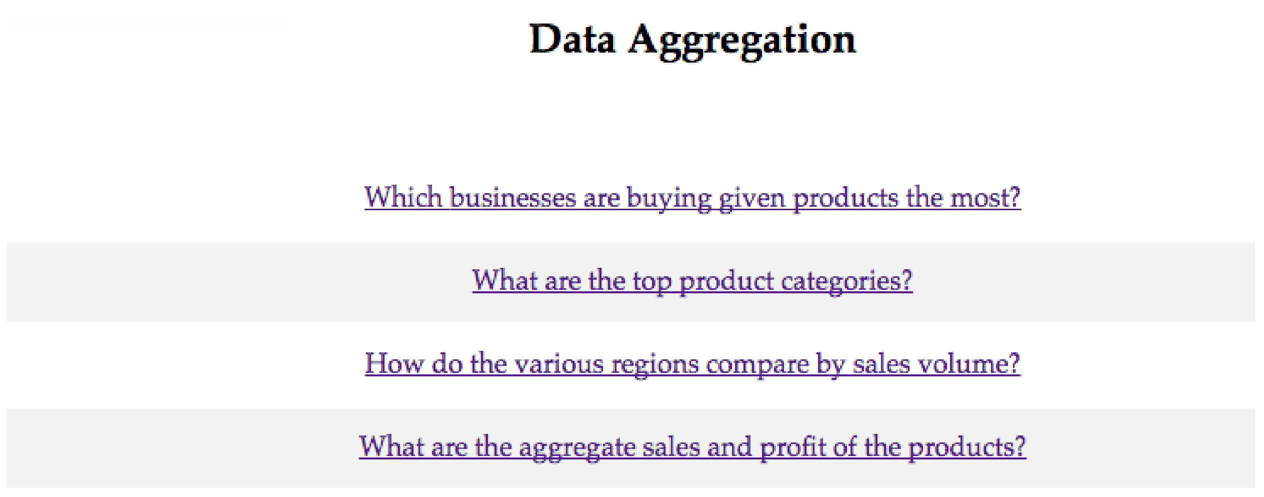
http://localhost/DB_FinalProject/processingOrders.php

Processing Orders

Transaction ID	Date	Customer	Salesperson	Total Price	Payment Method	Status
9	2017-03-22 00:00:00	Wendy Wu	Elle Wang	56.50	cash	processing
10	2017-03-25 00:00:00	Lisa Will	Brandy Sea	34.50	cash	processing

3.3 Data Aggregation

We have generated four tables to demonstrate data aggregation:



Data Aggregation

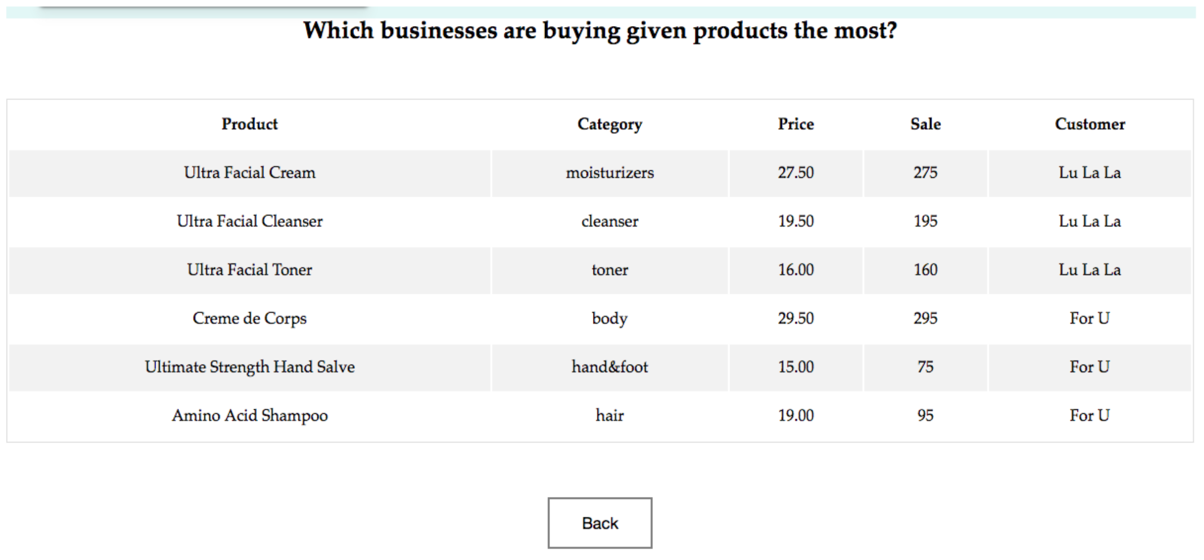
Which businesses are buying given products the most?

What are the top product categories?

How do the various regions compare by sales volume?

What are the aggregate sales and profit of the products?

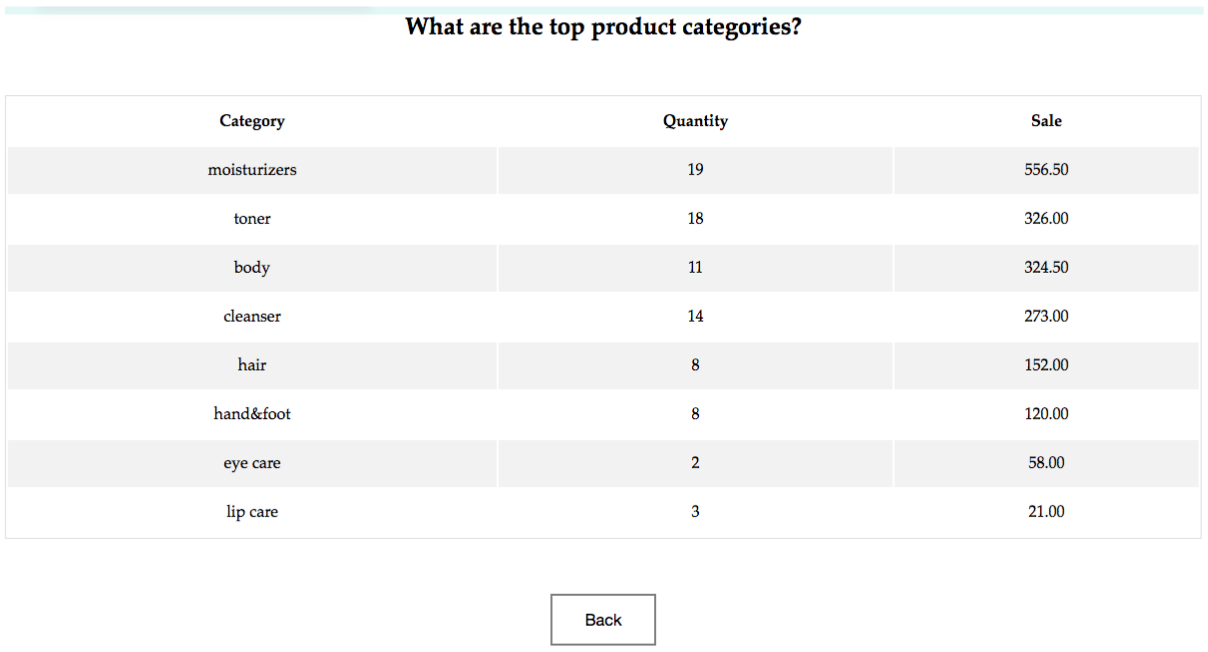
For the first question, the table below shows the business-type customer who bought the products the most:



Which businesses are buying given products the most?

Product	Category	Price	Sale	Customer
Ultra Facial Cream	moisturizers	27.50	275	Lu La La
Ultra Facial Cleanser	cleanser	19.50	195	Lu La La
Ultra Facial Toner	toner	16.00	160	Lu La La
Creme de Corps	body	29.50	295	For U
Ultimate Strength Hand Salve	hand&foot	15.00	75	For U
Amino Acid Shampoo	hair	19.00	95	For U

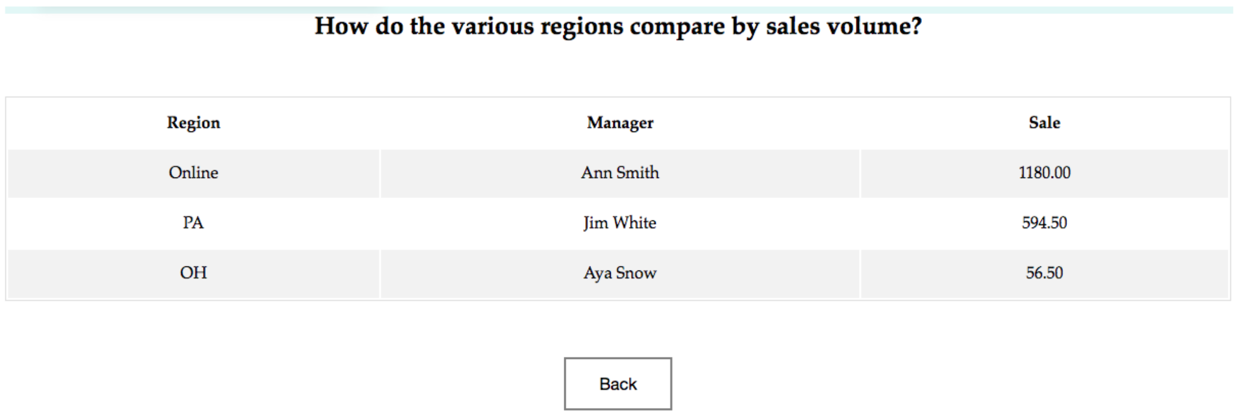
For the second question, the table below demonstrates the hottest product category:



What are the top product categories?

Category	Quantity	Sale
moisturizers	19	556.50
toner	18	326.00
body	11	324.50
cleanser	14	273.00
hair	8	152.00
hand&foot	8	120.00
eye care	2	58.00
lip care	3	21.00

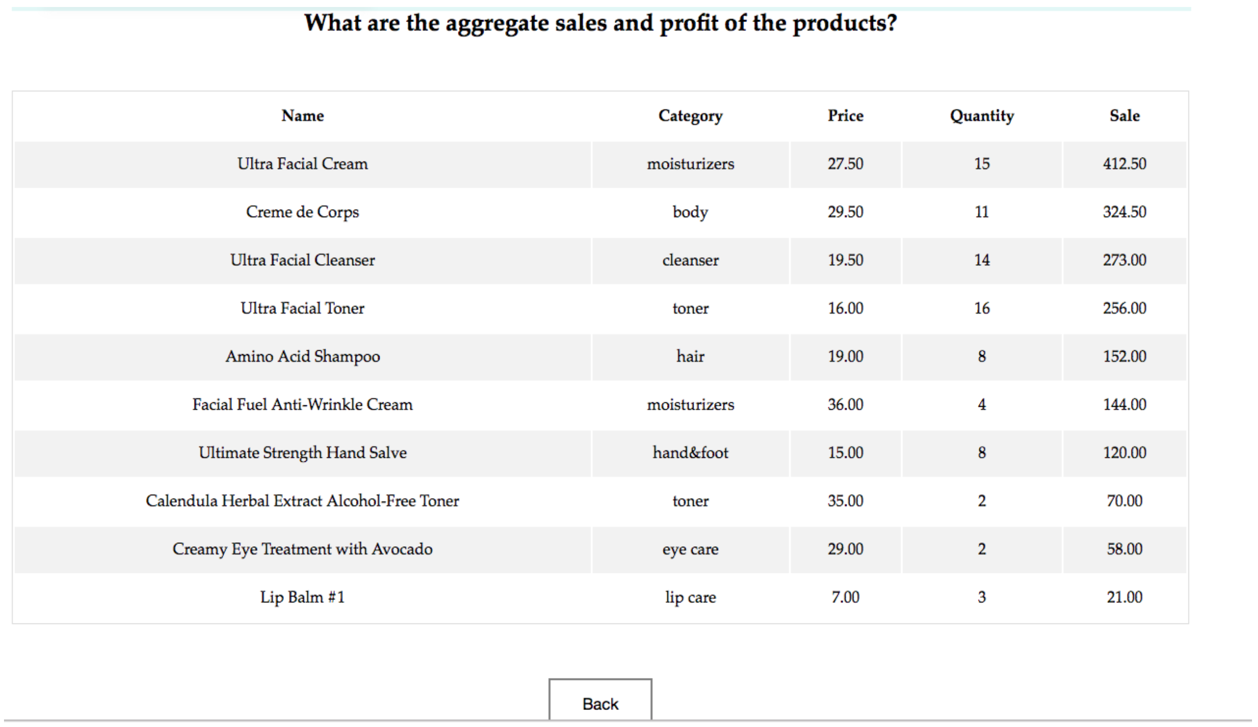
For the third question, the table below illustrates the different sales volume of each region:



How do the various regions compare by sales volume?

Region	Manager	Sale
Online	Ann Smith	1180.00
PA	Jim White	594.50
OH	Aya Snow	56.50

For the fourth one, the table below would provide a clear perspective of which products have made a significant profit:



What are the aggregate sales and profit of the products?

Name	Category	Price	Quantity	Sale
Ultra Facial Cream	moisturizers	27.50	15	412.50
Creme de Corps	body	29.50	11	324.50
Ultra Facial Cleanser	cleanser	19.50	14	273.00
Ultra Facial Toner	toner	16.00	16	256.00
Amino Acid Shampoo	hair	19.00	8	152.00
Facial Fuel Anti-Wrinkle Cream	moisturizers	36.00	4	144.00
Ultimate Strength Hand Salve	hand&foot	15.00	8	120.00
Calendula Herbal Extract Alcohol-Free Toner	toner	35.00	2	70.00
Creamy Eye Treatment with Avocado	eye care	29.00	2	58.00
Lip Balm #1	lip care	7.00	3	21.00

4. Conclusion

From all the frontend and backend design, now the website can fulfill the functions like browsing products, submitting transactions, and viewing aggregation information. The backend relational database design also can update inventory information automatically and justify BCNF to remove any redundancy. There are still some limitations in our project: first, again, different stores using the same inventory is still a problem that could be improved to meet the real situations. Second, we do not have any discount system for different customers or products, which could be added to our current design and make the system more complex.