CPSC 50900 Database Systems Project

A Project Report

On

Submitted By

S.No	Student Name	Student ID
1.	Nishanth Nutula	L30062798

Under the guidance of

Prof. Mathew Clavelli

Version: 1 **Date: 14/12/2021**

Description

The “Laptop Store system” is developed according to the current needs in different Fields. This is an online shopping Website which provides facility for purchasing Laptops and many more items. So by using these system users who want to purchase some products. Firstly, Register an account on this website. Secondly, log in through their Username and Password, and then select items that they want to purchase and add them to cart and finally checkout by giving payment details. So by using this portal users can easily purchase laptops from their home.

Scope

The Score of the Laptop Store Online Shopping application is to design and develop a graphical based user friendly system, which will consist of laptop products that are used in our daily life. For example – desktops, Laptops, pen drives, external hardware and many more. So by this automated system a user can make a purchase according to his needs then it is only a mouse click away to purchase these products.

Objectives

The Laptop plays an important role in our daily life. Anything we want we can get only with one mouse click. Speed, reliability, and accuracy of the computer make it a powerful tool for different purposes. A very important and basic need of today’s modern business world is the quick availability and processing of information using a computer. One can easily get the type of required information within a fraction of a second. The project that I have taken is also in this category which is used in our daily life whenever we want to purchase some items we can easily get them at our home.

Functional Requirements

Functional Requirements		
S.No	**Name**	**Description**
LSFR-1	Registration	users to Register in to the application
LSFR-2	Login	Admin or users to login with the credentials
LSFR-3	Forgot password	Application must allow users to reset their password for any reason
LSFR-4	profile	User can update his own profile
LSFR-5	Add Laptops	Admin will add laptops
LSFR-6	Manage Laptops	Admin can View/Edit/Delete Laptops and Desktops
LSFR-7	Manage Bookings	Admin will check all booking and payments done by customers
LSFR-8	Home Page	This page contains the main dashboard with Products , cart, payment, edit profile, search products, contact us etc.
LSFR-9	Search laptops	Customer can search desired laptop by company names
LSFR-10	Add to cart	Customer can add items to cart
LSFR-11	payment	Customer will make payment after his selection

LSFR-12	My orders	Customer can check his orders history
LSFR-13	Logout	Admin and customer can logout from website

Non-functional requirements

Non-Functional Requirements		
S. No	**Name**	**Description**
LS NFR-1	Usability	The user must be able to open the website on any operating system and must run on all the devices.
LS NFR-2	Availability	This system must be available at all times whenever the user wants to access it with the browser.
LS NFR-3	Security	To avoid unauthorized access to the device, the system contains a username and password. only users with correct passwords and usernames should be able to login to see the user's page.
LS NFR-4	Performance	The website provides adequate performance requirements.
LS NFR-5	User Experience	The web application must provide users with a user friendly looking and easy to use type user interface for the users.
LS NFR-6	Efficiency	Any interface between a user and the automated system shall have a maximum response time of 2-5 seconds.
LS NFR-8	Scalability	The website attendance limit to support 1000 users at a time.
LS NFR-10	Portability	The website runs on any device like laptops, mobile devices and tabs.

Other Requirements

Software Requirements:

- PHP
- Visual Studio Code (IDE), Notepad++. Photoshop
- MySql
- Xampp Server
- Web Technologies (HTML5, CSS3, Java Script, Bootstrap 3)
- Windows 7 and above

Hardware Components:

- Processor required minimum – i3
- Hard Disk required minimum – 5 GB and greater than this
- Minimum Memory – 2GB RAM

Relational Database Design Process

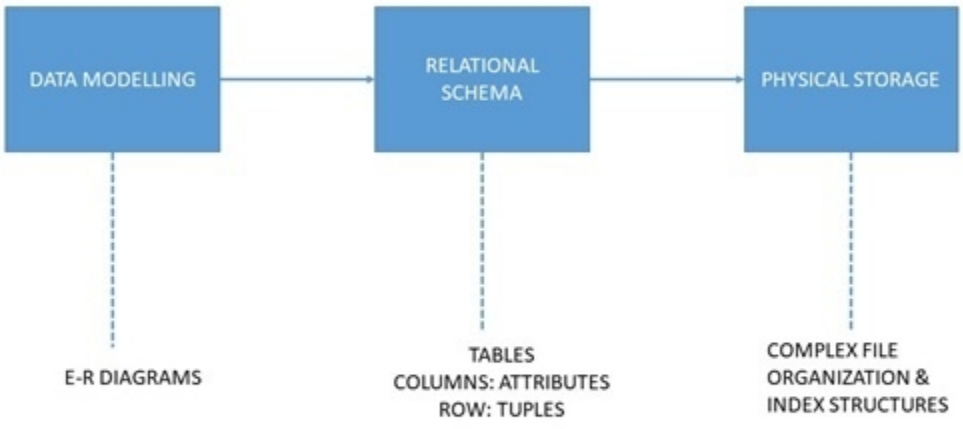


Table Names & attributes:

Admin: id, email, pass

Customer: id, name, email, phone, pass, status

Product: id, name, brand, price, pic

Orders: oid, email, name, brand, price, payment_id, status

Card: id, payment_id, cname, cno, cvv, expmonth, expyear

Admin					
	Column Name	**Data Type**	**Length**	**Nullable**	**Description**
1	id	INT	10	N	Contains Admin id
2	email	VARCHAR	100	N	Username for admin login
3	pass	VARCHAR	50	N	Password for admin login

Customer Registration					
	Column Name	**Data Type**	**Length**	**Nullable**	**Description**
1	id	INT	10	N	Contains renter/user id
2	name	VARCHAR	50	N	name for user login
3	email	VARCHAR	50	N	email for user login
4	phone	VARCHAR	50	N	Phone number
5	pass	VARCHAR	50	N	Password for user
6	status	VARCHAR	50	N	User account status(active/blocked)

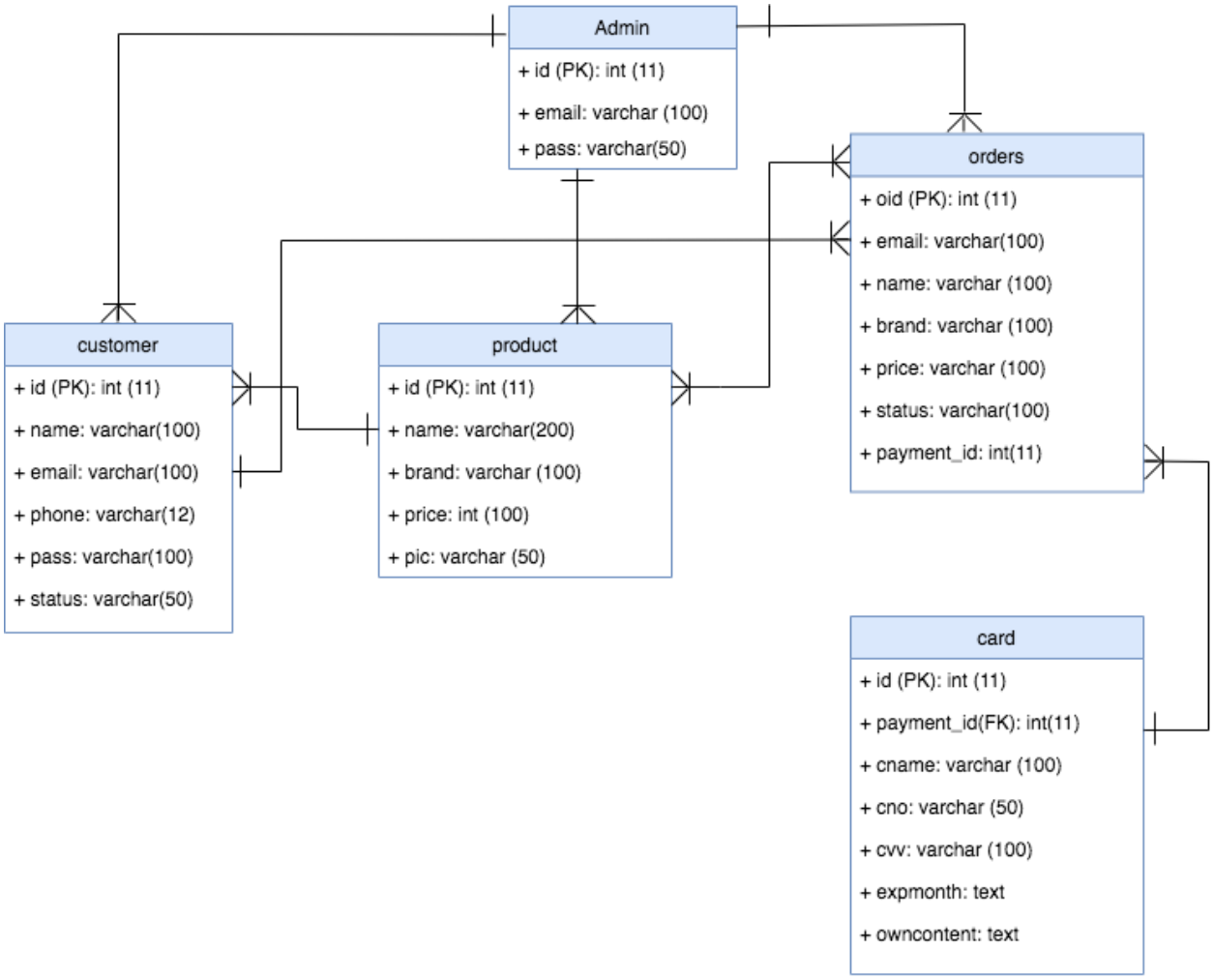
product					
	Column Name	**Data Type**	**Length**	**Nullable**	**Description**
1	id	INT	100	N	Contains product id
2	name	VARCHAR	200	N	product name

3	brand	VARCHAR	100	N	Brand name of product
4	price	INT	100	N	Product price
5	pic	VARCHAR	200	N	Product image

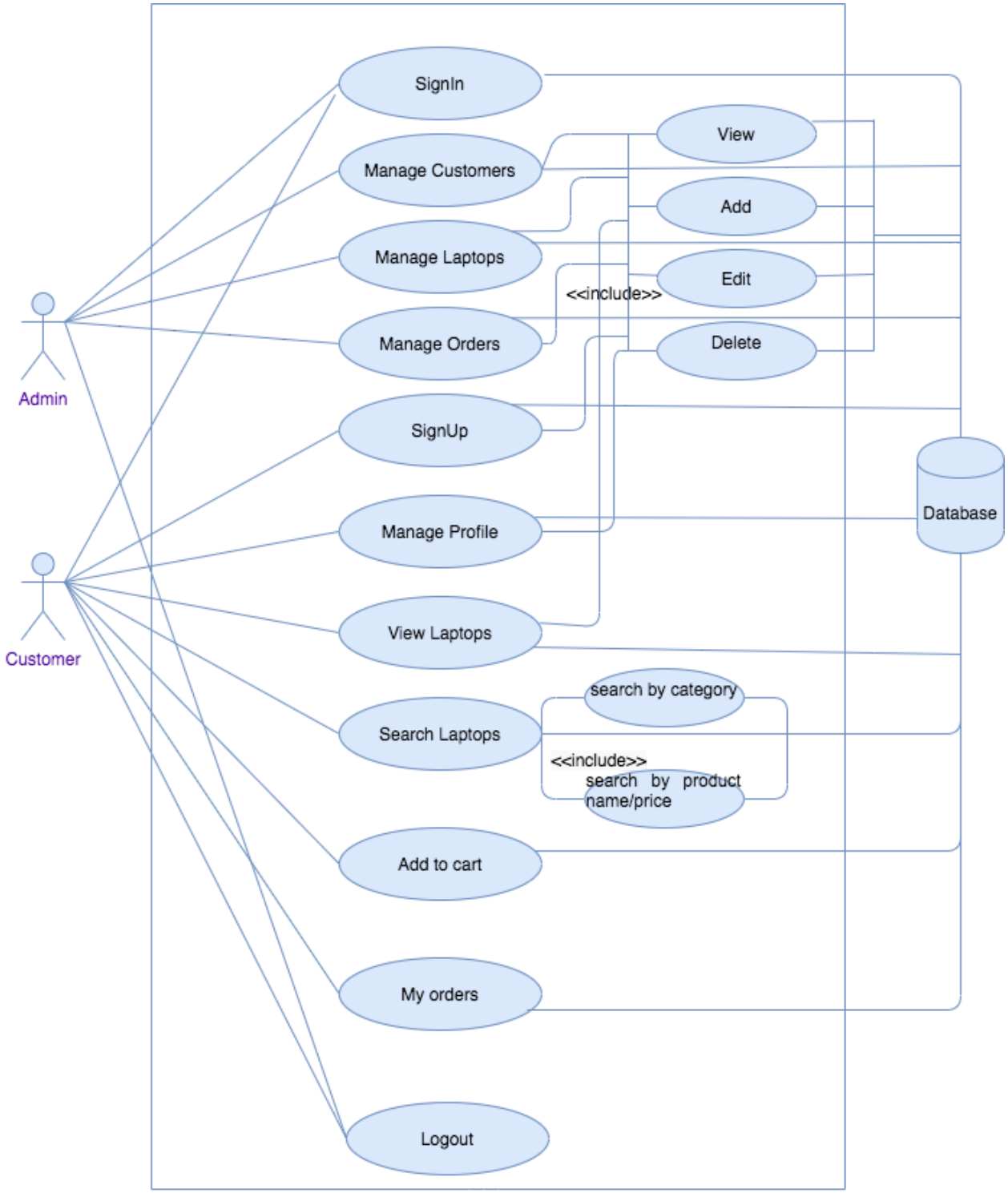
orders					
	Column Name	**Data Type**	**Length**	**Nullable**	**Description**
1	oid	INT	11	N	Contains order id
2	email	VARCHAR	100	N	Email of customer
3	name	VARCHAR	100	N	customer name
4	brand	VARCHAR	100	N	Brand name of product
5	price	VARCHAR	100	N	Product price
6	status	VARCHAR	100	N	Order status
7	payment_id	INT	11	N	Payment transaction id

card					
	Column Name	**Data Type**	**Length**	**Nullable**	**Description**
1	id	INT	11	N	Contains payment refernce id
2	payment_id	VARCHAR	100	N	Payment transaction id
3	cname	VARCHAR	100	N	customer name on card
4	cno	VARCHAR	100	N	Card number on card
5	cvv	VARCHAR	100	N	Security code
6	expmonth	INT	11	N	Card expiry month
7	expyear	INT	11	N	Card expiry year

ERD Diagram


Admin
+ id (PK): int (11)
+ email: varchar (100)
+ pass: varchar(50)
orders
+ oid (PK): int (11)
+ email: varchar(100)
+ name: varchar (100)
+ brand: varchar (100)
+ price: varchar (100)
+ status: varchar(100)
+ payment_id: int(11)
customer
+ id (PK): int (11)
+ name: varchar(100)
+ email: varchar(100)
+ phone: varchar(12)
+ pass: varchar(100)
+ status: varchar(50)
product
+ id (PK): int (11)
+ name: varchar(200)
+ brand: varchar (100)
+ price: int (100)
+ pic: varchar (50)
card
+ id (PK): int (11)
+ payment_id(FK): int(11)
+ cname: varchar (100)
+ cno: varchar (50)
+ cvv: varchar (100)
+ expmonth: text
+ owncontent: text

Use Case Diagram



Data Sources

In my application we have total five entities (admin, customer, products,orders and card)

Admin: id, email, pass

Customer: id, name, email, phone, pass, status

Product: id, name, brand, price, pic

Orders: oid, email, name, brand, price, payment_id, status

Card: id, payment_id, cname, cno, cvv, expmonth, expyear

In my example when a user places an order the purchase details will be stored in orders tables which are present in the laptop database. We can retrieve the information by making an api call and the output would be in either xml or json format.

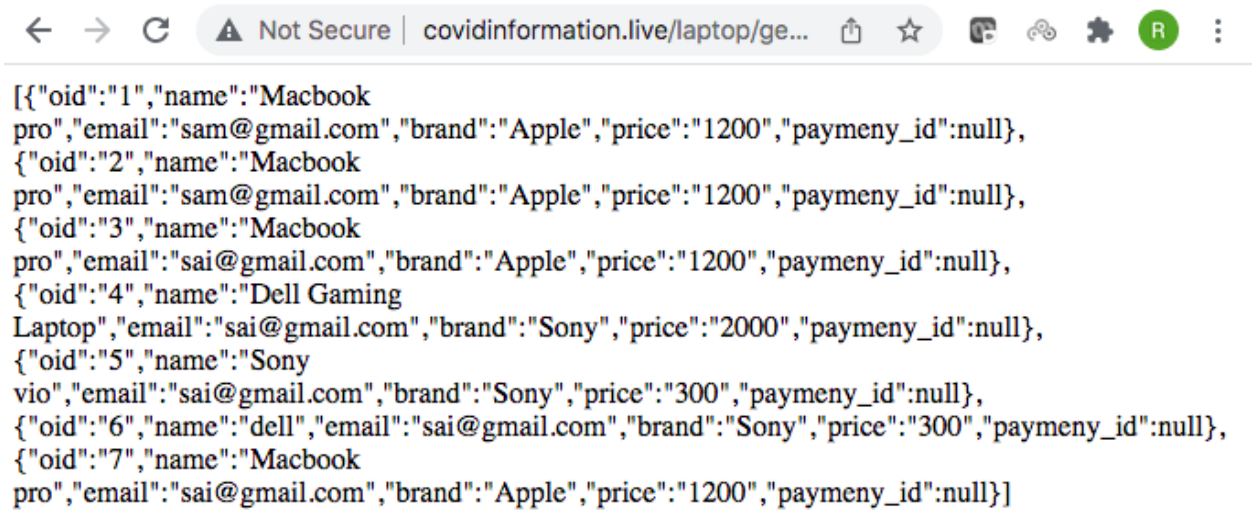
Web API Call

URL Method	http://localhost/laptop/getorders.php GET
Call	Get All customer Orders
Parameters	None
Response	Case1 { [{"oid":"1","name":"Macbook pro","email":"sam@gmail.com","brand":"Apple","price":"1200","paymeny_id":null},{"oid":"2","name":"Macbook pro","email":"sam@gmail.com","brand":"Apple","price":"1200","paymeny_id":null},{"oid":"3","name":"Macbook pro","email":"sai@gmail.com","brand":"Apple","price":"1200","paymeny_id":null},{"oid":"4","name":"Dell Gaming Laptop","email":"sai@gmail.com","brand":"Sony","price":"2000","paymeny_id":null},{"oid":"5","name":"Sony vio","email":"sai@gmail.com","brand":"Sony","price":"300","paymeny_id":null},{"oid":"6","name":"dell","email":"sai@gmail.com","brand":"Sony","price":"300","paymeny_id":null},{"oid":"7","name":"Macbook

	pro","email":"sai@gmail.com","brand":"Apple","price":"1200","paymeny_id":null}]
	}
	Case2
	{
	“Status”: “ERROR”,
	“Message”: “Server Issue”
	}
	Case3
	{
	“Status”: “ERROR”,
	“Message”: “No orders Found”
	}

Keys	“Status” Status of the response Possible values:“OK”, “WRONG”,“ERROR” Mandatory

Output Response Screenshot



Php code for my orders

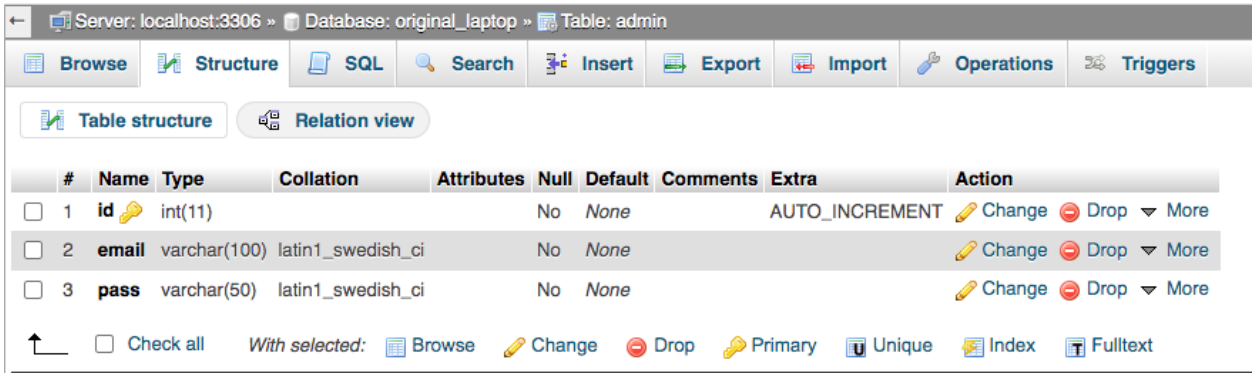
```
<?php
//Database connection
include 'config.php';
//sql query
$query ="SELECT * from orders" ;
$r = mysqli_query($conn,$query);
$rows = array();
while($row = mysqli_fetch_assoc($r)) {
$rows[] = array('oid' => $row['oid'],'name' => $row['name'],'email' =>
$row['email'],'brand' => $row['brand'],'price' => $row['price'],'paymeny_id' =>
$row['paymeny_id']);
}
echo json_encode($rows);
mysqli_close($conn);
?>
```

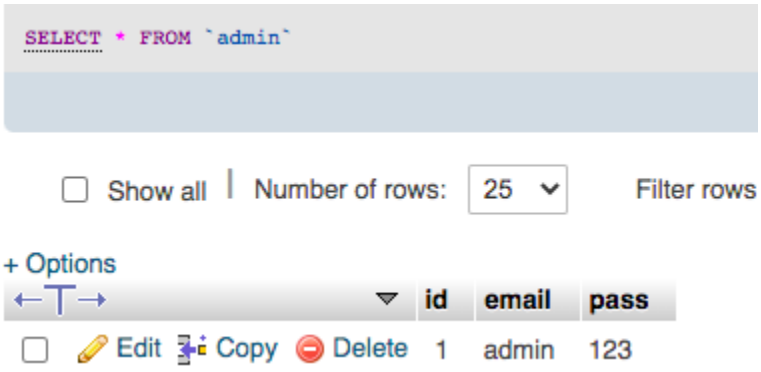
Data Definition Language Scripts

-- Table structure for table `admin` --

```
CREATE TABLE `admin`
( `id` int(11) NOT NULL,
`email` varchar(100) NOT NULL,
`pass` varchar(100) NOT NULL
);
```

Admin					
	Column Name	**Data Type**	**Length**	**Nullable**	**Description**
1	id	INT	10	N	Contains Admin id
2	email	VARCHAR	100	N	Username for admin login
3	pass	VARCHAR	50	N	Password for admin login

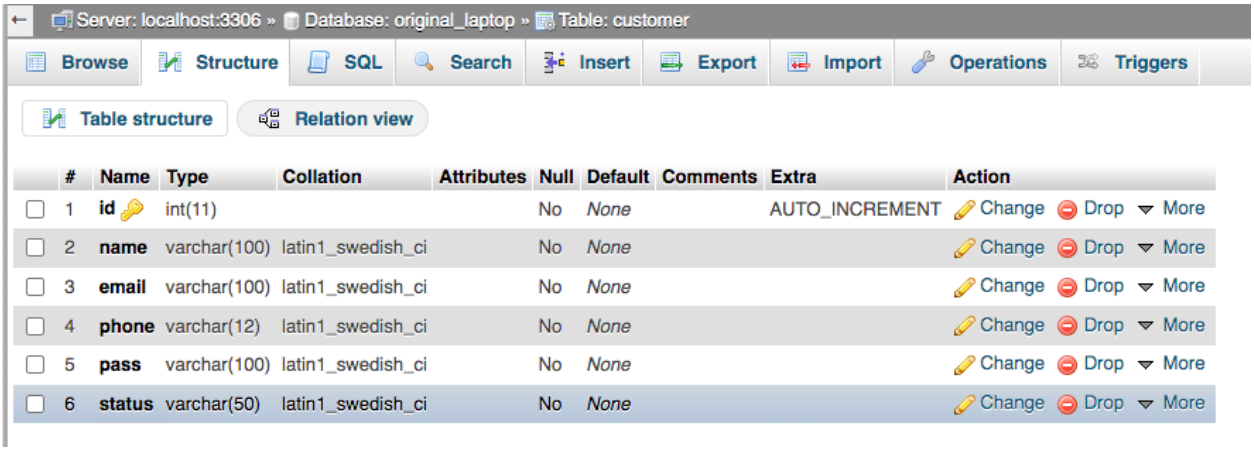


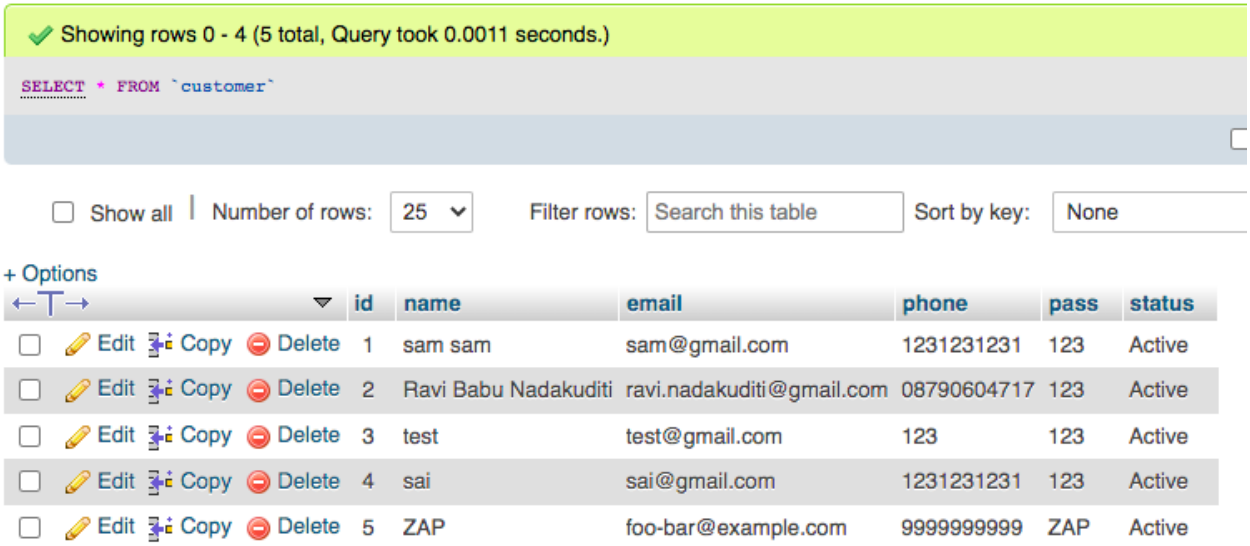


-- Table structure for table `customer` --

```
CREATE TABLE `customer`
( `id` int(11) NOT NULL,
`name` varchar(100) NOT NULL,
`email` varchar(100) NOT NULL,
`phone` varchar(12) NOT NULL,
`pass` varchar(100) NOT NULL,
`status` varchar(50) NOT NULL
);
```

Customer Registration					
	Column Name	**Data Type**	**Length**	**Nullable**	**Description**
1	id	INT	10	N	Contains renter/user id
2	name	VARCHAR	50	N	name for user login
3	email	VARCHAR	50	N	email for user login
4	phone	VARCHAR	50	N	Phone number
5	pass	VARCHAR	50	N	Password for user
6	status	VARCHAR	50	N	User account status(active/blocked)

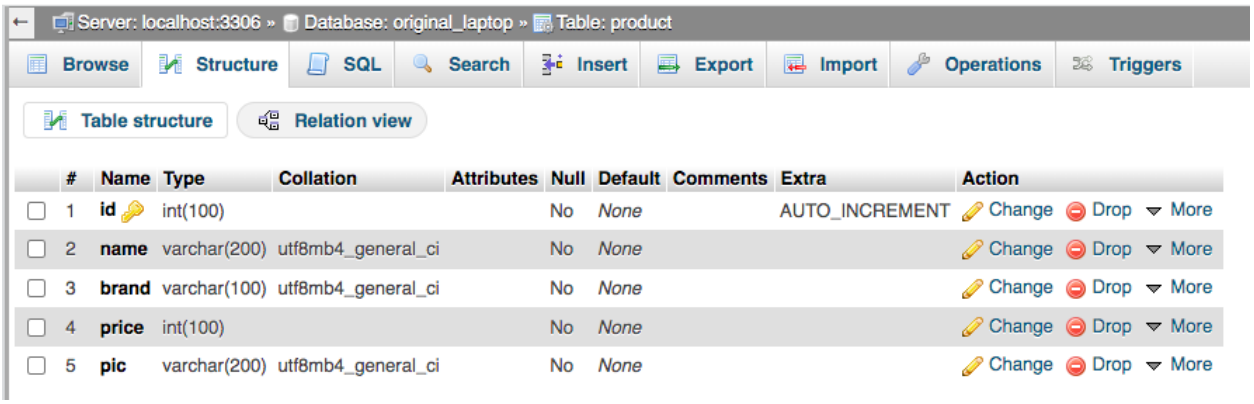


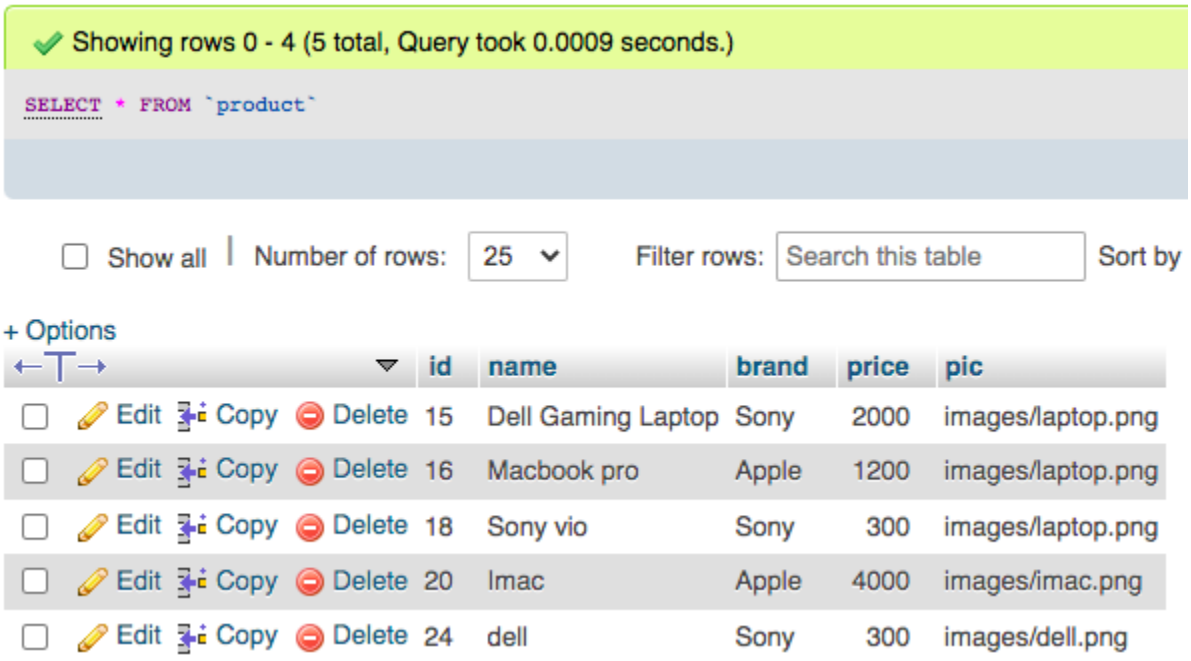


-- Table structure for table `product` --

```
CREATE TABLE `product`
(
`id` int(100) NOT NULL,
`name` varchar(200) NOT NULL,
`brand` varchar(100) NOT NULL,
`price` int(100) NOT NULL,
`pic` varchar(200) NOT NULL
);
```

product					
	Column Name	**Data Type**	**Length**	**Nullable**	**Description**
1	id	INT	100	N	Contains product id
2	name	VARCHAR	200	N	product name
3	brand	VARCHAR	100	N	Brand name of product
4	price	INT	100	N	Product price
5	pic	VARCHAR	200	N	Product image

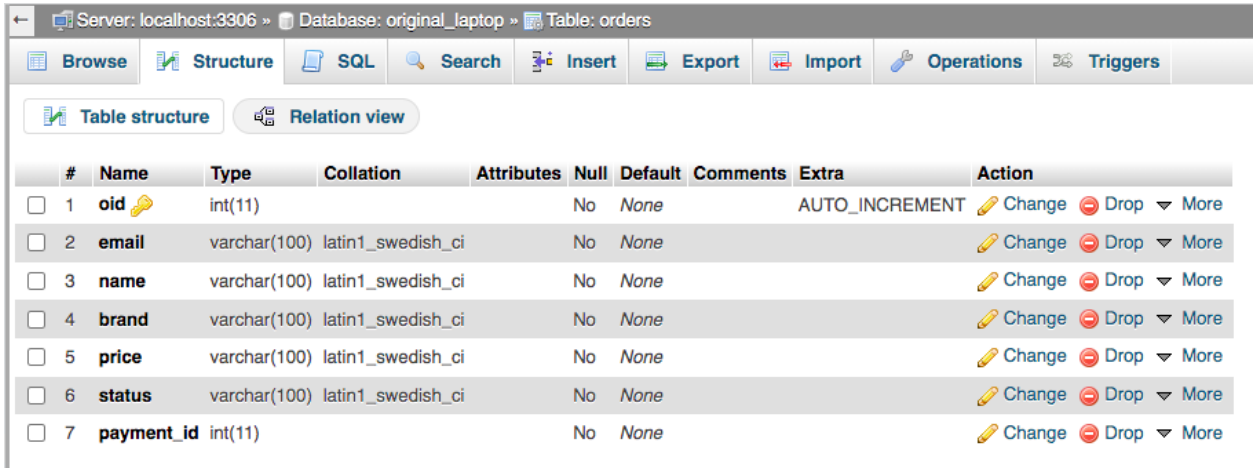


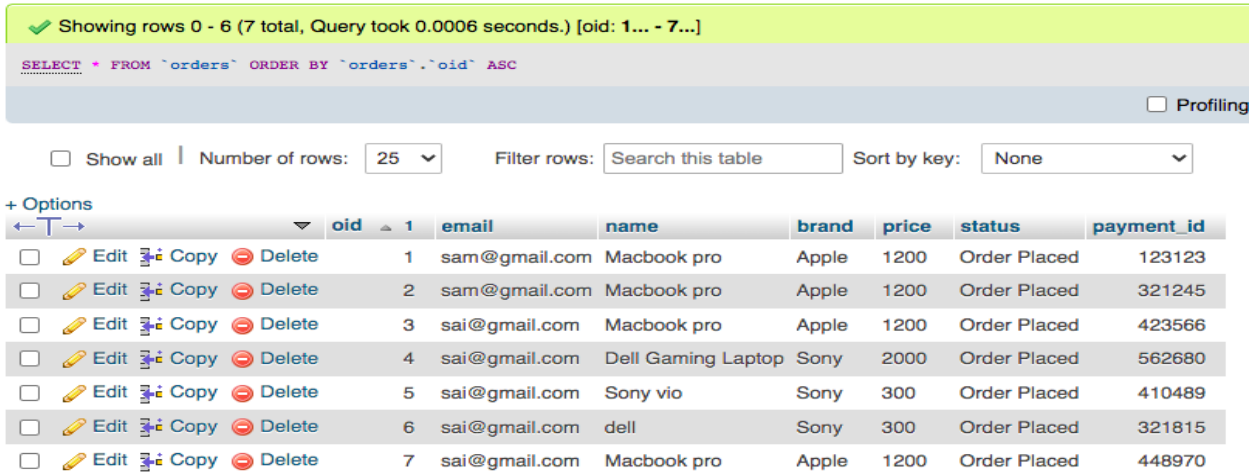


-- Table structure for table `orders` --

```
CREATE TABLE `orders`
(
`oid` int(11) NOT NULL,
`email` varchar(100) NOT NULL,
`name` varchar(100) NOT NULL,
`brand` varchar(100) NOT NULL,
`price` varchar(100) NOT NULL,
`status` varchar(100) NOT NULL,
`payment_id` int(11) NOT NULL
);
```

orders					
	Column Name	**Data Type**	**Length**	**Nullable**	**Description**
1	oid	INT	11	N	Contains order id
2	email	VARCHAR	100	N	Email of customer
3	name	VARCHAR	100	N	customer name
4	brand	VARCHAR	100	N	Brand name of product
5	price	VARCHAR	100	N	Product price
6	status	VARCHAR	100	N	Order status
7	payment_id	INT	11	N	Payment transaction id

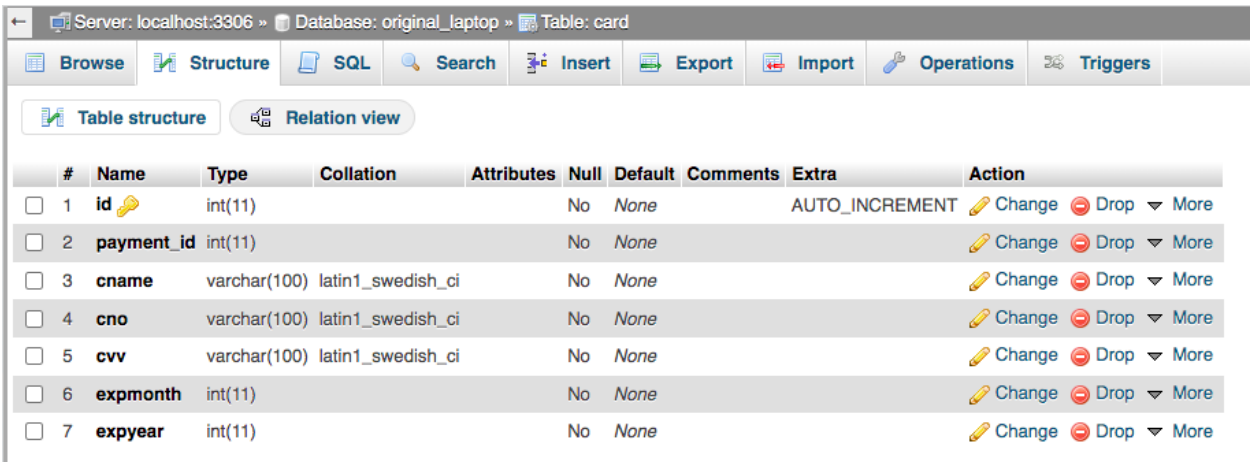


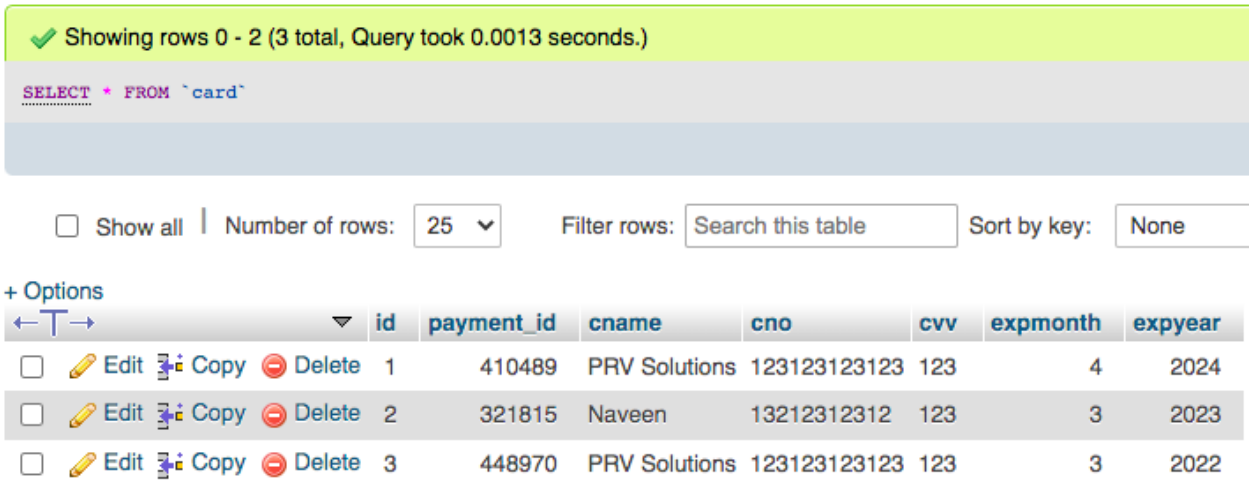


-- Table structure for table `card` --

```
CREATE TABLE `card`
(
`id` int(11) NOT NULL,
`payment_id` int(11) NOT NULL,
`cname` varchar(100) NOT NULL,
`cno` varchar(100) NOT NULL,
`cvv` varchar(100) NOT NULL,
`expmonth` int(11) NOT NULL,
`expyear` int(11) NOT NULL
);
```

card					
	Column Name	**Data Type**	**Length**	**Nullable**	**Description**
1	id	INT	11	N	Contains payment refernce id
2	payment_id	VARCHAR	100	N	Payment transaction id
3	cname	VARCHAR	100	N	customer name on card
4	cno	VARCHAR	100	N	Card number on card
5	cvv	VARCHAR	100	N	Security code
6	expmonth	INT	11	N	Card expiry month
7	expyear	INT	11	N	Card expiry year



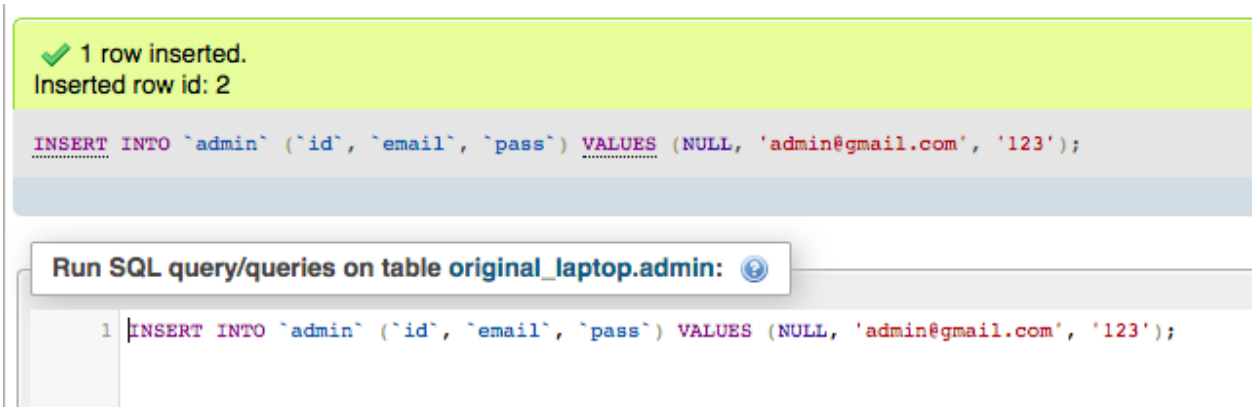


Data Manipulation Language Scripts

Insert Statements

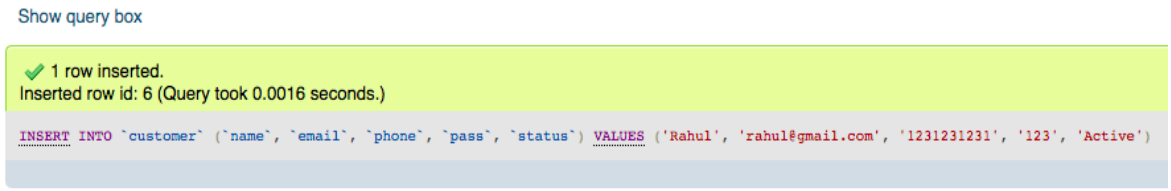
1. -- Dumping data for table `admin` --

   INSERT INTO `admin` (`id`, `email`, `pass`) VALUES (1, 'admin@gmail.com', '123');



2. -- Dumping data for table `customer` --

   INSERT INTO `customer` (`id`, `name`, `email`, `phone`, `pass`, `status`) VALUES (1, 'sam sam', 'sam@gmail.com', '1231231231', '123', 'Active'), (2, 'Ravi Babu Nadakuditi', 'ravi.nadakuditi@gmail.com', '08790604717', '123', 'Active'), (3, 'test', 'test@gmail.com', '123', '123', 'Active'), (4, 'sai', 'sai@gmail.com', '1231231231', '123', 'Active'), (5, 'ZAP', 'foo-bar@example.com', '9999999999', 'ZAP', 'Active');



3. -- Dumping data for table `orders` --

   INSERT INTO `orders` (`oid`, `email`, `name`, `brand`, `price`, `status`, `payment_id`) VALUES (1, 'sam@gmail.com', 'Macbook pro', 'Apple', '1200', 'Order Placed', 123123), (2, 'sam@gmail.com', 'Macbook pro', 'Apple', '1200', 'Order Placed', 321245), (3, 'sai@gmail.com', 'Macbook pro', 'Apple', '1200', 'Order Placed', 423566), (4, 'sai@gmail.com', 'Dell Gaming Laptop', 'Sony', '2000', 'Order Placed', 562680), (5, 'sai@gmail.com', 'Sony vio', 'Sony', '300', 'Order Placed', 410489), (6, 'sai@gmail.com', 'dell', 'Sony', '300', 'Order Placed', 321815), (7, 'sai@gmail.com', 'Macbook pro', 'Apple', '1200', 'Order Placed', 448970);

Update Statements

1. Update customer profile

   UPDATE customer

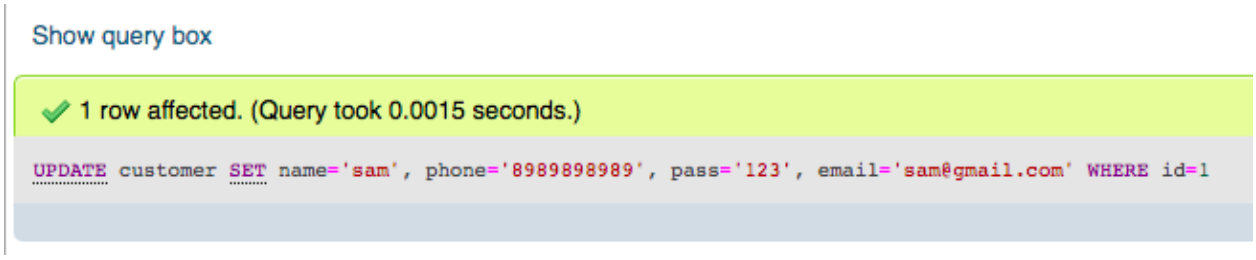
   SET

   name='sam', phone='8989898989,

   pass='123',

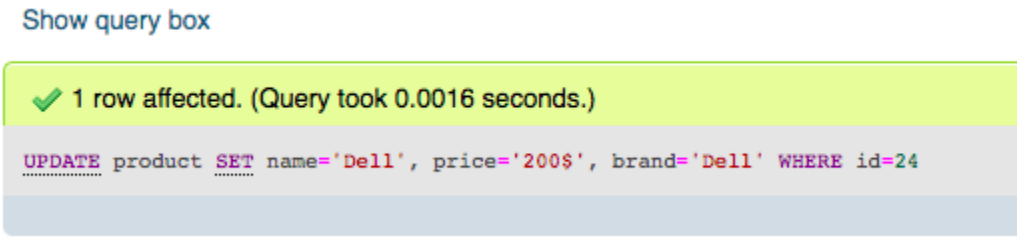
   email='sam@gmail.com

   WHERE id='3'";



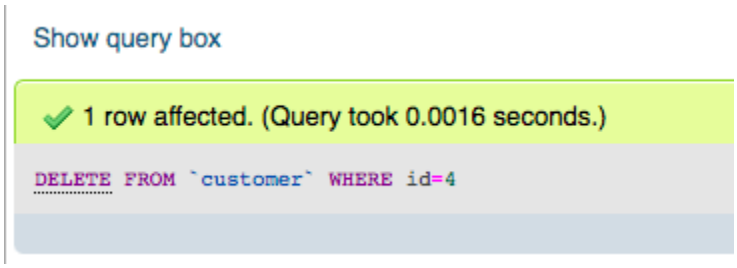
2. Update product information
   UPDATE product
   SET

name='Dell',
price='200$',
brand='Dell'
WHERE id='24'



Delete Statement

Delete from customer where id=4;



Select Statement

Select * from customer

Showing rows 0 - 5 (6 total, Query took 0.0012 seconds.)

```
SELECT * FROM `customer`
```

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

+ Options

	id	name	email	phone	pass	status
Edit Copy Delete	1	sam	sam@gmail.com	8989898989	123	Active
Edit Copy Delete	2	Ravi Babu Nadakuditi	ravi.nadakuditi@gmail.com	08790604717	123	Active
Edit Copy Delete	3	test	test@gmail.com	123	123	Active
Edit Copy Delete	4	sai	sai@gmail.com	1231231231	123	Active
Edit Copy Delete	5	ZAP	foo-bar@example.com	9999999999	ZAP	Active
Edit Copy Delete	6	Rahul	rahul@gmail.com	1231231231	123	Active

Join Statements

1. SELECT * FROM `orders` join card on orders.payment_id=card.payment_id

Showing rows 0 - 2 (3 total, Query took 0.0011 seconds.)

```
SELECT * FROM `orders` join card on orders.payment_id=card.payment_id
```

Profiling [Edit inline] [Edit] [Explain SQL] [C

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

+ Options

oid	email	name	brand	price	status	payment_id	id	payment_id	cname	cno	cvv	expmonth	expyear
5	sai@gmail.com	Sony vio	Sony	300	Order Placed	410489	1	410489	PRV Solutions	123123123123	123	4	2024
6	sai@gmail.com	dell	Sony	300	Order Placed	321815	2	321815	Naveen	13212312312	123	3	2023
7	sai@gmail.com	Macbook pro	Apple	1200	Order Placed	448970	3	448970	PRV Solutions	123123123123	123	3	2022

2. SELECT * FROM `orders` join customer on orders.email=customer.email

Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete features are not available.

Showing rows 0 - 1 (2 total, Query took 0.0010 seconds.)

```
SELECT * FROM `orders` join customer on orders.email=customer.email
```

Profiling [Edit inline] [Edit]

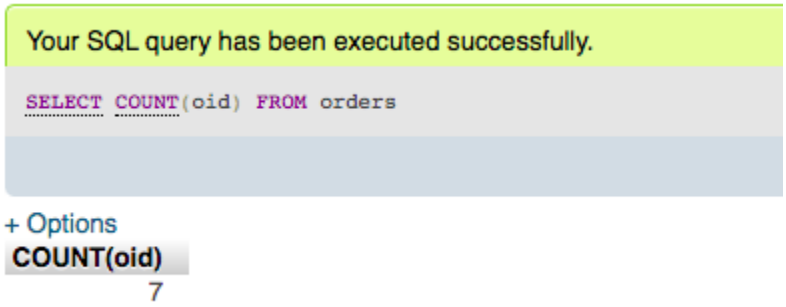
Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

+ Options

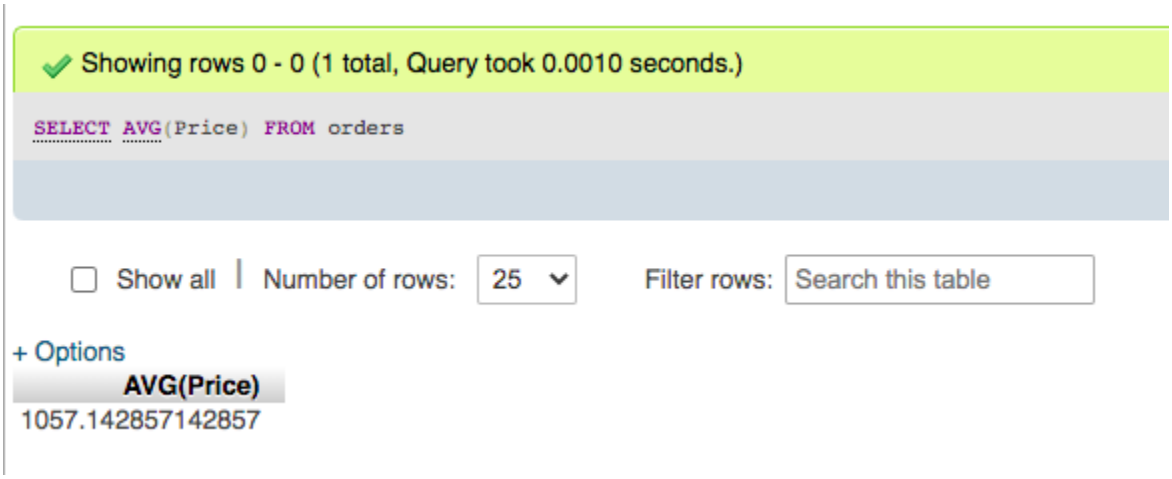
oid	email	name	brand	price	status	payment_id	id	name	email	phone	pass	status
1	sam@gmail.com	Macbook pro	Apple	1200	Order Placed	123123	1	sam	sam@gmail.com	8989898989	123	Active
2	sam@gmail.com	Macbook pro	Apple	1200	Order Placed	321245	1	sam	sam@gmail.com	8989898989	123	Active

Summary Statements

1. Total order count
   SELECT COUNT(oid) FROM orders



2. Average price of products
   SELECT AVG(Price) FROM orders;



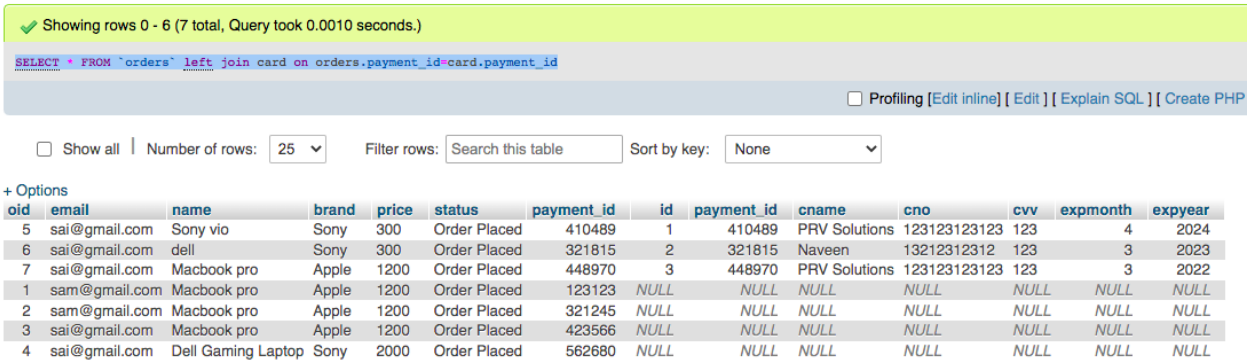
Multi table Query

SELECT * FROM `orders`

left join card

on

orders.payment_id=card.payment_id



Showing rows 0 - 6 (7 total, Query took 0.0010 seconds.)

```
SELECT * FROM `orders` left join card on orders.payment_id=card.payment_id
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP

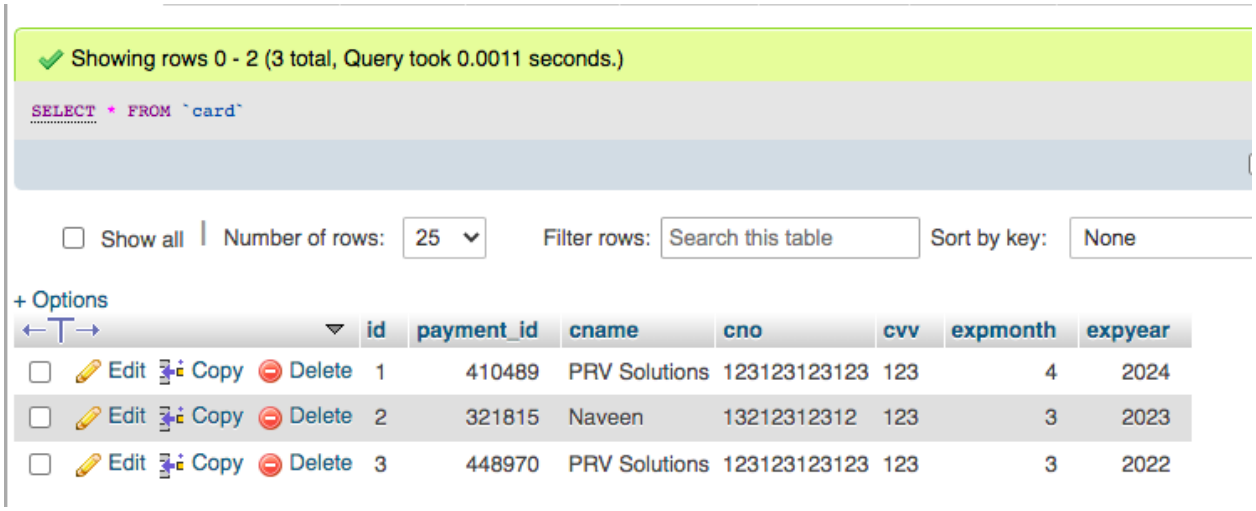
Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

+ Options

oid	email	name	brand	price	status	payment_id	id	payment_id	cname	cno	cvv	expmonth	expyear
5	sai@gmail.com	Sony vio	Sony	300	Order Placed	410489	1	410489	PRV Solutions	123123123123	123	4	2024
6	sai@gmail.com	dell	Sony	300	Order Placed	321815	2	321815	Naveen	13212312312	123	3	2023
7	sai@gmail.com	Macbook pro	Apple	1200	Order Placed	448970	3	448970	PRV Solutions	123123123123	123	3	2022
1	sam@gmail.com	Macbook pro	Apple	1200	Order Placed	123123	NULL	NULL	NULL	NULL	NULL	NULL	NULL
2	sam@gmail.com	Macbook pro	Apple	1200	Order Placed	321245	NULL	NULL	NULL	NULL	NULL	NULL	NULL
3	sai@gmail.com	Macbook pro	Apple	1200	Order Placed	423566	NULL	NULL	NULL	NULL	NULL	NULL	NULL
4	sai@gmail.com	Dell Gaming Laptop	Sony	2000	Order Placed	562680	NULL	NULL	NULL	NULL	NULL	NULL	NULL

My Choice Query

Select * from card;



Showing rows 0 - 2 (3 total, Query took 0.0011 seconds.)

```
SELECT * FROM `card`
```

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

+ Options

	id	payment_id	cname	cno	cvv	expmonth	expyear
Edit Copy Delete	1	410489	PRV Solutions	123123123123	123	4	2024
Edit Copy Delete	2	321815	Naveen	13212312312	123	3	2023
Edit Copy Delete	3	448970	PRV Solutions	123123123123	123	3	2022

Creating Indexes

-- Indexes for table `admin` --
ALTER TABLE `admin` ADD PRIMARY KEY (`id`);

-- Indexes for table `card` --
ALTER TABLE `card` ADD PRIMARY KEY (`id`);

-- Indexes for table `customer` --
ALTER TABLE `customer` ADD PRIMARY KEY (`id`);

-- Indexes for table `orders` --
ALTER TABLE `orders` ADD PRIMARY KEY (`oid`);

-- Indexes for table `product` --
ALTER TABLE `product` ADD PRIMARY KEY (`id`);

Unique index

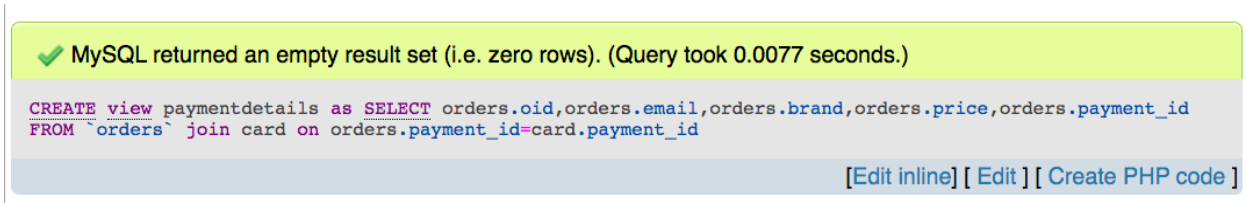
Creates a unique index on a customer table. Duplicate values are not allowed for email

-- Indexes for table `customer` --

ALTER TABLE `customer`
ADD
PRIMARY KEY (`id`), ADD UNIQUE KEY `email` (`email`);

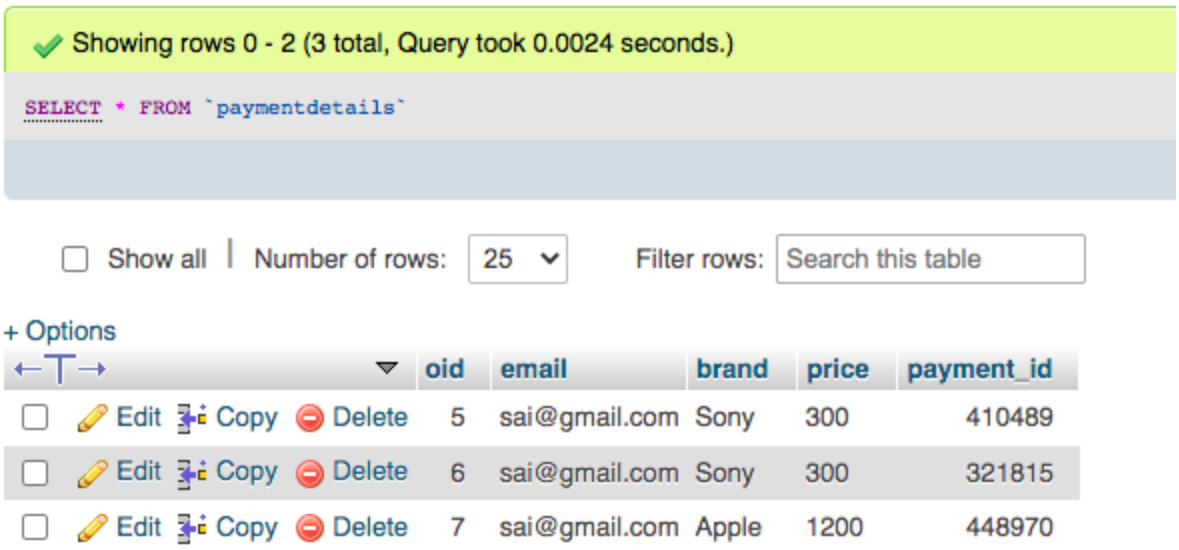
Creating Views

1. CREATE view paymentdetails as SELECT orders.oid,orders.email,orders.brand,orders.price,orders.payment_id FROM `orders` join card on orders.payment_id=card.payment_id

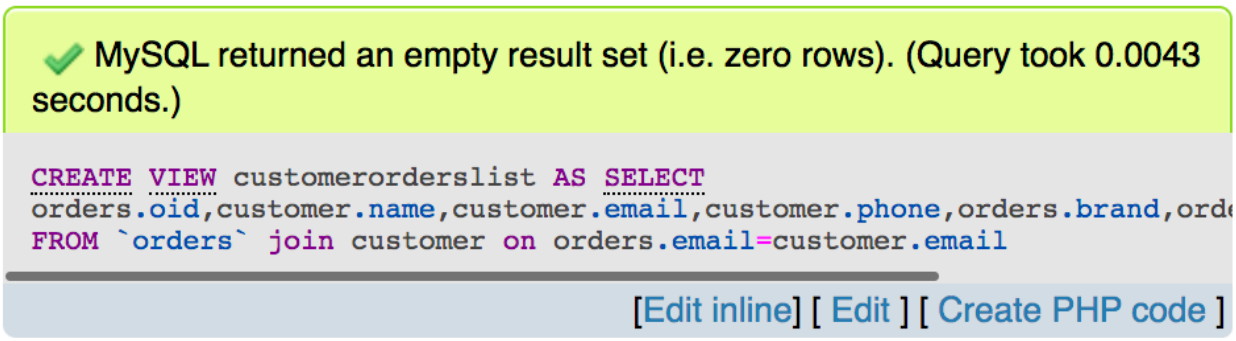


Select statement for view, our view name is paymentdetails. by using this view we can see total payment details at any time.

Select * from paymentdetails;

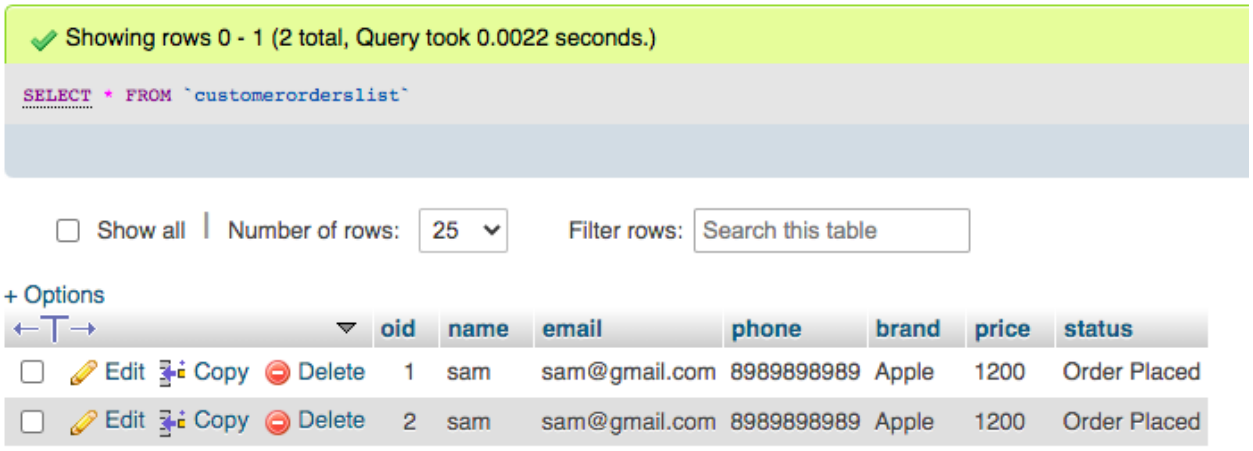


2. CREATE VIEW customerorderslist AS SELECT orders.oid,customer.name,customer.email,customer.phone,orders.brand,orders.price,orders.status FROM `orders` join customer on orders.email=customer.email



Select statement for view, our view name is customerorderslist by using this view we can see total customer orders by product wise at any time.

Select * from customerorderslist;



Triggers

Creating Trigger

DELIMITER $$

Create Trigger after_payment_details

AFTER INSERT ON `card`

FOR EACH ROW

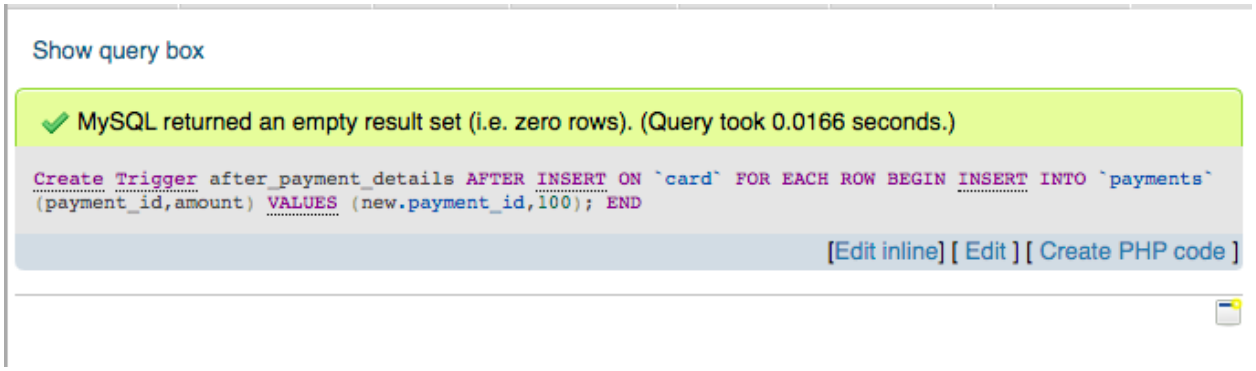
BEGIN

INSERT INTO `payments` (payment_id,amount) VALUES (new.payment_id,100);

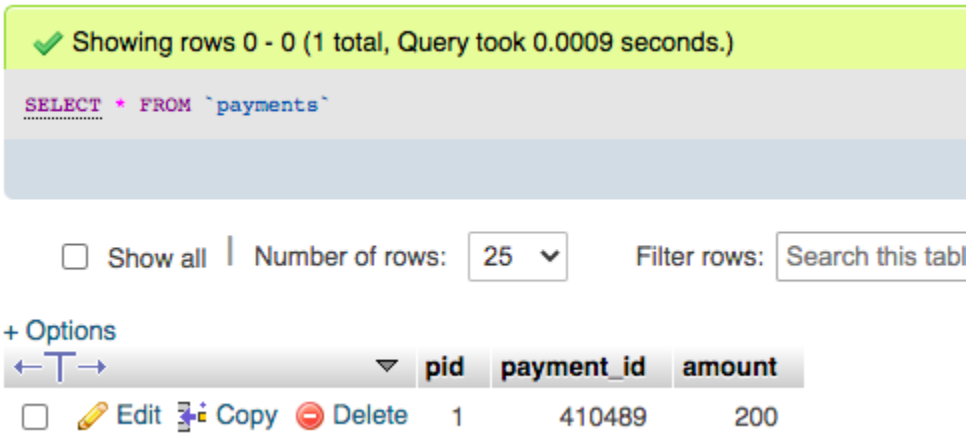
END$$

DELIMITER ;

Trigger created successfully with the name **after_payment_details**

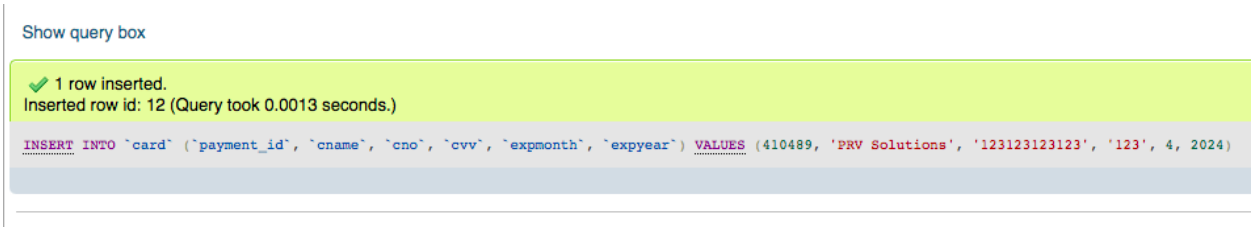


Select statement for payment

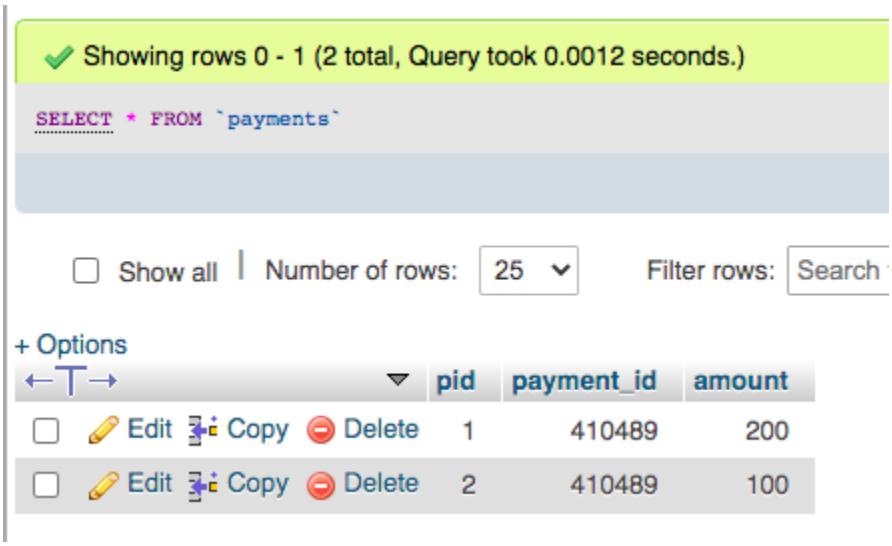


After inserting 1 row in card table effects the payment table by calling trigger

INSERT INTO `card` (`payment_id`, `cname`, `cno`, `cvv`, `expmonth`, `expyear`) VALUES (410489, 'PRV Solutions', '123123123123', '123', 4, 2024);



Data inserted in payments table by calling trigger automatically

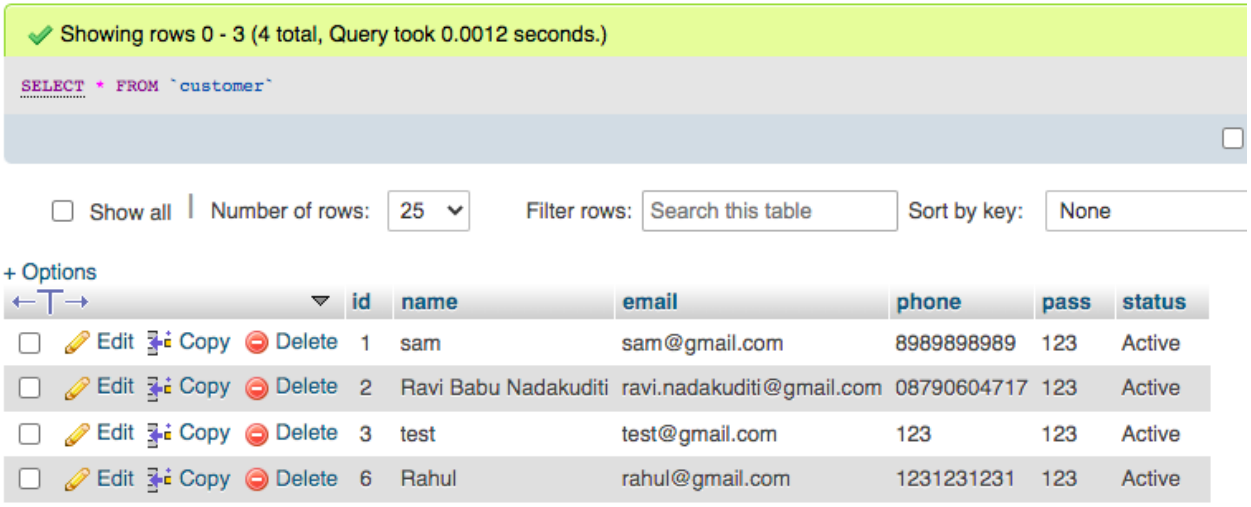


Transactions

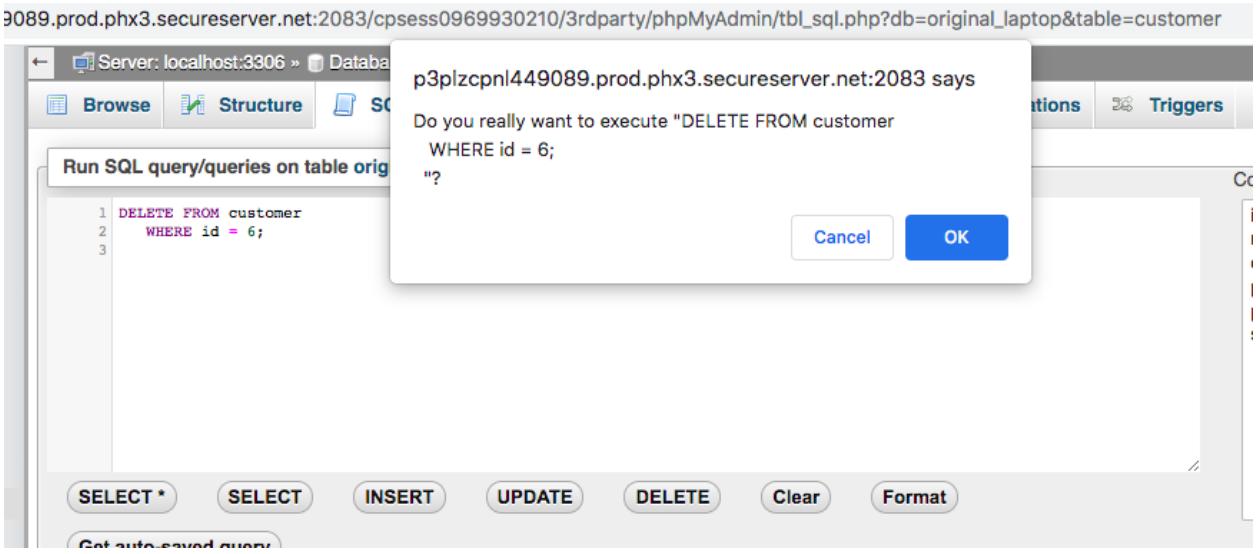
The COMMIT command is the transactional command used to save changes invoked by a transaction to the database.

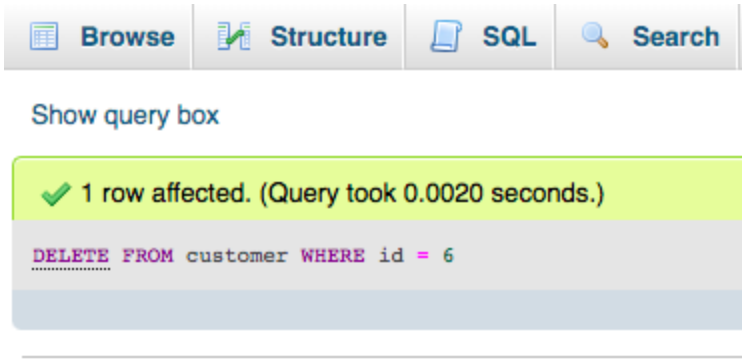
Example

Consider the customer table having the following records



Following is an example which would delete those records from the table which have id = 6 and then COMMIT the changes in the database.





Following is an example, which would delete those records from the table which have the id =6 and then ROLLBACK the changes in the database.

```
ROLLBACK;
```

Locking

Lock using the LOCK TABLE statement.

LOCK TABLE card Read;

LOCK TABLE card Write;

INSERT INTO `card`

(`id`, `payment_id`, `cname`, `cno`, `cvv`, `expmonth`, `expyear`)

VALUES

(6, 410489, 'PRV Solutions', '123123123123', '123', 4, 2024);

MySQL throws the following error:

Error Code: 1099. Table card was locked with a READ lock and can't be updated.

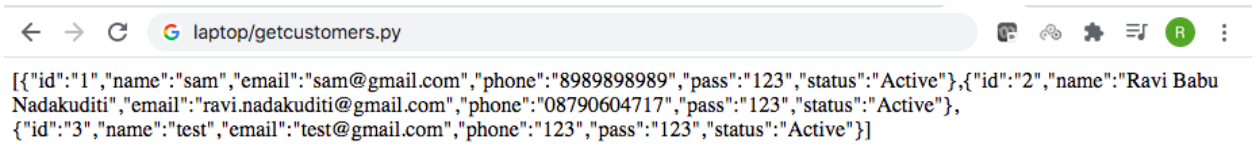
Python Programing

Fetching the customer table from database

```
#establishing the connection
conn = mysql.connector.connect(
  username='root', password='', hostname='127.0.0.1', database='original_laptop')
#Creating a cursor object using the cursor() method
cursor = conn.cursor()
#Retrieving single row
sql = '''SELECT * from customer'''
#Executing the query
cursor.execute(sql)
#Fetching all rows from the table
data = cursor.fetchmany();
print(data)

#Closing the connection
conn.close()
```

Output

[{"id":"1","name":"sam","email":"sam@gmail.com","phone":"8989898989","pass":"123","status":"Active"},{"id":"2","name":"Ravi Babu Nadakuditi","email":"ravi.nadakuditi@gmail.com","phone":"08790604717","pass":"123","status":"Active"},{"id":"3","name":"test","email":"test@gmail.com","phone":"123","pass":"123","status":"Active"}]



BONUS POINTS

PHP Programing (Search Functionality)

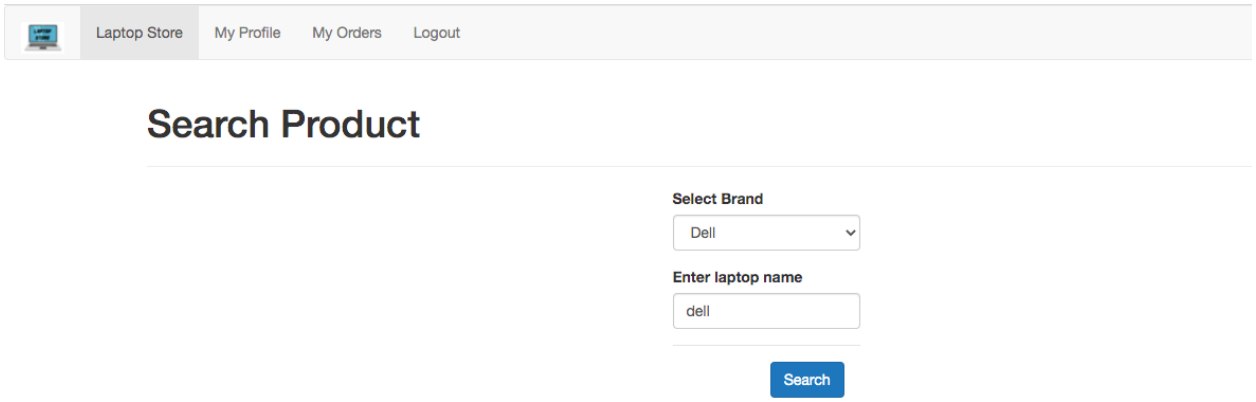
HTML Form

```
<form method="GET" action="searchproduct.php">
                <div class="form-group">
                   <label>Select Brand</label>
                  <select class="form-control" name="brand" id="brand">
                            <?php
                            include 'config.php';
                            $sql = "SELECT * FROM product";
                            $result = $conn->query($sql);
                            if ($result->num_rows > 0) {
                               // output data of each row
                               while ($row = $result->fetch_assoc()) {
                                  echo '
                               <option >' . $row["brand"] . '   </option>
                               ';
                               }
                            } else {
                               echo "No Brand found";
                            }
                            $conn->close();
                                    ?>
                  </select>
                </div>
                <div class="form-group">
                   <label>Enter laptop name</label>
                   <input type="text" class="form-control" id="name" name="name"  />
                </div>

                <div class="modal-footer">
                          <button class="btn btn-primary" type="submit">Search</button>
                        </div>
              </form>
```

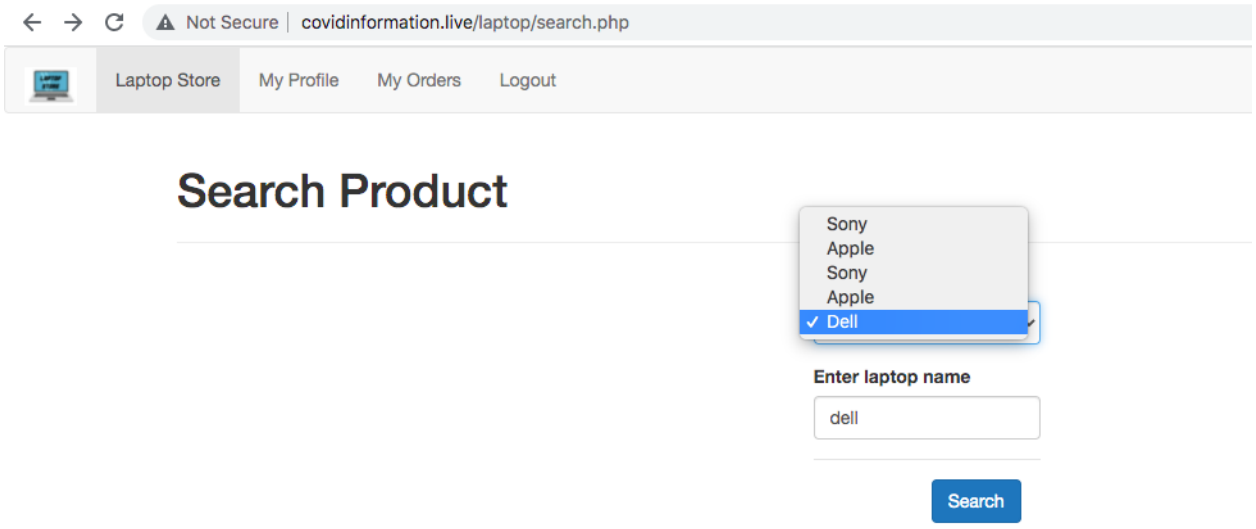
Php Code

```
<?php
            include 'config.php';
            $name=$_GET['name'];
            $brand=$_GET['brand'];
            $sql = "SELECT * FROM product where name='$name' or brand='$brand'";
            $result = $conn->query($sql);
            if ($result->num_rows > 0) {
               // output data of each row
               while ($row = $result->fetch_assoc()) {
                  echo '
                        <div class="col-md-4" style="text-align:center:margin-bottom:20px">

                           <img src=' . $row["pic"] . '  style="width:200px"/>
                           <h2>' . $row["name"] . '   </h2>

                           <td>$' . $row["price"] . ' CAD   </td>
                        <br/>

                           <a href="details.php?id=' . $row["id"] . '&name=' . $row["name"] .
'&price=' . $row["price"] . '&img=' . $row["pic"] . '&brand=' . $row["brand"] . '">
                           <button class="btn btn-primary">Buy</button></a>
                        <hr/>

                        </div>
            ';
               }
            } else {
               echo "No Results results";
            }
            $conn->close();
            ?>
```

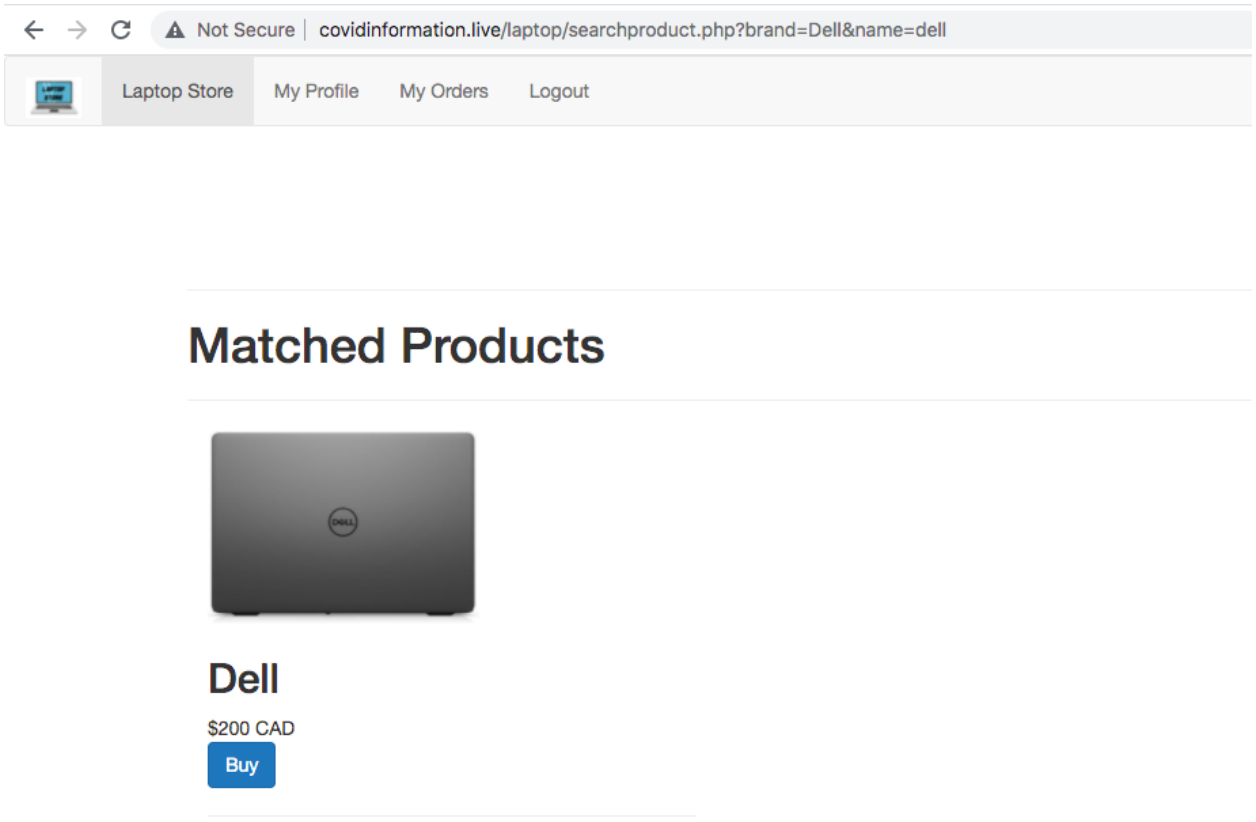
Search Product Html Form



Select Brand and enter search name

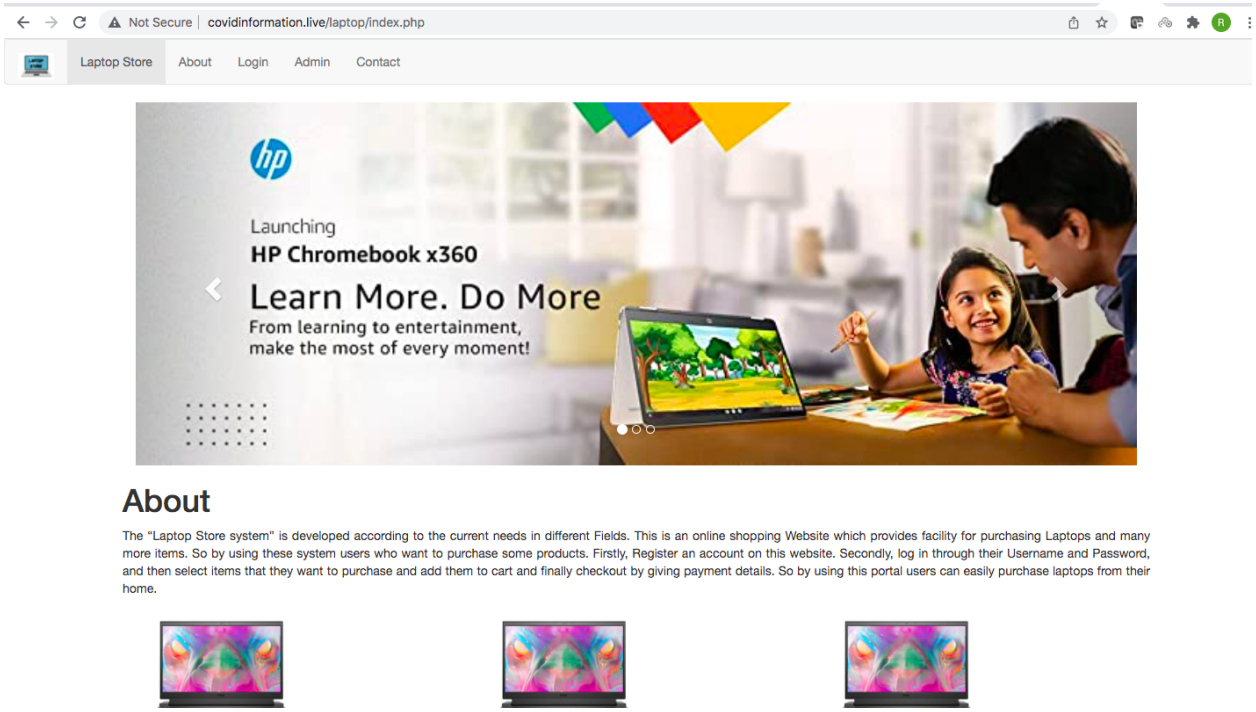


Result Page

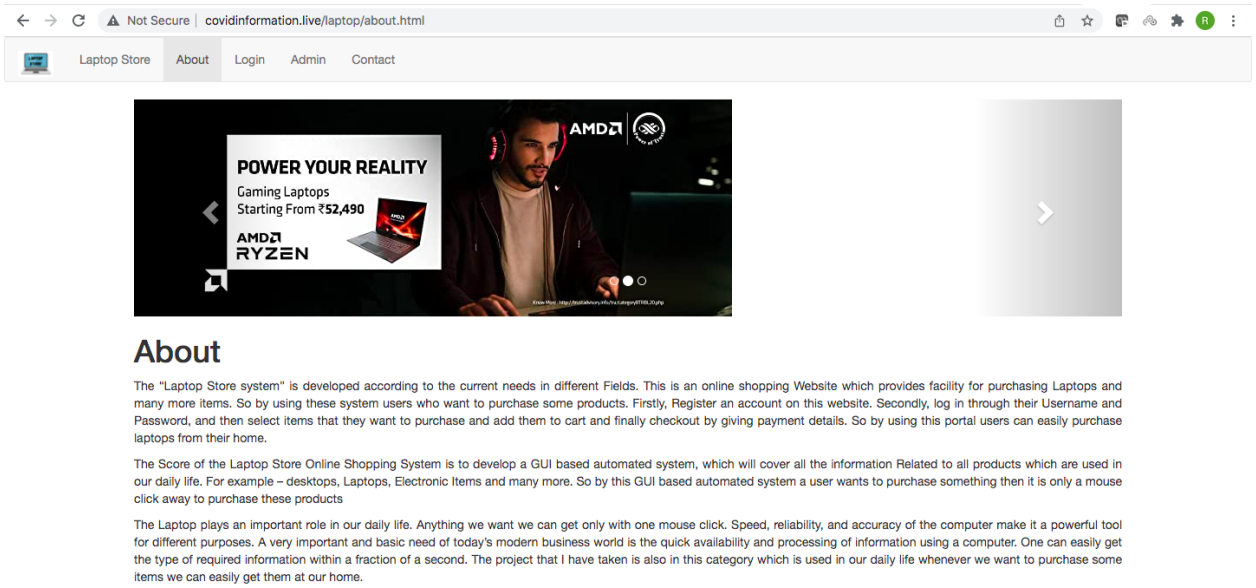

Not Secure | covidinformation.live/laptop/searchproduct.php?brand=Dell&name=dell
Laptop Store
My Profile
My Orders
Logout
Matched Products
Dell
$200 CAD
Buy

Project Enhancement Output Screens (Complete Project)

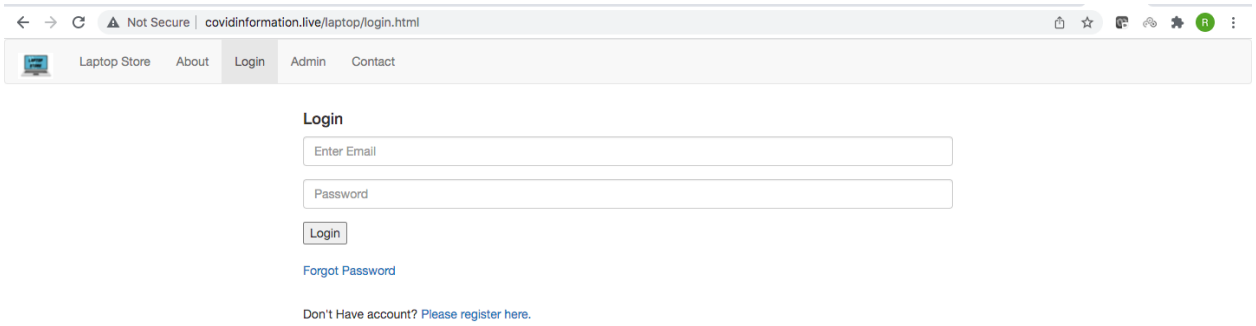
Laptops Application Home Screen



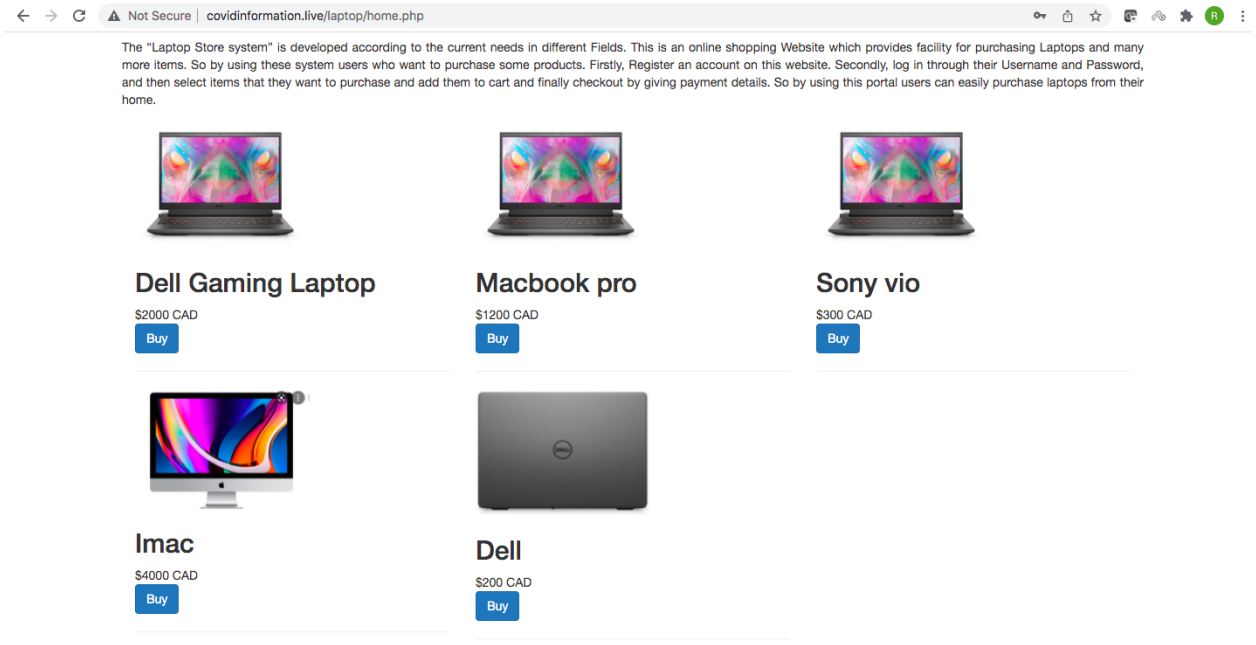
About Us Page



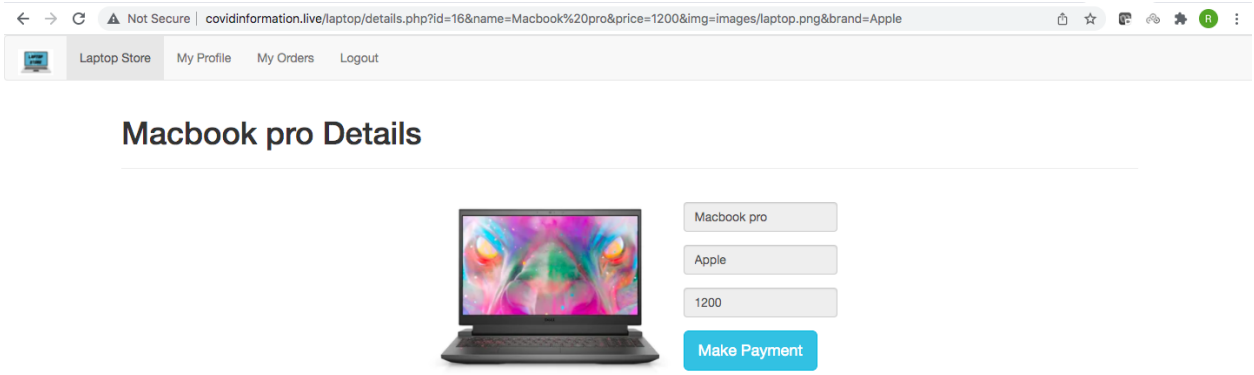
User Login Page



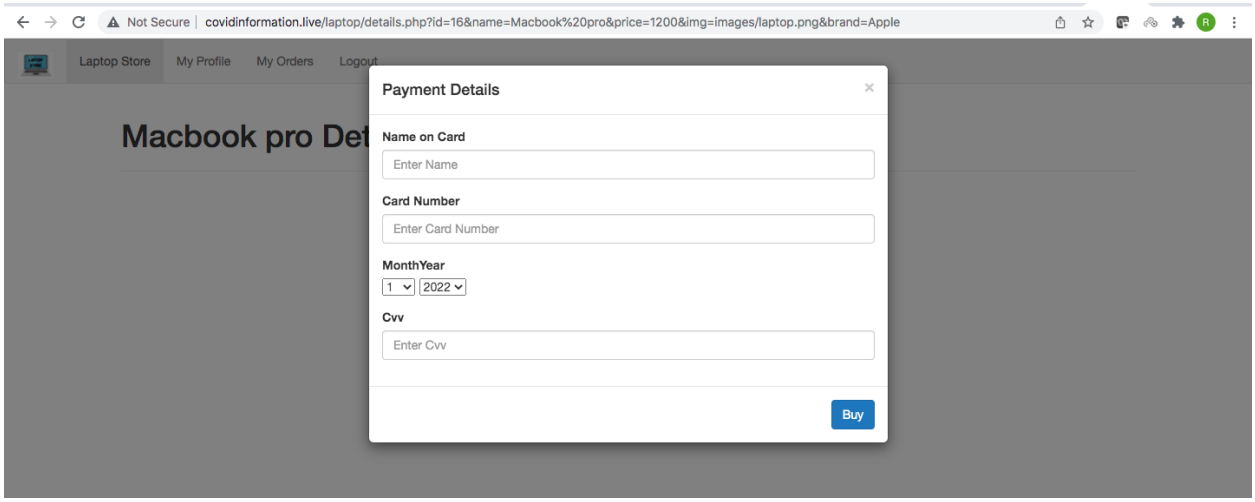
User Home Page with products



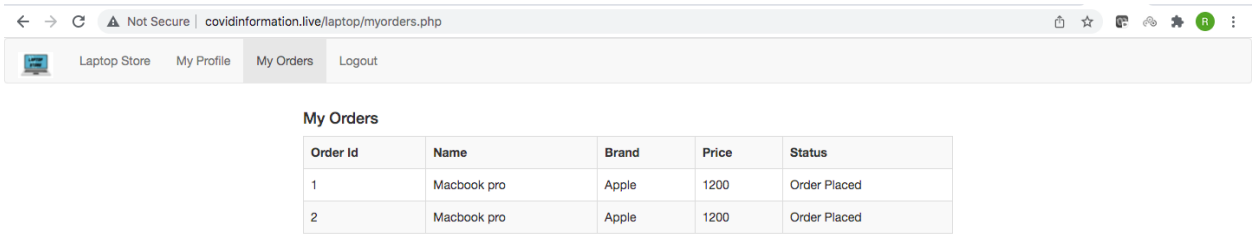
Selected product Details page



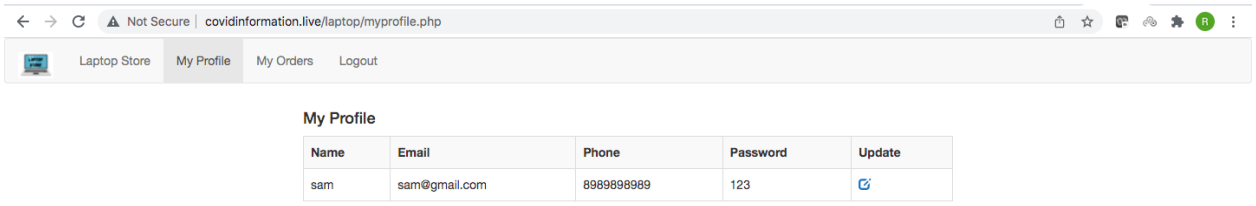
Payment details page



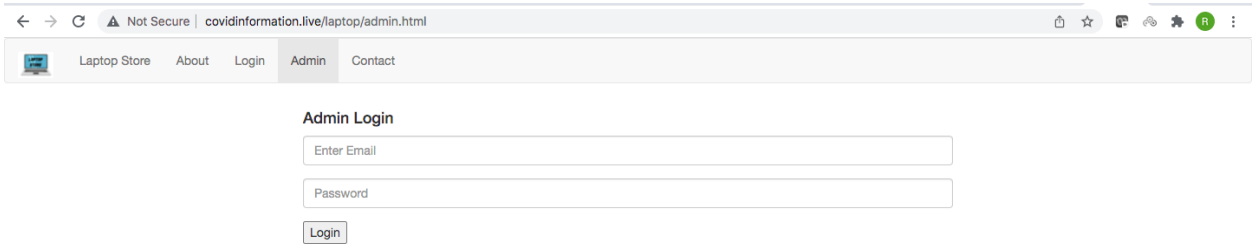
Customer can check his orders and status



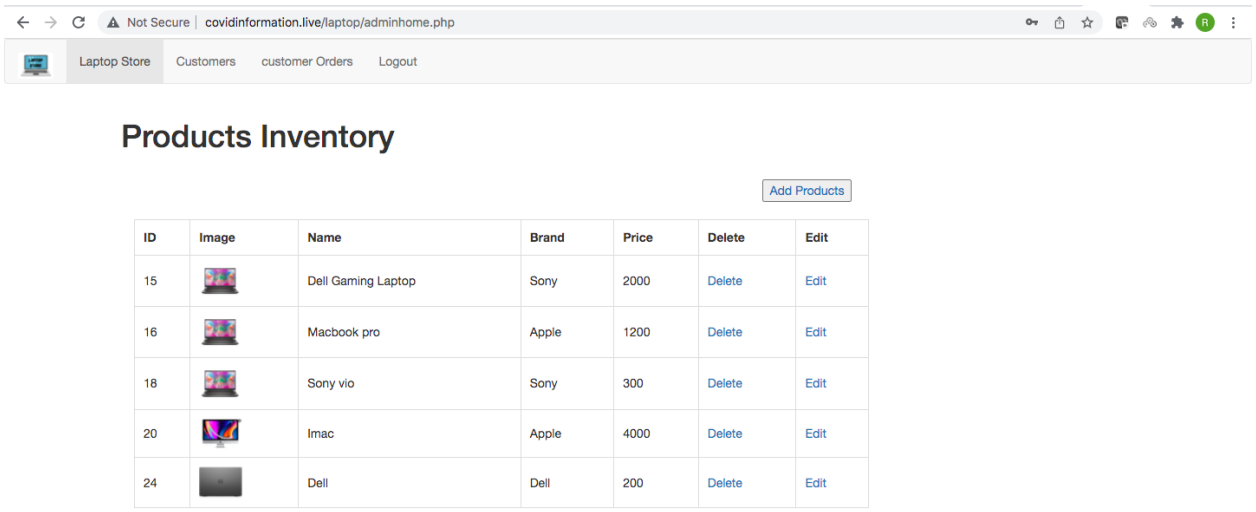
Customer can check his profile and update



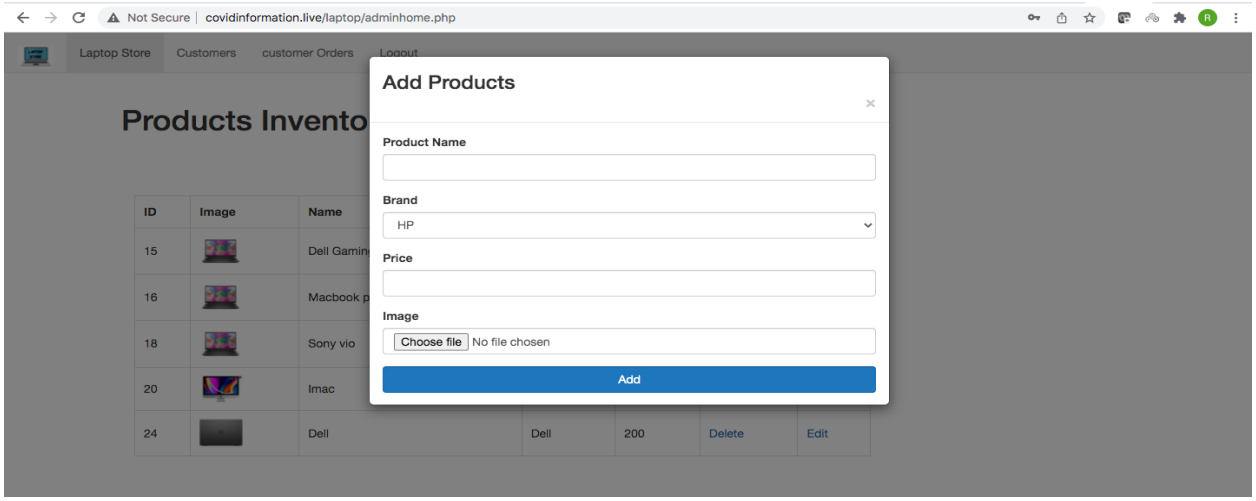
Admin login Form



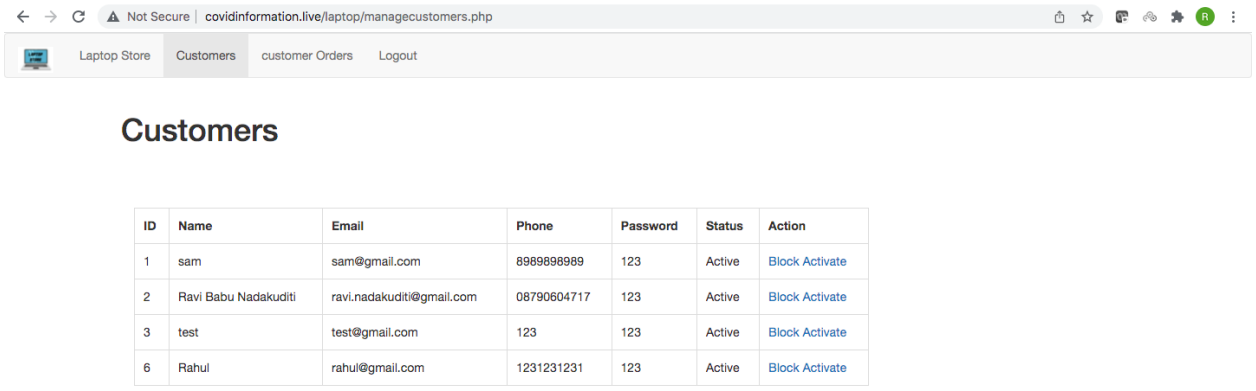
Admin home page with products inventory



Admin can manage products (Add/View/edit/Delete)

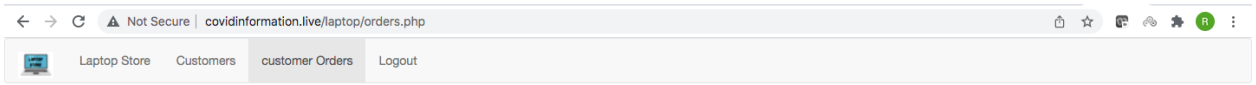


Admin can view all registered customer



ID	Name	Email	Phone	Password	Status	Action
1	sam	sam@gmail.com	8989898989	123	Active	Block Activate
2	Ravi Babu Nadakuditi	ravi.nadakuditi@gmail.com	08790604717	123	Active	Block Activate
3	test	test@gmail.com	123	123	Active	Block Activate
6	Rahul	rahul@gmail.com	1231231231	123	Active	Block Activate

Admin can view all customer orders



Customer Orders

Order Id	Name	Brand	Price	Status
1	Macbook pro	Apple	1200	Order Placed
2	Macbook pro	Apple	1200	Order Placed
3	Macbook pro	Apple	1200	Order Placed
4	Dell Gaming Laptop	Sony	2000	Order Placed
5	Sony vio	Sony	300	Order Placed
6	dell	Sony	300	Order Placed
7	Macbook pro	Apple	1200	Order Placed