Online Retail Store

Scope:

In a world marked by technological advancements and the rapid growth of internet users worldwide, how we live and shop has undergone significant changes. The aftermath of the COVID-19 pandemic has further accelerated this shift, with more people turning to online shopping for convenience and safety.

In light of these developments, our project takes on the challenge of designing an innovative online retail store. Our goal is to create a platform that not only meets the needs of consumers in this digital age but also provides a seamless and enjoyable shopping experience. It offers different functionalities to clients and store managers.

Functional Requirements:

The online retail store provides the following functionalities to clients/users:

Users can create an account when they visit our online store for the first time. If they have already made an account before, they can log in using their username and password. Once they are in, they can tell us more about themselves. They can add their name, address, and other details so we can send orders to the right place. They can also save their preferred payment methods. This way, they do not have to type in their card details whenever they want to buy something.

Users can save multiple addresses if they send a gift to someone else or want their order delivered to a different place. It is like having different delivery options. They can save different ways to pay for the orders. They can choose what works best for them, whether a credit card, debit card, or other methods.

Users will find various products neatly organized into categories when exploring our online store. Each product has detailed information, pictures, and prices, making choosing what they like easy. Imagine a virtual shopping cart – they can add products that they want. When they are ready to pay, we offer different payment options for a smooth checkout experience. They will see their order summary before confirming, just like reviewing the shopping list before heading to the cashier.

We have unique subscription plans, like being a member of a club. Depending on the user's plan (premium or elite), they get perks like quick and free delivery, discounts, and exclusive offers. We keep track of all past orders and purchases. Users can quickly check their order history and see where their current order is during delivery.

Users can cancel an order within a specific timeframe. If they are unhappy with a product, we have a process for returning it and getting a refund. Users can share their thoughts on products and services. We have a customer support system to answer their questions and address concerns.

Our online store works smoothly on any computer, phone, or tablet device. Using our search bar, users can filter products based on different criteria like price or popularity to find exactly what they want. We take users' online safety seriously. Our website has secure logins and payment methods to protect their information. We make sure to keep track of how many products we have. If something is running low, we will let the user know. Our online store supports multiple languages and regional preferences. We use tools to understand what users like and analyze popular products. That helps us make better decisions and improve our offerings. We always work to improve things based on user feedback and the latest trends. Expect regular updates to enhance user shopping experience.

Additional features offered for the store manager:

With the feature of product tracking, managers can effortlessly monitor and keep track of the availability of products within the store. This ensures efficient inventory management and fulfillment of customer orders. Managers can easily generate profit and loss statements, offering a clear and organized overview of the store's financial performance.

Stakeholders:

1. Clients/Users- The users can purchase products.
2. Store Operators- They manage store accounts and inventory.

Frontend:

1. HTML
2. CSS
3. Javascript

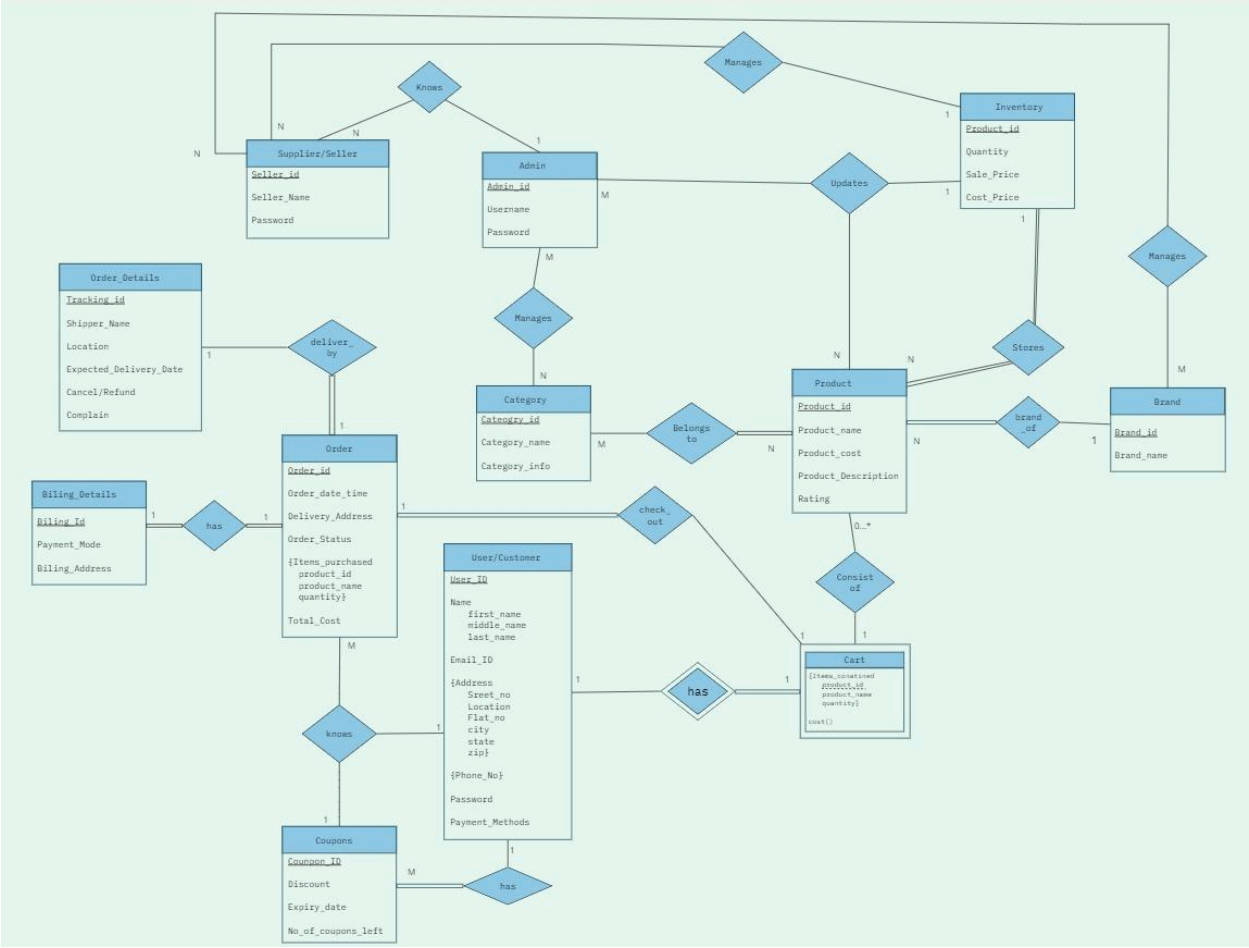
4. Bootstrap

Backend:

1. MySQL
2. Python
3. Django

S.No.	Name	Roll No.	Tut Group
1.	Monika Singh	2021169	2

Entity Relationship Model



Relational Model

Entities, Attributes & Schemas

1. admin_table

Schema: Admin(Admin_id, Username, Password)

Admin_id	INT PRIMARY KEY NOT NULL AUTO_INCREMENT
Username	VARCHAR(50) NOT NULL
Password	VARCHAR(50) NOT NULL

2. Product

Schema: Product(Product_id, Product_name, Product_cost,Product_Description,Rating, Brand_id(FK))

Product_id	INT PRIMARY KEY NOT NULL AUTO_INCREMENT
Product_name	VARCHAR(50) NOT NULL
Product_cost	DECIMAL(10,2) NOT NULL CHECK product_cost>0
Product_Description	VARCHAR(150) NOT NULL
Rating	INT NOT NULL
Brand_id	VARCHAR(50) NOT NULL FOREIGN KEY REFERENCES Brand

3. order_table

Schema: Order(Order_id, Order_date_time, Delivery_Address, Order_Status, Traking_id, Unique_id, Billing_Id, Counpon_ID)

Order_id	INT PRIMARY KEY NOT NULL AUTO_INCREMENT
Order_date_time	DATE TIME NOT NULL
Delivery_Address	VARCHAR(50) NOT NULL
Order_Status	VARCHAR(50) NOT NULL
Traking_id	INT NOT NULL FOREIGN KEY REFERENCES Order_Details
Unique_id	INT NOT NULL FOREIGN KEY REFERENCES User
Billing_Id	INT NOT NULL FOREIGN KEY REFERENCES Biling_Details
Counpon_ID	Varchar(40) Default NULL Foreign Key references Coupons

4. Order_Details

Schema: Order_Details(Tracking_id, Shipper_Name, Location, Expected_Delivery_Date, Cancel/Refund, Complain)

Tracking_id	INT PRIMARY KEY NOT NULL AUTO_INCREMENT
Shipper_Name	VARCHAR(50) NOT NULL

Location	VARCHAR(50) NOT NULL
Expected_Delivery_Date	DATE NOT NULL
Cancel/Refund	VARCHAR(50) NOT NULL
Complain	VARCHAR(50) NOT NULL

5. Supplier/Seller

Schema: Seller(Seller_id, Seller_Name, Delivery_speed)

Seller_id	INT PRIMARY KEY NOT NULL AUTO_INCREMENT
Seller_Name	VARCHAR(50) NOT NULL
Password	INT NOT NULL
Delivery_speed	INT NOT NULL

6. User/Customer

Schema: User(User_ID, Name, Email_ID, Address, Phone_No, Password, Payment_Methods)

User_ID	INT PRIMARY KEY NOT NULL AUTO_INCREMENT
Name	VARCHAR(50) NOT NULL
Email_ID	VARCHAR(50) NOT NULL UNIQUE
Address	VARCHAR(50)

	NOT NULL
Phone_No	VARCHAR(50) NOT NULL
Password	VARCHAR(50) NOT NULL
Payment_Methods	VARCHAR(50) NOT NULL

7. coupon_data

Schema: Coupons(Counpon_ID, Discount, Expiry_date, No_of_coupons_left, Unique_id,isUsed)

Counpon_ID	VARCHAR(40) PRIMARY KEY NOT NULL AUTO_INCREMENT
Discount	INT NOT NULL CHECK DISCOUNT>0
Expiry_date	DATE NOT NULL
No_of_coupons_left	INT NOT NULL
Unique_id	INT NOT NULL FOREIGN KEY REFERENCES User
isUsed	INT Default value 0

8. billing_details

Schema: Biling_Details(Biling_Id, Payment_Mode, Biling_Address)

Biling_Id	INT PRIMARY KEY NOT NULL AUTO_INCREMENT

Payment_Mode	VARCHAR(30) NOT NULL
Biling_Address	VARCHAR(50) NOT NULL

9. brand

Schema: Brand(Brand_id, Brand_name)

Brand_id	INT PRIMARY KEY NOT NULL AUTO_INCREMENT
Brand_name	VARCHAR(50) PRIMARY KEY NOT NULL

10. inventory

Schema: Inventory(Product_id, Quantity, Sale_Price, Cost_Price)

Product_id	INT PRIMARY KEY NOT NULL FOREIGN KEY REFERENCES PRODUCT
Quantity	INT NOT NULL CHECK Quantity > 0
Sale_Price	INT NOT NULL
Cost_Price	INT NOT NULL

11.cart_data

Schema: Cart(Unique_ID)

Unique_id	INT PRIMARY KEY NOT NULL

	FOREIGN KEY REFERENCES User CHECK Quantity > 0

12.category

Schema: Category(Category_id, Category_name, Category_info)

Category_id	INT PRIMARY KEY NOT NULL AUTO_INCREMENT
Category_name	VARCHAR(50) NOT NULL
Category_info	VARCHAR(50)

Schemas for Relations & Multivalued Attributes

1. items_contained (in Cart)

Schema: items_contained(Unique_ID, product_id, product_name, quantity)

Unique_id	INT NOT NULL FOREIGN KEY REFERENCES cart_data
product_id	INT NOT NULL
product_name	VARCHAR(50) NOT NULL
quantity	INT NOT NULL CHECK Quantity>0

PRIMARY KEY (Unique_id, product_id)

2. items_purchased (in Order)

Schema: items_purchased(Order_id, product_id, product_name, quantity, Total_Cost)

Order_id	INT NOT NULL FOREIGN KEY REFERENCES Order
product_id	INT NOT NULL
product_name	VARCHAR(50) NOT NULL
quantity	INT NOT NULL Check Quantity > 0
Total_Cost	INT DEFAULT=0 Check Cost > 0

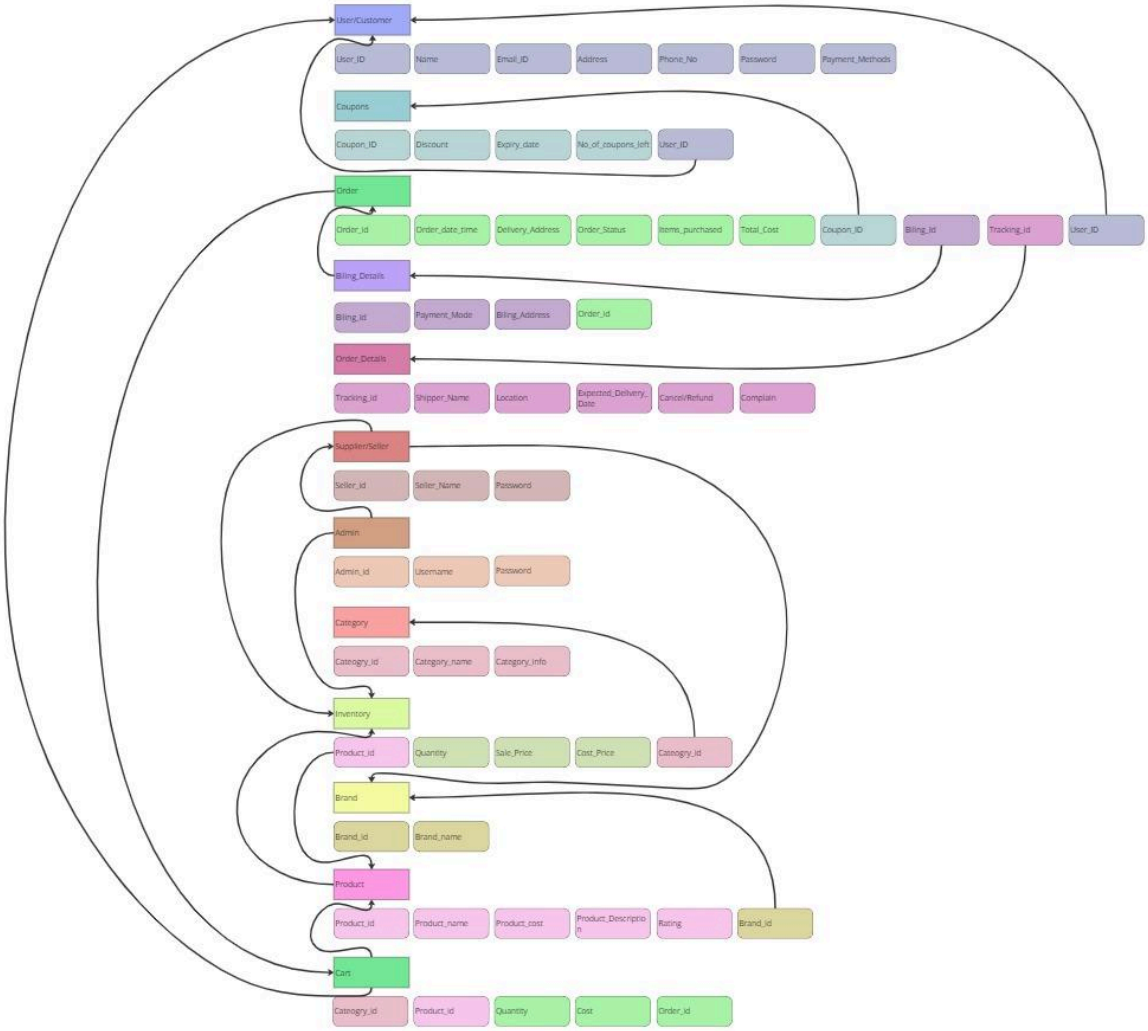
PRIMARY KEY (Order_id, product_id)

3. belongsTo (relation b/w Product & Category)

Schema: Belongsto(Product_id,Category_id)

Product_id	INT NOT NULL FOREIGN KEY REFERENCES Product
Category_id	INT NOT NULL FOREIGN KEY REFERENCES Category

PRIMARY KEY (Product_id,Category_id)



Weak entity:

Taking inspiration from Blinkit, we tried to imitate the functioning of our database model similar to it. Since on Blinkit, there is no functionality of a guest cart similar to that we have made our cart, which needs to be associated with a user, therefore, Cart is a weak entity in our database model.

Ternary Relation:

1. Between Inventory, Admin, and Product:
The admin is responsible for managing and taking updates from the inventory, and the products share a relationship with the inventory. Therefore, the three will be in ternary relationship where the Admin can modify Inventory and Product. Changes in Product quantity reflect in Inventory

as Inventory uses Product IDs, and Order placing or removal of Products also must reflect in the Inventory.

2. Between User, Orders, and Coupons:
Whenever an order is placed/initiated, the user may use a coupon to place/order it, hence, they will be in a ternary relationship. The coupon is also related to order because if it is being used, it has an isUsed attribute set to 1 indicating used.

Views & Grants Used

In my project, there are two main stakeholders: users and admins. We have specific user roles named "customer" and "administer," corresponding to regular users and administrators. The "user" entity stores emails and passwords in the database, while the "admin" table contains a username and passkey. Users are identified by their passwords and admins by their usernames.

For the "customer" user, we grant select access to view products, brands, shippers, categories, and products belonging to a category. However, the customer has limited access and can only view these elements without the ability to make changes.

The "customer" user has updated access to specific tables like "order_table," "inventory," "items_purchased," "billing_details," "items_contained," and "coupon_data." This access allows the customer to apply coupons, update the inventory when purchasing products, and modify quantities in the shopping cart.

Additionally, the "customer" has insert access for tables like "items_contained," "billing_details," "order_table," and "items_purchased." This allows users to add items to the cart, purchase, and insert relevant details. The "customer" also has to delete access to remove items from the cart and delete orders.

On the other hand, the "administrator" has broad privileges across most tables, except for "admin_table," where they can only view other admins but cannot add, delete, or update entries. The root user retains exclusive privileges for admin_table.

The "administrator" lacks access to "cart_details" and "items_contained" to ensure they do not have unnecessary information or control over user carts and their contents.

```
-- Create user_role for customers
CREATE ROLE IF NOT EXISTS user_role;
GRANT SELECT ON eCartBazzar.usableCouponView TO user_role;
GRANT SELECT ON eCartBazzar.userProductView TO user_role;
GRANT SELECT ON eCartBazzar.categoryUserView TO user_role;
GRANT SELECT, UPDATE ON eCartBazzar.inventory TO user_role;
GRANT SELECT, UPDATE, INSERT, DELETE ON eCartBazzar.items_purchased TO user_role;
GRANT SELECT, UPDATE, INSERT, DELETE ON eCartBazzar.order_table TO user_role;
GRANT SELECT ON eCartBazzar.seller TO user_role;
GRANT SELECT, INSERT, UPDATE, DELETE ON eCartBazzar.billing_details TO user_role;
GRANT SELECT ON eCartBazzar.belongsto TO user_role;
GRANT SELECT ON eCartBazzar.product TO user_role;
GRANT SELECT ON eCartBazzar.brand TO user_role;
GRANT SELECT ON eCartBazzar.category TO user_role;
GRANT SELECT, UPDATE, INSERT, DELETE ON eCartBazzar.items_contained TO user_role;
GRANT SELECT ON eCartBazzar.cart TO user_role;
GRANT SELECT, UPDATE ON eCartBazzar.coupons TO user_role;
GRANT SELECT ON eCartBazzar.user TO user_role;
GRANT SELECT ON eCartBazzar.protectedUserView TO user_role;


-- Create admin_role for admins
CREATE ROLE IF NOT EXISTS admin_role;

-- Grant permissions to admin_role
GRANT ALL ON eCartBazzar.user TO admin_role;
GRANT ALL ON eCartBazzar.coupons TO admin_role;
-- Not granting admin any data related to cart and order
GRANT ALL ON eCartBazzar.category TO admin_role;
GRANT ALL ON eCartBazzar.brand TO admin_role;
GRANT ALL ON eCartBazzar.product TO admin_role;
GRANT ALL ON eCartBazzar.belongsto TO admin_role;
GRANT ALL ON eCartBazzar.billing_details TO admin_role;
-- For the time being, an admin does not have to write alter on other admins
GRANT SELECT ON eCartBazzar.admin TO admin_role;
GRANT ALL ON eCartBazzar.seller TO admin_role;
GRANT SELECT, UPDATE, INSERT, DELETE, CREATE, DROP ON eCartBazzar.order_table TO admin_role;
GRANT SELECT, UPDATE, INSERT, DELETE, CREATE, DROP ON eCartBazzar.items_purchased TO admin_role;
GRANT ALL ON eCartBazzar.inventory TO admin_role;
GRANT ALL ON eCartBazzar.userProductView TO admin_role;
GRANT ALL ON eCartBazzar.categoryUserView TO admin_role;
GRANT ALL ON eCartBazzar.protectedUserView TO admin_role;
GRANT ALL ON eCartBazzar.usableCouponView TO admin_role;
```

We have four views :

userProductView: Displays product details for a user.
categoryUserView: Displays categories for a user based on their orders.
protectedUserView: Displays users with admin privileges.
usableCouponView: Displays usable coupons.

```sql
-- Create a view to display products for a specific user
CREATE VIEW userProductView  AS
SELECT
    Product_id,
    Product_name,
    Product_cost,
    Product_Description,
    Rating,
    Brand_id
FROM
    Product;

SELECT * FROM userProductView;
```

```sql
-- Create a view to display categories for a user
CREATE VIEW categoryUserView  AS
SELECT DISTINCT U.User_ID, C.Category_id, C.Category_name
FROM User U
JOIN Order_Table OT ON U.User_ID = OT.Unique_id
JOIN Items_purchased IP ON OT.Order_id = IP.Order_id
JOIN Product P ON IP.Product_id = P.Product_id
JOIN BelongsTo B ON P.Product_id = B.Product_id
JOIN Category C ON B.Category_id = C.Category_id;
SELECT * FROM categoryUserView;
-- Create a view to display users with admin privileges
CREATE VIEW protectedUserView  AS
SELECT
    u.User_ID,
    u.Name,
    u.Email_ID,
    u.Address,
    u.Phone_No,
    u.Payment_Methods
FROM
    User u
JOIN
    Admin a ON u.User_ID = a.Admin_id;
SELECT * FROM protectedUserView;
```

```sql
-- Create a view for Usable Coupons
CREATE VIEW usableCouponView  AS
SELECT *
FROM Coupons
WHERE isUsed = 1 AND Expiry_date >= CURDATE();
SELECT * FROM usableCouponView;
```

Indexing Attributes

Indexes are created for optimization purposes. For example:

A unique index on Category_name in the Category table.
Index on Delivery_speed in the Seller table.

```
-- Create unique index on category_name in Category table
CREATE UNIQUE INDEX categoryname ON Category(Category_name);

-- Create index on Delivery_speed in Seller table
CREATE INDEX shipperspeed ON Seller(Delivery_speed);

-- Create index on brand_name in Brand table
CREATE INDEX brandn ON Brand(Brand_name);

-- Create unique index on product_id in Product table
CREATE UNIQUE INDEX prod_id ON Product(Product_id);

-- Create unique index on (product_name, brand_name) in Product table
CREATE UNIQUE INDEX product_name_brand_name ON Product(Product_name, Brand_id);

-- Create index on billing_id in Billing_Details table
CREATE INDEX billing ON Billing_Details(Billing_Id);

-- Create unique index on (Email_ID, Password) in User table
CREATE UNIQUE INDEX username_password ON User(Email_ID, Password);

-- Create unique index on (Unique_id, Product_ID, Quantity) in Items_contained table
CREATE UNIQUE INDEX items_contained_index ON Items_contained(Unique_id, Product_id, Quantity);
```

Triggers Implemented

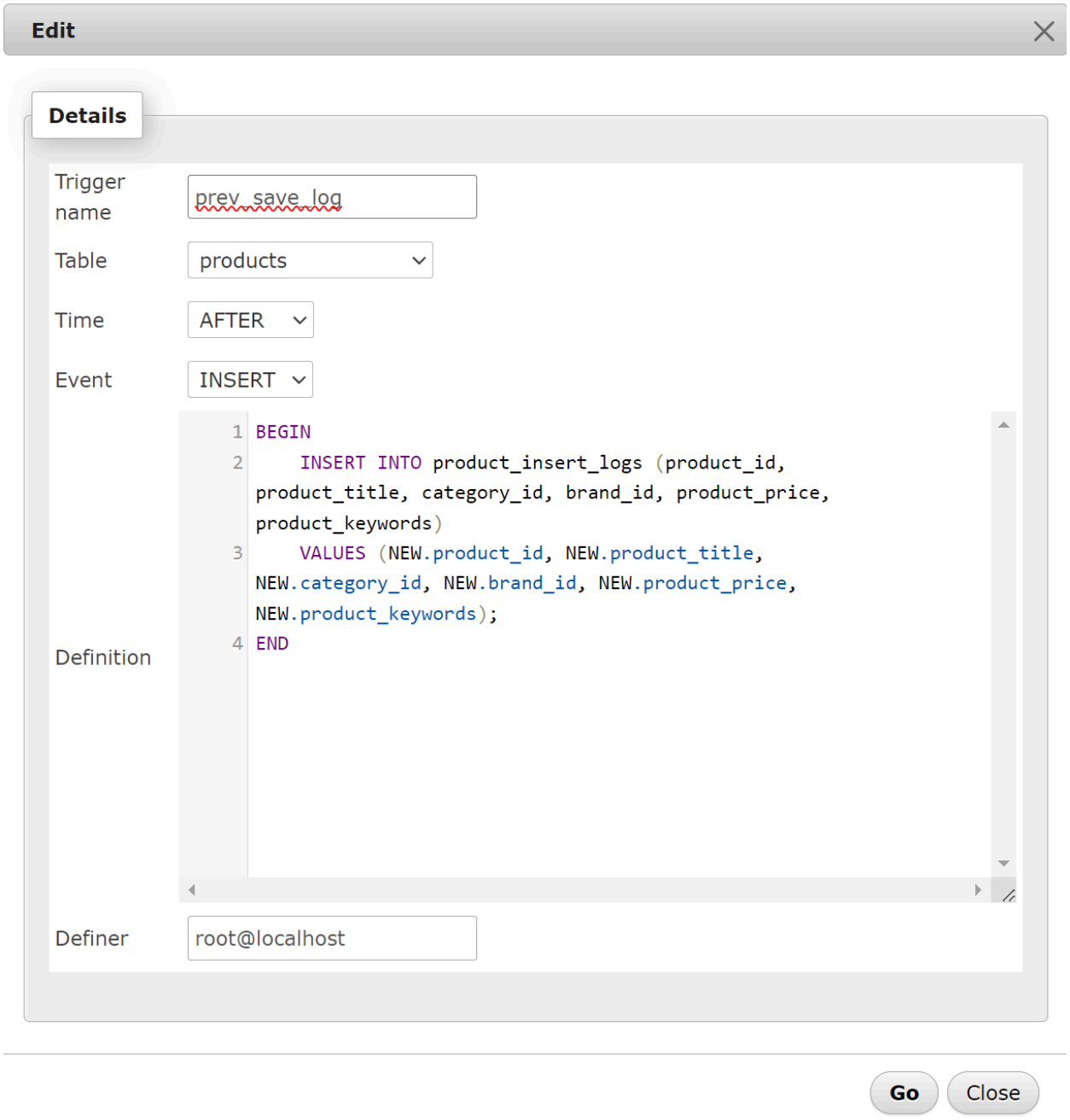
I've implemented one Trigger that handles coupons being used for a given order.

```
DELIMITER $$
CREATE TRIGGER `setCoupon` BEFORE INSERT ON `Items_purchased`
FOR EACH ROW BEGIN
    DECLARE coupon_used INT;
    DECLARE order_id_temp INT;

    -- Get the Order_ID associated with the current row
    SET order_id_temp = (SELECT Order_id FROM Order_Table WHERE Order_id = NEW.Order_id);

    -- Check if the Order_ID is valid and get the Coupon_ID associated with the order
    IF order_id_temp IS NOT NULL THEN
        SET coupon_used = (SELECT Coupon_ID FROM Order_Table WHERE Order_id = NEW.Order_id);
    ELSE
        SET coupon_used = NULL;
    END IF;

    -- Update the Coupons table to mark the coupon as used if Coupon_ID is not null
    IF coupon_used IS NOT NULL THEN
        UPDATE Coupons SET isUsed = 1 WHERE Coupon_ID = coupon_used;
    END IF;
END $$
DELIMITER ;
```

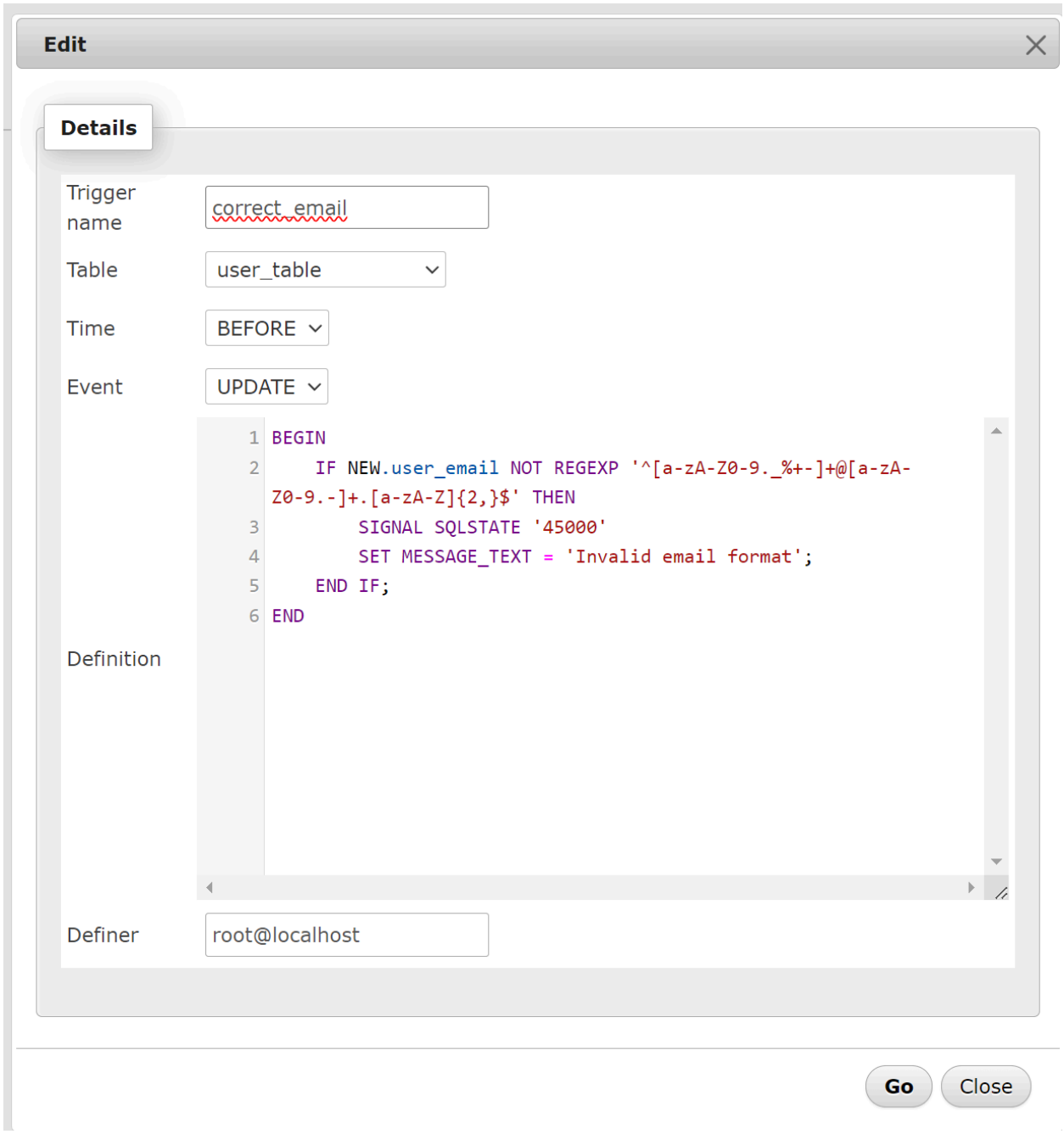
This trigger handles the product cost

```
DELIMITER $$
CREATE TRIGGER `calculateCostAndTotal` BEFORE INSERT ON `Items_purchased`
FOR EACH ROW BEGIN
    DECLARE item_cost DECIMAL(10, 2);
    DECLARE discount INT;
    DECLARE final_cost DECIMAL(10, 2);
    -- Get the current cost of the product from the Product table
    SET item_cost = (SELECT Product_cost FROM Product WHERE Product_id = NEW.Product_id);
    -- Calculate the cost of the item
    SET NEW.Cost = item_cost * NEW.Quantity;

    -- Get the discount applicable if any
    SET discount = (SELECT Discount FROM Coupons WHERE Coupon_ID = (SELECT Coupon_ID FROM Order_Table
    WHERE Order_id = NEW.Order_id));

    -- Calculate the final cost after applying discount
    IF discount IS NOT NULL THEN
        SET final_cost = NEW.Cost * (1 - (discount / 100));
    ELSE
        SET final_cost = NEW.Cost;
    END IF;
    -- Update the total cost of the order in Order_Table
    UPDATE Order_Table SET Total_Cost = Total_Cost + final_cost WHERE Order_id = NEW.Order_id;
END $$
DELIMITER ;
```

This trigger save the deleted item from cart in other table



This trigger check weather the email id entered is correct or not.



Embedded SQL Queries:

Admin inserting products

```
<!-- select categories -->
        <div class="form-outline mb-4 w-50 m-auto">
            <select name="product_category" id="product_category"
class="form-select" required="required">
                <option value="">Select Category</option>
                <?php
                $get_carts = "SELECT * FROM categories";
                $run_carts = mysqli_query($con, $get_carts);
                while($row_carts = mysqli_fetch_array($run_carts)){
                    $category_id = $row_carts['category_id'];
                    $category_title = $row_carts['category_title'];
                    echo "<option
value='$category_id'>$category_title</option>";
                }
                ?>
            </select>
        </div>
        <!-- select brands -->
        <div class="form-outline mb-4 w-50 m-auto">
            <select name="product_brand" id="product_brand"
class="form-select" required="required">
                <option value="">Select Brand</option>
                <?php
                $get_brands = "SELECT * FROM brands";
                $run_brands = mysqli_query($con, $get_brands);
                while($row_brands = mysqli_fetch_array($run_brands)){
                    $brand_id = $row_brands['brand_id'];
                    $brand_title = $row_brands['brand_name'];
                    echo "<option
value='$brand_id'>$brand_title</option>";
                }
                ?>
            </select>
        </div>
```

```php
<?php


function getproducts(){
    global $con;
    // cond to check isset or not
    if(!isset($_GET['category'])){
        if(!isset($_GET['brand'])){

    $sql = "SELECT * FROM products order by rand() LIMIT 0,6";
              $result = mysqli_query($con, $sql);
              while($row = mysqli_fetch_array($result)){
                $product_id = $row['product_id'];
                $product_title = $row['product_title'];
                $product_description = $row['product_description'];
                $product_price = $row['product_price'];
                $product_image1 = $row['product_image1'];
                $product_category = $row['category_id'];
                $product_brand = $row['brand_id'];
                echo "<div class='col-md-4  mb-2'>
                <div class='card'>
                    <img src='./images/$product_image1'
class='card-img-top' alt='$product_title'>
                    <div class='card-body'>
                         <h5 class='card-title'>$product_title</h5>
                        <p class='card-text'>$product_description</p>
                        <p class='card-price'>Price: $product_price</p>
                        <a href='index.php?add_to_cart=$product_id'
class='btn btn-info'>Add To Cart</a>
                        <a
href='product_details.php?product_id=$product_id' class='btn
btn-secondary'>View More</a>
                    </div>
                    </div>
                </div>";

            }
}}}

// getting all products
```

```php
function getallproduct(){
    global $con;
    // cond to check isset or not
    if(!isset($_GET['category'])){
        if(!isset($_GET['brand'])){

    $sql = "SELECT * FROM products order by rand()";
    $result = mysqli_query($con, $sql);
    while($row = mysqli_fetch_array($result)){
    $product_id = $row['product_id'];
    $product_title = $row['product_title'];
    $product_description = $row['product_description'];
    $product_price = $row['product_price'];
    $product_image1 = $row['product_image1'];
    $product_category = $row['category_id'];
    $product_brand = $row['brand_id'];
    echo "<div class='col-md-4  mb-2'>
    <div class='card'>
        <img src='./images/$product_image1' class='card-img-top'
alt='$product_title'>
        <div class='card-body'>
                <h5 class='card-title'>$product_title</h5>
            <p class='card-text'>$product_description</p>
            <p class='card-price'>Price: $product_price</p>
            <a href='index.php?add_to_cart=$product_id' class='btn
btn-info'>Add To Cart</a>
            <a href='product_details.php?product_id=$product_id'
class='btn btn-secondary'>View More</a>
        </div>
        </div>
    </div>";

}
}}}

function getuniquecategories(){
    global $con;
    // cond to check isset or not
    if(isset($_GET['category'])){
    $category_id = $_GET['category'];
```

```
    $sql = "SELECT * FROM products where category_id='$category_id'";
              $result = mysqli_query($con, $sql);
              $num_of_rows=mysqli_num_rows($result);
              if($num_of_rows==0){
                echo "<script>alert('No products in this
category')</script>";
                echo "<h2> No products found in this category</h2>";
                exit();
              }
              while($row = mysqli_fetch_array($result)){
                $product_id = $row['product_id'];
                $product_title = $row['product_title'];
                $product_description = $row['product_description'];
                $product_price = $row['product_price'];
                $product_image1 = $row['product_image1'];
                $product_category = $row['category_id'];
                $product_brand = $row['brand_id'];
                echo "<div class='col-md-4  mb-2'>
                <div class='card'>
                    <img src='./images/$product_image1'
class='card-img-top' alt='$product_title'>
                    <div class='card-body'>
                         <h5 class='card-title'>$product_title</h5>
                        <p class='card-text'>$product_description</p>
                        <p class='card-price'>Price: $product_price</p>
                        <a href='index.php?add_to_cart=$product_id'
class='btn btn-info'>Add To Cart</a>
                        <a
href='product_details.php?product_id=$product_id' class='btn
btn-secondary'>View More</a>
                    </div>
                    </div>
                </div>";

            }
}}


function displaybrands(){
```

```
    global $con;
    $select_brands="SELECT * FROM brands";
$run_brands=mysqli_query($con,$select_brands);
while($row_brands=mysqli_fetch_array($run_brands)){
    $brand_id=$row_brands['brand_id'];
    $brand_title=$row_brands['brand_name'];
    echo "<li class='nav-item text-center'>
    <a href='index.php?brand=$brand_id' class='nav-link
text-light'>$brand_title</a>
    </li>";
}
}
function getuniquebrands(){
    global $con;
    // cond to check isset or not
    if(isset($_GET['brand'])){
    $brand_id = $_GET['brand'];

    $sql = "SELECT * FROM products where brand_id='$brand_id'";
              $result = mysqli_query($con, $sql);
              $num_of_rows=mysqli_num_rows($result);
              if($num_of_rows==0){
                echo "<script>alert('No products in this
brand')</script>";
                echo "<h2> No products found in this brand</h2>";
                exit();
              }
              while($row = mysqli_fetch_array($result)){
                $product_id = $row['product_id'];
                $product_title = $row['product_title'];
                $product_description = $row['product_description'];
                $product_price = $row['product_price'];
                $product_image1 = $row['product_image1'];
                $product_category = $row['category_id'];
                $product_brand = $row['brand_id'];
                echo "<div class='col-md-4  mb-2'>
                <div class='card'>
                    <img src='./images/$product_image1'
class='card-img-top' alt='$product_title'>
                    <div class='card-body'>
```

```
                        <h5 class='card-title'>$product_title</h5>
                        <p class='card-text'>$product_description</p>
                        <p class='card-price'>Price: $product_price</p>
                        <a href='index.php?add_to_cart=$product_id'
class='btn btn-info'>Add To Cart</a>
                        <a
href='product_details.php?product_id=$product_id' class='btn
btn-secondary'>View More</a>
                    </div>
                    </div>
                </div>";

            }
}}

function displaycategory(){
    global $con;
    $select_categories="SELECT * FROM categories";
$run_categories=mysqli_query($con,$select_categories);
while($row_categories=mysqli_fetch_array($run_categories)){
    $category_id=$row_categories['category_id'];
    $category_title=$row_categories['category_title'];
    echo "<li class='nav-item text-center'>
    <a href='index.php?category=$category_id' class='nav-link
text-light'>$category_title</a>
    </li>";
}
}

//searching products
function searchproduct(){
    global $con;
    if(isset($_GET['search_data_product'])){
    $value = $_GET['search_data'];
    $sql = "SELECT * FROM products where product_title like '%$value%'";
    $result = mysqli_query($con, $sql);
    $num_of_rows=mysqli_num_rows($result);
    if($num_of_rows==0){
        echo "<script>alert('No products in this brand')</script>";
        echo "<h2> No products found in this brand</h2>";
```

```
        exit();
      }
    while($row = mysqli_fetch_array($result)){
    $product_id = $row['product_id'];
    $product_title = $row['product_title'];
    $product_description = $row['product_description'];
    $product_price = $row['product_price'];
    $product_image1 = $row['product_image1'];
    $product_category = $row['category_id'];
    $product_brand = $row['brand_id'];
    echo "<div class='col-md-4  mb-2'>
    <div class='card'>
        <img src='./images/$product_image1' class='card-img-top'
alt='$product_title'>
        <div class='card-body'>
                <h5 class='card-title'>$product_title</h5>
            <p class='card-text'>$product_description</p>
            <p class='card-price'>Price: $product_price</p>
            <a href='index.php?add_to_cart=$product_id' class='btn
btn-info'>Add To Cart</a>
            <a href='product_details.php?product_id=$product_id'
class='btn btn-secondary'>View More</a>
        </div>
        </div>
    </div>";

}
}}

// view details
function viewdetails(){
    global $con;
    // cond to check isset or not
    if(isset($_GET['product_id'])){
    if(!isset($_GET['category'])){
        if(!isset($_GET['brand'])){
        $product_id=$_GET['product_id'];
    $sql = "SELECT * FROM products where product_id = $product_id";
    $result = mysqli_query($con, $sql);
    while($row = mysqli_fetch_array($result)){
```

```
    $product_id = $row['product_id'];
    $product_title = $row['product_title'];
    $product_description = $row['product_description'];
    $product_price = $row['product_price'];
    $product_image1 = $row['product_image1'];
    $product_image2=$row['product_image2'];
    $product_image3=$row['product_image3'];
    $product_category = $row['category_id'];
    $product_brand = $row['brand_id'];
    echo "<div class='col-md-4  mb-2'>
    <div class='card'>
        <img src='./images/$product_image1' class='card-img-top'
alt='$product_title'>
        <div class='card-body'>
                <h5 class='card-title'>$product_title</h5>
            <p class='card-text'>$product_description</p>
            <p class='card-price'>Price: $product_price</p>
            <a href='index.php?add_to_cart=$product_id' class='btn
btn-info'>Add To Cart</a>
            <a href='product_details.php?product_id=$product_id'
class='btn btn-secondary'>View More</a>
        </div>
        </div>
    </div>
    <div class='col-md-8'>
        <div class='row'>
            <div class='col-md-12'>
                <h4 class='text-center text-info mb-5'>related
products</h4>
            </div>
            <div class='col-md-6'>
                <img src='./images/$product_image2' class='card-img-top'
alt='$product_title'>
            </div>
            <div class='col-md-6'>
                <img src='./images/$product_image3' class='card-img-top'
alt='$product_title'>
            </div>
        </div>
    </div>";
```

```php
}
}}}
}

//cart func

function cart(){
if(isset($_GET['add_to_cart'])){
global $con;
$ip = getIPAddress();
$product_id=$_GET['add_to_cart'];
$sql = "SELECT * FROM cart_details where ip_address='$ip' AND
product_id='$product_id'";
$result = mysqli_query($con, $sql);
if (mysqli_num_rows($result) > 0) {
    echo "<script>alert('Product already added to cart')</script>";
    echo "<script>window.open('index.php','_self')</script>";
}else{
    $insert_query = "INSERT INTO
cart_details(ip_address,product_id,quantity)
VALUES('$ip','$product_id',0)";
    $result = mysqli_query($con, $insert_query);
    echo "<script>alert('Product added to cart')</script>";
    echo "<script>window.open('index.php','_self')</script>";

}
}}

//func to get cart item num

function getcartitem(){
    if(isset($_GET['add_to_cart'])){
        global $con;
        $ip = getIPAddress();
        $sql = "SELECT * FROM cart_details where ip_address='$ip'";
        $result = mysqli_query($con, $sql);
        $count_cart_item=mysqli_num_rows($result);
    }else{
        global $con;
```

```
        $ip = getIPAddress();
        $sql = "SELECT * FROM cart_details where ip_address='$ip'";
        $result = mysqli_query($con, $sql);
        $count_cart_item=mysqli_num_rows($result);
        }
        echo $count_cart_item;
}

//total price func

function totalprice(){
    global $con;
    $ip = getIPAddress();
    $total=0;
    $sql = "SELECT * FROM cart_details where ip_address='$ip'";
    $result = mysqli_query($con, $sql);
     while ($row=mysqli_fetch_array($result)){
         $product_id=$row['product_id'];
         $sql1 = "SELECT * FROM products where product_id='$product_id'";
         $result1 = mysqli_query($con, $sql1);
         while ($row1=mysqli_fetch_array($result1)){
             $product_price=array($row1['product_price']);
             $product_values=array_sum($product_price);
             $total=$total+$product_values;
         }
     }
    echo $total;
}
?>
```