Oasis-An Online Retail Store

Harsh Parimal Popat 2021048
Harshil Mital 2021050

Project Scope

This project aims to create an end-to-end web application for an online retail store, Oasis. The application will have a working user interface, but the focus will be on creating a fully functional back end.

Oasis will aim to provide its customers with the ability to browse and order from an extensive catalog of products. Customers will be able to add products to their cart , make payments, and avail of discounts through discount codes.
It will provide the pickup/delivery address and delivery payment status to the delivery partners.
Enables the administrator multiple functionalities such as addition and deletion of products, generation of new deals and discount codes. Also gives access to inventory data and other statistical information.

It will enable ease of inventory management by maintaining a complete database of products, suppliers, delivery partners, and customers and capturing interactions between the various entities.

Technical Requirements

Tech Stack:

Database Management: MySQL
Front-End (provisional): React.js, Bootstrap, HTML, CSS, JavaScript
Back-End (provisional): Node.js, JavaScript

- Delivery partners can only carry out one delivery at a time. Another delivery is only allotted to them only when the ongoing delivery is completed.

- A customer can only make one order at a time. They cannot order another set of items until their ongoing order has been delivered and payment is complete.

- Customers can only apply one discount code in each order. By default the highest available discount will be applied.

- Customers can make just one account using the same email address/phone number.

- Delivery partners are only allotted to deliver an order if they are in the same city as the drop location.

- Account details for delivery partners are generated by the administrator and by the delivery partners themselves.

- **Atomicity** will be ensured by making sure that transactions take place at once or else won't take place.

- The application would ensure database **consistency** before and after the transaction.
- **Data Constraints** will be put in place in such a way that sensitive data such as user passwords, bank details are not visible to anyone else including the administrator.

Functional Requirements

Customer Functionality

- Login
- Create Account
- Browse and filter through products
- Review/Checkout Cart
- Add/Delete items to/from cart
- Apply discount code
- Select payment option and make payment
- Check delivery status/select date of delivery
- Give product feedback

Delivery Partner Functionality

- Sign in using account details given by admin
- Update delivery status
- Gets the package details (pickup/drop location) and delivery time
- Accept payment and then send it to the store.

Administrator Functionality

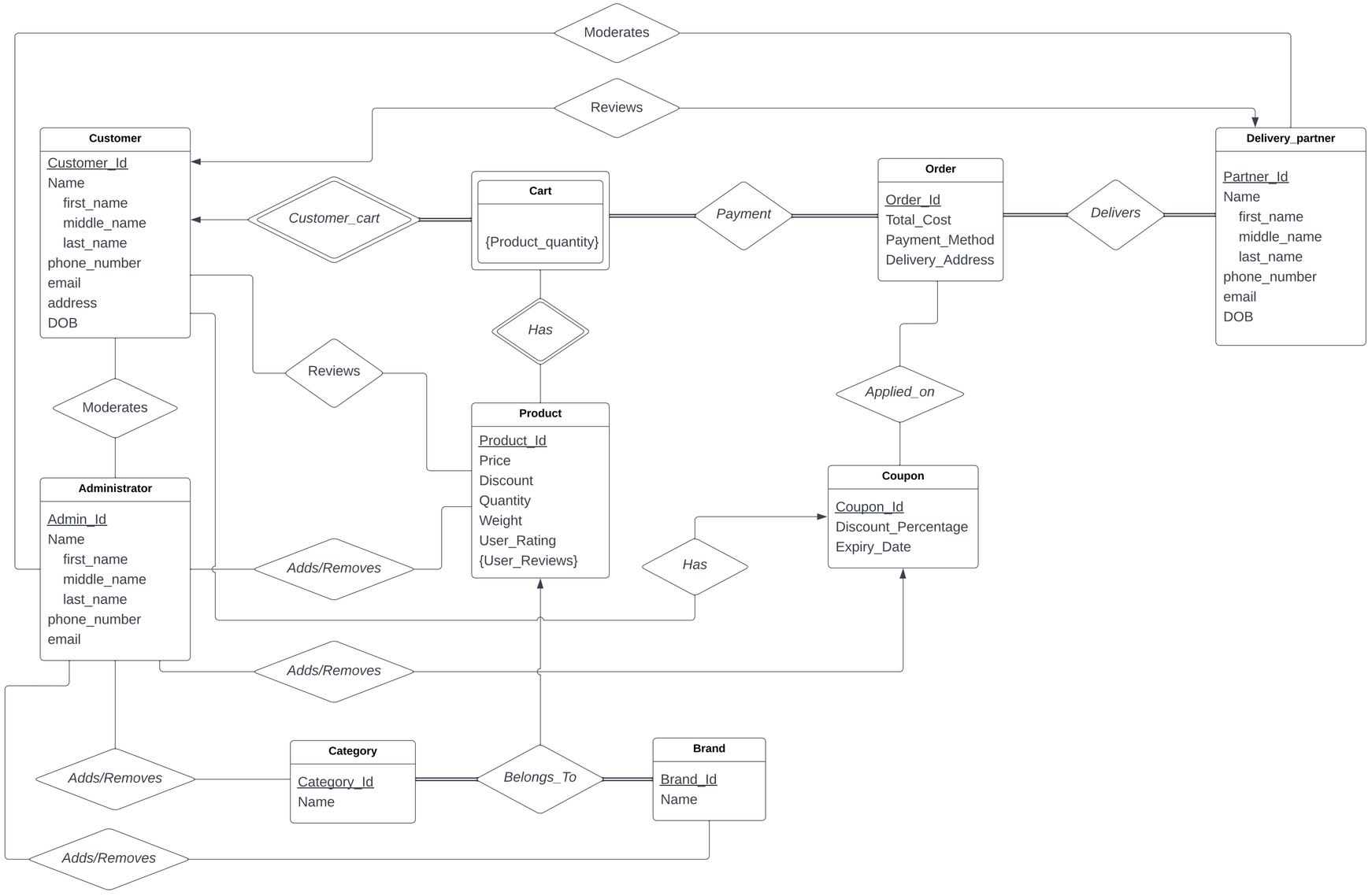
- Manage user accounts
- Add/Update/Delete products
- Create discount codes
- Update product prices/discounts
- Check and approve the list of sellers
- Check sales history
- Access financial records
- Access/update inventory details

Oasis - Entity Relationship Model

Harsh Parimal Popat **2021048**
Harshil Mital **2021050**

Entities

- Customer
- Delivery_partner
- Product
- Brand
- Category
- Administrator
- Coupons
- Order
- Cart



Oasis - Relational Model

Harsh Parimal Popat **2021048**
Harshil Mital **2021050**

Primary Key Foreign Key

Customer

Customer_ID	first_name	middle_name	last_name	phone_number	email	address	DOB

Cart_Products

Customer_ID	Product_ID

Product

Product_ID	Price	Discount	Quantity	Weight	User_Rating	Brand_ID	Category_ID

User_Reviews

Customer_ID	Product_ID	Review

Brand

Brand_ID	Name

Category

Category_ID	Name

Administrator

Admin_ID	first_name	middle_name	last_name	phone_number	email

Delivery Partner

Partner_ID	first_name	middle_name	last_name	phone_number	email	address	DOB

Coupon

Coupon_ID	Discount	Expiry

User_Coupons

Customer_ID	Coupon_ID

Order

Order_ID	Total_Cost	Payment_metho	Delivery_Add

Order_Products

Order_ID	Customer_ID	Product_ID

Order_Coupons

Order_ID	Coupon_ID

Order_Delivery

Order_ID	Partner_ID

Oasis-An Online Retail Store

Harsh Parimal Popat 2021048
Harshil Mital 2021050

MySQL queries for the creation of tables:

```
create table Customer (Customer_ID int NOT NULL
AUTO_INCREMENT, First_Name varchar(20) NOT NULL,
Middle_Name varchar(20), Last_Name varchar(20) NOT
NULL, Phone_Number char(10) NOT NULL, Email varchar(50)
NOT NULL, Address varchar(200) NOT NULL, DOB DATE NOT
NULL, PRIMARY KEY (Customer_ID));
Create table Coupon (Coupon_ID int NOT NULL
AUTO_INCREMENT,Discount float NOT NULL,Expiry DATE NOT
NULL, PRIMARY KEY (Coupon_ID));
create table Brand(Brand_ID int NOT NULL
AUTO_INCREMENT, Name varchar(20) NOT NULL, PRIMARY KEY
(Brand_ID));
create table Category(Category_ID int NOT NULL
AUTO_INCREMENT, Name varchar(20) NOT NULL, PRIMARY KEY
(Category_ID));
create table Orders(Order_ID int NOT NULL
AUTO_INCREMENT,Total_Cost float NOT NULL,
Payment_Method varchar(20) NOT NULL, Delivery_Address
varchar(100) NOT NULL, PRIMARY KEY (Order_ID));
Create table Product(Product_ID int NOT NULL
AUTO_INCREMENT,Price float NOT NULL,Discount float NOT
NULL, Quantity int NOT NULL,Weight float,User_Rating
```

```
int,Brand_ID int NOT NULL,Category_ID int NOT NULL,
PRIMARY KEY (Product_ID));
create table Administrator(Admin_ID int NOT NULL
AUTO_INCREMENT,first_name varchar(20) NOT NULL,
middle_name varchar(20), last_name varchar(20) NOT
NULL,phone_number char(10) NOT NULL,email varchar(50)
NOT NULL, PRIMARY KEY (Admin_ID));
create table Delivery_Partner(Partner_ID int NOT NULL
AUTO_INCREMENT,first_name varchar(20) NOT NULL,
middle_name varchar(20), last_name varchar(20) NOT
NULL, phone_number char(10) NOT NULL, email varchar(50)
NOT NULL, address varchar(100) NOT NULL, DOB DATE NOT
NULL, PRIMARY KEY (Partner_ID));
create table Cart_Products(Customer_ID int NOT NULL,
Product_ID int NOT NULL, FOREIGN KEY (Customer_ID)
REFERENCES Customer(Customer_ID));
create table User_Reviews(Customer_ID int NOT NULL
,Product_ID int NOT NULL, Review varchar(200) NOT
NULL);
create table User_Coupon(Customer_ID int NOT NULL,
Coupon_ID int NOT NULL);
create table Order_Products(Order_ID int NOT NULL ,
Customer_ID int NOT NULL, Product_ID int NOT NULL);
create table Order_Coupons(Order_ID int NOT NULL,
Coupon_ID int NOT NULL);
create table Order_Delivery (Order_ID int NOT NULL,
Partner_ID int NOT NULL);
Alter table Cart_Products ADD FOREIGN KEY (Product_ID)
REFERENCES Product(Product_ID);
Alter table product ADD FOREIGN KEY (Brand_ID)
REFERENCES brand(Brand_ID);
Alter table product ADD FOREIGN KEY (Category_ID)
REFERENCES category(Category_ID);
Alter table User_Reviews ADD FOREIGN KEY (Customer_ID)
REFERENCES Customer(Customer_ID);
```

```
Alter table User_Reviews ADD FOREIGN KEY (Product_ID)
REFERENCES Product(Product_ID);
Alter table User_Coupon ADD FOREIGN KEY (Customer_ID)
REFERENCES Customer(Customer_ID);
Alter table User_Coupon ADD FOREIGN KEY (Coupon_ID)
REFERENCES Coupon(Coupon_ID);
Alter table Order_Products ADD FOREIGN KEY (Order_ID)
REFERENCES Orders(Order_ID);
Alter table Order_Products ADD FOREIGN KEY
(Customer_ID) REFERENCES Customer(Customer_ID);
Alter table Order_Products ADD FOREIGN KEY (Product_ID)
REFERENCES Product(Product_ID);
Alter table Order_Coupons ADD FOREIGN KEY (Order_ID)
REFERENCES Orders(Order_ID);
Alter table Order_Coupons ADD FOREIGN KEY (Coupon_ID)
REFERENCES Coupon(Coupon_ID);
Alter table Order_Delivery ADD FOREIGN KEY (Order_ID)
REFERENCES Orders(Order_ID);
Alter table Order_Delivery ADD FOREIGN KEY (Partner_ID)
REFERENCES Delivery_Partner(Partner_ID);
```

We have generated the data for the following using an online data generator (https://filldb.info/)

Table Name	No of Rows
Administrator	**20**
Brand	**100**
Cart_products	**500**
Category	**50**

Coupon	**700**
Customer	**1000**
Delivery_Partner	**150**
Order_Coupons	**50**
Order_Delivery	**75**
Order_Products	**75**
Orders	**75**
Product	**1500**
User_Coupon	**100**
User_Reviews	**3000**

Oasis-An Online Retail Store

Harsh Parimal Popat 2021048
Harshil Mital 2021050

Project Deadline - 4

Queries to demonstrate constraints

1. **To segregate the products on the basis of good (rating > 3), moderate (rating = 3) and bad (rating < 3)**

```
SELECT
    product_ID,
    Price,
    Discount,
    Quantity,
    Brand_ID,
    Category_ID
    User_Rating,

    CASE
        WHEN User_Rating > 3 THEN 'good'
        WHEN User_Rating = 3 THEN 'moderate'
        ELSE 'bad'
    END AS Ratings
FROM product;
```

2. **To add a coupon worth 20% to any person with more than 1 order.**

```
SET @coupon_id = NULL;

INSERT INTO coupon (`Discount`, `Expiry`)
SELECT 20, DATE_ADD(NOW(), INTERVAL 30 DAY)
WHERE EXISTS (
  SELECT 1
  FROM order_products
  WHERE Customer_ID IN (
```

```
    SELECT Customer_ID
    FROM order_products
    GROUP BY Customer_ID
    HAVING COUNT(`Order_ID`) > 1
  )
);

SET @coupon_id = LAST_INSERT_ID();

INSERT INTO user_coupon (Customer_ID, Coupon_ID)
SELECT DISTINCT order_products.`Customer_ID`, @coupon_id
FROM order_products
WHERE Customer_ID IN (
  SELECT Customer_ID
  FROM order_products
  GROUP BY Customer_ID
  HAVING COUNT(`Order_ID`) > 1
);
```

3. **To give a coupon worth 30% to all customers whose birthday it is today.**

```
SET @coupon_id = NULL;

INSERT INTO coupon (`Discount`, `Expiry`)
SELECT 20, DATE_ADD(NOW(), INTERVAL 30 DAY)
WHERE EXISTS (
  SELECT 1
  FROM customer
  WHERE DAYOFYEAR(`DOB`) = DAYOFYEAR(NOW())
);

SET @coupon_id = LAST_INSERT_ID();

INSERT INTO user_coupon (Customer_ID, Coupon_ID)
SELECT customer.`Customer_ID`, @coupon_id
FROM customer
WHERE DAYOFYEAR(`DOB`) = DAYOFYEAR(NOW());
```

4. **To count the number of orders made by UPI, Card and Cash and the total cost earning from each payment method**

```
SELECT
  payment_method,
  COUNT(*) AS Transaction_Type_Count,
  ROUND(SUM(total_cost),2) AS Total_Earning
FROM
  orders
GROUP BY
  payment_method;
```

5. **To group products by categories and further group them on the basis of brand in their respective categories and then sort them according to prices.**

```
SELECT c.Name AS Category, b.Name AS Brand, p.Price
FROM Category c
JOIN Product p ON c.Category_ID = p.Category_ID
JOIN Brand b ON b.Brand_ID = p.Brand_ID
ORDER BY Category, Brand, Price ASC;
```

6. **To delete a category/brand if the number of products in that category/brand is 0.**

```
DELETE p, c
FROM product p
JOIN category c ON p.category_id = c.category_id
WHERE p.quantity = 0
AND NOT EXISTS (
      SELECT *
      FROM product p2
      WHERE p2.category_id = p.category_id
      AND p2.quantity > 0
);
```

7. **To delete the order after it has been delivered**

```
DELIMITER //

CREATE TRIGGER delete_order AFTER UPDATE ON orders
FOR EACH ROW
    IF NEW.order_delivery_status = 'Delivered' THEN
        DELETE FROM orders WHERE Order_ID = OLD.Order_ID;
    END IF;

//

DELIMITER ;
```

8. **To delete the coupons if they have expired.**

```
CREATE TRIGGER delete_expired_coupons
BEFORE INSERT ON Coupon
FOR EACH ROW
  DELETE FROM Coupon
  WHERE Expiry < CURDATE();
```

9. **To return top products from each category. Also we can check for the adequate quantity of product , or the admin can order more if the quantity had become less.**

```
SELECT p1.product_id,
p1.user_rating,p1.Quantity,p1.price,p1.Brand_ID p1.category_id
FROM product p1
WHERE (
  SELECT COUNT(*)
  FROM product p2
  WHERE p2.category_id = p1.category_id
    AND p2.user_rating > p1.user_rating
) < 5
ORDER BY p1.category_id, p1.user_rating DESC;
```

10. **To return the maximum ordered category from different age ranges of users.**

```
SELECT
  CASE
```

```
    WHEN YEAR(CURDATE()) - YEAR(c.dob) BETWEEN 20 AND 30 THEN
'20-30'
    WHEN YEAR(CURDATE()) - YEAR(c.dob) BETWEEN 30 AND 40 THEN
'30-40'
    WHEN YEAR(CURDATE()) - YEAR(c.dob) BETWEEN 40 AND 50 THEN
'40-50'
    WHEN YEAR(CURDATE()) - YEAR(c.dob) BETWEEN 50 AND 60 THEN
'50-60'
    ELSE '60+'
  END AS age_group,
  COUNT(op.order_id) AS order_count
FROM
  customer c
  INNER JOIN order_products op ON c.customer_id = op.customer_id
GROUP BY
  age_group;
```

11. To remove all delivery partners whose age is > 40.

```
DELETE FROM order_delivery
WHERE partner_ID IN (
    SELECT partner_ID
    FROM delivery_partner
    WHERE TIMESTAMPDIFF(YEAR, DOB, CURDATE()) >= 40
);

DELETE FROM delivery_partner
WHERE TIMESTAMPDIFF(YEAR, DOB, CURDATE()) >= 40;
```

12. To display customers and their cart prices.

```
SELECT c.customer_id,
        c.first_name,
        c.last_name,
        c.phone_number,
        c.email,
        ROUND(SUM(p.price),2) AS total_cost

FROM customer c
JOIN cart_products cp ON c.customer_id = cp.customer_id
JOIN product p ON cp.product_id = p.product_id
GROUP BY c.customer_id;
```

Oasis-An Online Retail Store

Harsh Parimal Popat 2021048
Harshil Mital 2021050

Project Deadline - 5

EMBEDDED QURIES OLAP QUERIES AND TRIGGERS

EMBEDDED QURIES:

1.Select *from customer;

2.Insert into customer values (We input the values from the user);

OLAP QUERIES:

1.Total revenue generated by each brand and category:

```
SELECT b.Brand_ID AS Brand_ID, c.Category_ID AS Category_ID,
SUM(o.Total_Cost) AS Revenue
FROM Orders o
JOIN Order_Products op ON o.Order_ID = op.Order_ID
JOIN Product p ON op.Product_ID = p.Product_ID
JOIN Brand b ON p.Brand_ID = b.Brand_ID
JOIN Category c ON p.Category_ID = c.Category_ID
GROUP BY b.Name, c.Name WITH ROLLUP;
```

2. Group by user rating and Brand_ID from cube to check the sum of Quantity and Price

For Cube : there is only rollup so we use union to do :

```
Select Brand_ID,User_Rating,sum(Quantity) AS Total Quantity,
AVG(price) AS average_price
from product
Group by Brand_ID,User_Rating WITH ROLLUP
UNION
Select Brand_ID,User_Rating,sum(Quantity),AVG(price) AS
average_price
from product
Group by User_Rating, Brand_ID WITH ROLLUP;
```

3. To count the number of orders made by UPI, Card and Cash and the total cost earning from each payment method

```
SELECT
payment_method,
COUNT(*) AS Transaction_Type_Count,
ROUND(SUM(total_cost),2) AS Total_Earning
FROM
orders
GROUP BY
payment_method;
```

4.To get the total cost of all orders by payment method, brand, and category, with subtotals for each payment method, brand, and category:

```
SELECT Payment_Method, b.Name AS Brand_Name, c.Name AS
Category_Name, SUM(Total_Cost) AS Total_Cost
FROM Orders o
JOIN Order_Products op ON o.Order_ID = op.Order_ID
JOIN Product p ON op.Product_ID = p.Product_ID
JOIN Brand b ON p.Brand_ID = b.Brand_ID
JOIN Category c ON p.Category_ID = c.Category_ID
GROUP BY Payment_Method, b.Name, c.Name WITH ROLLUP;
```

5. To return the maximum ordered category from different age ranges of users.

```
SELECT
CASE

WHEN YEAR(CURDATE()) - YEAR(c.dob) BETWEEN 20 AND 30 THEN
'20-30'
WHEN YEAR(CURDATE()) - YEAR(c.dob) BETWEEN 30 AND 40 THEN
'30-40'
WHEN YEAR(CURDATE()) - YEAR(c.dob) BETWEEN 40 AND 50 THEN
'40-50'
WHEN YEAR(CURDATE()) - YEAR(c.dob) BETWEEN 50 AND 60 THEN
'50-60'
ELSE '60+'
END AS age_group,
COUNT(op.order_id) AS order_count
FROM
customer c
INNER JOIN order_products op ON c.customer_id = op.customer_id
GROUP BY
age_group;
```

6. Total loss generated from the discounts given to the user to stay on the website and buy the products used for analysis.

```
SELECT
    COALESCE(Brand.Name, 'Total') AS Brand,
    COALESCE(Category.Name, 'Total') AS Category,
    SUM(Product.Quantity * (Product.Price - Product.Price *
(Product.Discount))) AS Loss
FROM
    Product
    JOIN Brand ON Product.Brand_ID = Brand.Brand_ID
    JOIN Category ON Product.Category_ID = Category.Category_ID
    JOIN Coupon ON Product.Product_ID = Coupon.Coupon_ID
GROUP BY
    Brand.Name, Category.Name
    WITH ROLLUP;
```

TRIGGERS;

1.Set weight of item to 0 if the weight values is NULL.

```
DELIMITER $$

CREATE TRIGGER set_weight_to_zero
BEFORE INSERT
ON product FOR EACH ROW
BEGIN
  IF NEW.weight IS NULL THEN
    SET NEW.weight = 0;
  END IF;
END$$
```

2. Set the price of the product to 0 if the quantity of the product gets update to 0.

```
DELIMITER $$

 create trigger remove_product
  before update on product
 for each row
   begin
  if NEW.quantity=0 then
    set NEW.price = 0;
End if;
END$$
```

Test case:
select * from product where category_id=1;
update product set Quantity=0 where product_ID=591;
select * from product where category_id=1;

3. To delete the order after it has been delivered

```
DELIMITER //
CREATE TRIGGER delete_order AFTER UPDATE ON orders
FOR EACH ROW
IF NEW.order_delivery_status = 'Delivered' THEN
DELETE FROM orders WHERE Order_ID = OLD.Order_ID;
END IF;
//
DELIMITER ;
```

4. To delete the coupons if they have expired.

```
CREATE TRIGGER delete_expired_coupons
BEFORE INSERT ON Coupon
FOR EACH ROW
DELETE FROM Coupon
WHERE Expiry < CURDATE();
```

Oasis-An Online Retail Store

Harsh Parimal Popat 2021048
Harshil Mital 2021050

Project Deadline-6
Transactions

T1: Customer (`Customer_ID`) places the order (`Order_ID`) for the product (`Product_ID`) in their cart to be delivered to the address (`Address`).

R(Product_ID from Cart_Products with Customer_ID)
R(Quantity from Product with Product_ID)
Quantity = Quantity - 1
W(Quantity)
R(Price from Product with Product_ID) // to charge the customer

T2: Admin updates the quantity and price of the product (Product_ID)

R(Quantity from Product with Product_ID)
Quantity = Quantity + 100
W(Quantity)
R(Price from Product with Product_ID)
Price = Price - 250
W(Price)

Conflict Serializable

T1	T2
R(Product_ID from Cart_Products with Customer_ID)	
R(Quantity from Product with Product_ID) Quantity = Quantity - 1	
W(Quantity)	
	R(Price from Product with Product_ID) Price = Price - 250
	R(Quantity from Product with Product_ID) Quantity = Quantity + 100
	W(Quantity)
R(Price from Product with Product_ID)	
	W(Price)

Non-Conflict Serializable

T1	T2
R(Product_ID from Cart_Products with Customer_ID)	
	R(Quantity from Product with Product_ID) Quantity = Quantity + 100
R(Quantity from Product with Product_ID) Quantity = Quantity - 1	
	W(Quantity)
W(Quantity)	
	R(Price from Product with Product_ID) Price = Price - 250
	W(Price)
R(Price from Product with Product_ID)	

Oasis-An Online Retail Store

Harsh Parimal Popat 2021048
Harshil Mital 2021050

Project Deadline-6
USER GUIDE

The OASIS is a online shopping platform here you can buy anything you want.
It is made in python (CLI).
In first we get the login page where you are given 3 options :
To Enter as :
Administrator
Customer
Delivery Partner

For logging in you enter the id given to you that is the customer_id , Adminstrator_id, Delivery_partner_id. For Password bases we have given the phone number of the person logging in as the phone number of a person is unique.

For Administrator:
After logging in you get the options as:
You get a menu driven program:
1. Print the details of the logged in administrator
2. Print the products (All products, Particular Product by product id)
3. Add products to the database

4. Update product details already in the database
5. Print the Brand Details or Add a new brand similar for Category
6. Print the Delivery partner and Customer and order details(even for a particular customer details of the order) from the id provided.
7. Print the Total Sales till date and the mode of payment done

For Customer:

1. Print the customer Details or An option to chage the name address or anything related to the customer details
2. Browse Products and add them to the cart from the product id
3. Get all the customers not delivered order or the delivered order and the delivery person details if not delivered
4. To display the customers cart and the cart total , you get an option to check out or empty your cart , while checking you can apply coupons available to the particular customer and select the mode of payment , all the update and insertion done in this option
5. To display all your available coupon with the customer

For Delivery Partner:

1. The delivery partner can see his details and all
2. He can see the orders that are given to him and delivery them to the customer get all the details all at once
3. He can change the order delivery status to delivered as soon as the order is delivered and the then by trigger the product get removed from the tables accordingly