



Report on

---

## **Omnimart : E-Commerce Database**

---

May 25, 2024

Prepared By:

2211247042, Fuwad Hasan

2211023642, Maharun Afroz

2211987042, Al-Amin Rabbi

Course Instructor:

Prof. Dr. Kamruddin Nur

## Contents

<b>1 Project Title: Omnimart, E-commerce Database</b>	<b>6</b>
<b>2 Project Description</b>	<b>6</b>
<b>3 Project Objective</b>	<b>6</b>
<b>4 Project Deliverable</b>	<b>6</b>
<b>5 Database Design</b>	<b>7</b>
5.1 Conceptual Design Diagram . . . . .	7
5.2 Logical Design Diagram . . . . .	7
5.3 Physical Design Diagram . . . . .	8
<b>6 Implementation</b>	<b>9</b>
6.1 Data Population . . . . .	9
6.2 Data Manipulation . . . . .	9
<b>7 Query Implementation</b>	<b>10</b>
7.1 User Queries . . . . .	10
7.2 SQL Queries . . . . .	11
<b>8 Database Testing</b>	<b>23</b>
<b>9 Conclusion</b>	<b>25</b>
<b>10 Acknowledgements</b>	<b>25</b>

## List of Figures

1	Conceptual design of the database . . . . .	7
2	Logical design of the database . . . . .	7
3	Physical design of the database . . . . .	8

## List of Tables

1	<a href="#">Team Contributions</a>	5
---	------------------------------------	---

## Project Overview

Omnimart E-commerce database project will ensure a structured data storage of a vast number of customers and sellers. There will be a huge inventory for the product information in category with reviews. The project will also manage the data of the orders and offers on products with cart and wishlist. We will also have an internal employees and admin data set who will manage this e-commerce site.

## Contributions

**Table 1:** Team Contributions

ID	Name	Tasks	Percentage
2211247042	Fuwad Hasan	-	-
2211023642	Maharun Afroz	-	-
2211987042	Al-Amin Rabbi	-	-

# **1 Project Title: Omnimart, E-commerce Database**

## **2 Project Description**

Omnimart e-commerce database project aims to develop database for an all-in-one e-commerce site which will connect sellers with customers. User Friendly interface and seamless experience will be our top notch priority. Users will be able to register their accounts and order products right away. Users can save their favourite products to wishlist for later purchase and save orders for reordering and get offers based on their purchase history. Sellers can add, update and manage their products. Products will be categorized for easy navigation and users can filter products based on their preferences. Customers can add products from multiple sellers and proceed to a secure checkout process. Customers can choose from different payment options such as mobile banking, Cash on Delivery, Card payment etc. Sellers will have access to their dashboard where they can manage their incoming orders and process them for on time delivery. Customers will be able to track their order status in real time. Customers will be able to leave reviews and ratings for the products they have purchased. These reviews will be displayed on the product pages to enhance transparency and trust. Products will be ranked based on customer reviews. Our database will handle these functionalities in low latency and efficiently.

## **3 Project Objective**

The objectives of this project are:

1. Provide an all-in-one e-commerce platform.
2. Connects sellers and customers for convenient price and quality products.
3. Customers can enlist products for future purchases.
4. Sellers can update their products from a single place.
5. Reviews and rating system is available to enhance transparency and trust.
6. Payments are secure and maintain concurrency.
7. Purchase history is available at all times.
8. Handle a large volume of queries efficiently.

## **4 Project Deliverable**

1. Requirement Analysis
2. Database Design
  - Conceptual ERD
  - Logical ERD
  - Physical ERD
3. Project Implementation and Demonstration

## 5 Database Design

### 5.1 Conceptual Design Diagram

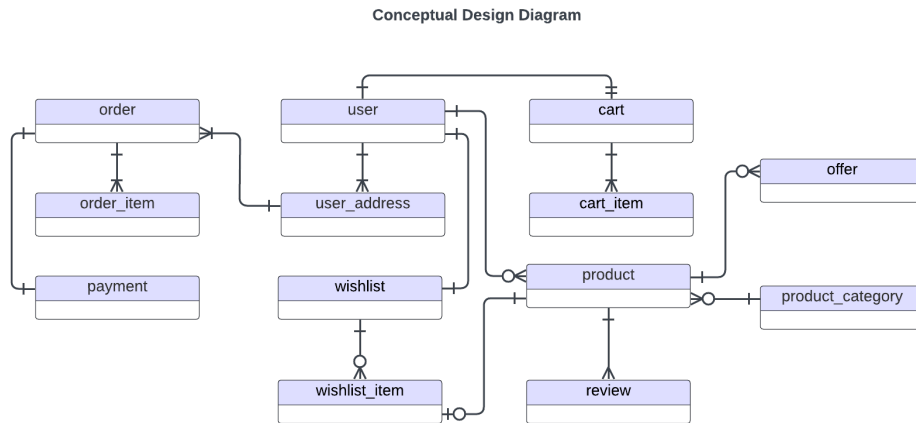


Figure 1: Conceptual design of the database

### 5.2 Logical Design Diagram

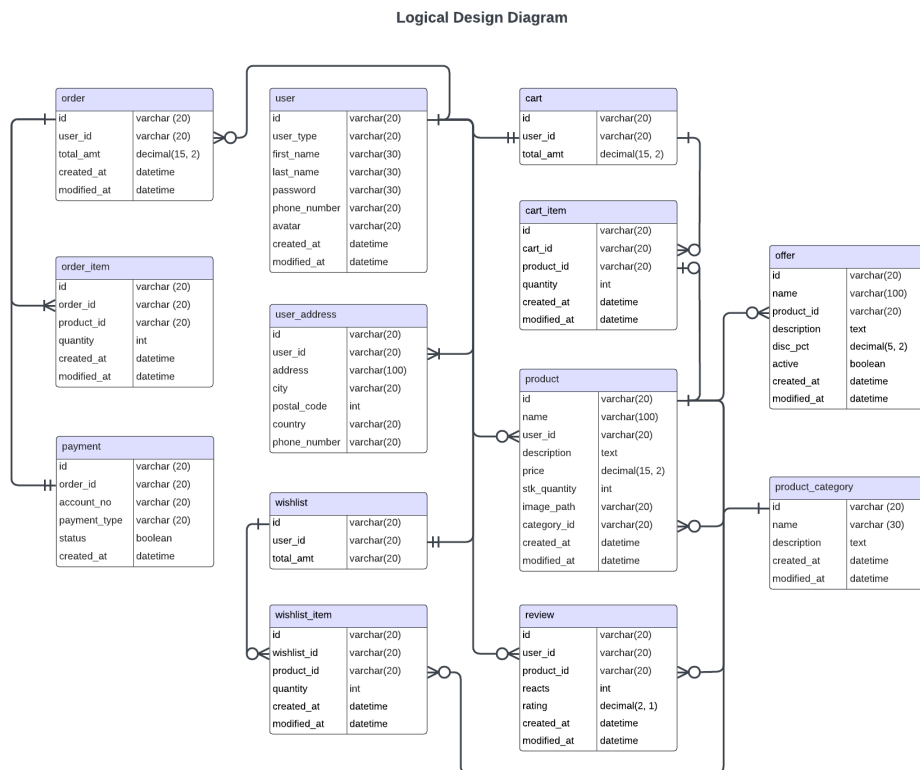


Figure 2: Logical design of the database

### 5.3 Physical Design Diagram

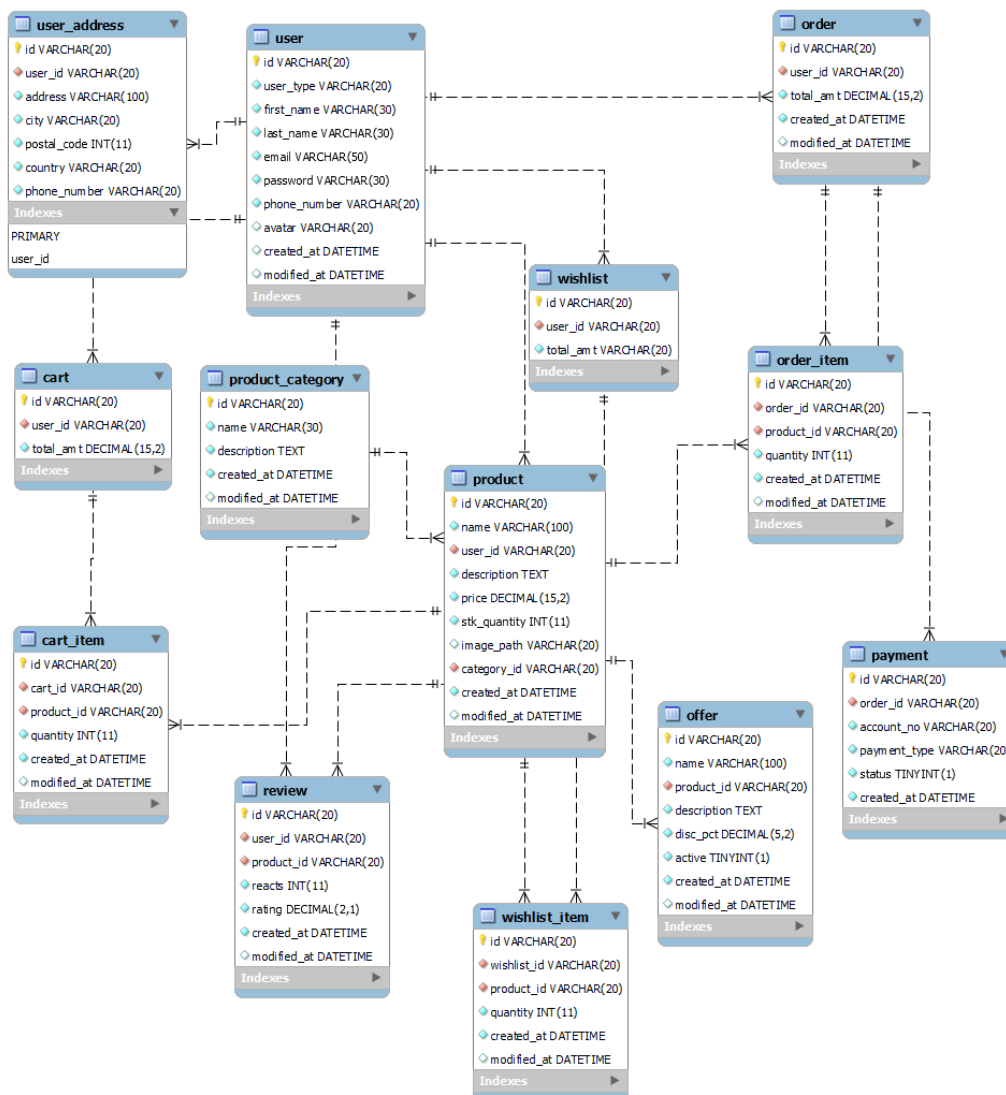


Figure 3: Physical design of the database



## 6 Implementation

### 6.1 Data Population

We populated the database by first creating SQL code based on the tables outlined in the ER Diagram. Then, we generated random data for each table and inserted it into the database.

Here's an example of one of tables.

**Listing 1:** Data Population Example

```
create database if not exists 'omnimart';

CREATE TABLE 'user_address' (
  'id' varchar(20) not null,
  'user_id' varchar(20) not null,
  'address' varchar(100) not null,
  'city' varchar(20) not null,
  'postal_code' int not null,
  'country' varchar(20) not null,
  'phone_number' varchar(20) not null,
  primary key ('id'),
  FOREIGN KEY ('user_id') REFERENCES 'user'('id')
);

-- Inserting data into the user_address table with unique address IDs
INSERT INTO 'user_address' ('id', 'user_id', 'address', 'city', 'postal_code', 'country', 'phone_number')
VALUES

-- Addresses for employees
('21001', 'EMP00001', '123_Main_St', 'New_York', 10001, 'USA', '1234567890'),
('21002', 'EMP00002', '456_Oak_Ave', 'Los_Angeles', 90001, 'USA', '5559876543'),

-- Addresses for sellers
('22001', 'SEL00001', '123_Red_St', 'New_York', 10001, 'USA', '9876543210'),
('22002', 'SEL00002', '456_Blue_Ave', 'Los_Angeles', 90001, 'USA', '7771234567'),

-- Addresses for customers
('33001', 'CUS00001', '123_Main_St', 'New_York', 10001, 'USA', '5551234567'),
('33002', 'CUS00002', '456_Oak_Ave', 'Los_Angeles', 90001, 'USA', '6666543210');
```

### 6.2 Data Manipulation

Manipulation of data is the process of manipulating or changing information to make it more organized and readable. DML(Data Manipulation Language) is used to accomplish this. DML is capable of adding, removing and altering databases.

SQL, a data manipulation language, is used to communicate with DBMS(Database Management System). SQL uses these DML commands for data manipulation.

- **Select:** This command selects a section of the database (a few rows/columns) to work on.
- **UPDATE:** This command is required to make changes to the existing data.
- **INSERT:** This command is used to insert data in a different location or move data.
- **DELETE:** This command is used to delete the redundant or duplicate data from the table.

We've used Data Manipulation for our project's business operations and optimization. It is the key feature to deal with the data in the way that needs it to use data properly and turn it into valuable information such as analyzing financial data, consumer behavior, and doing trend analysis.

Our data manipulations can be further observed in the query section.

## **7 Query Implementation**

### **7.1 User Queries**

1. User searches for a product
2. User searches for products in a price range
3. Display best-selling products
4. Recommend products of the same category from user's wishlist
5. Identify repeating customers
6. Cancel an order
7. Show the seller which product is being ordered the most
8. Show orders that were placed in April of each year sorted in descending order of total amount spent for each customer
9. Calculate the total revenue of each product of a seller for a given period
10. Show which product is under 1500 taka after a discount
11. User updates price and stock quantity of a product
12. Show all items in a users cart
13. Get the list of reviews for a specific product
14. Display all items in a user's wishlist
15. Retrieve the total spending of each user on orders
16. Update the address of a specific user
17. Apply a discount to all products in a specific category
18. Update the payment status for a specific order
19. Update the offer discount for a specific product
20. Remove an item from a user's cart
21. Users search for a particular product's review and sellers
22. Users search for a particular vendor's lowest price product
23. User search for a product showing from different sellers and order by price
24. Retrieve all products with their category names

25. Find all orders made by a specific user
26. Find all active offers on products
27. Retrieve the top 10 products with the highest sales
28. List products with their average rating and number of reviews
29. Find products that have not been sold
30. Find the most popular product categories based on the number of sales

## 7.2 SQL Queries

1. User searches for a product

```
SELECT * FROM product
WHERE name LIKE '%smartphone%'
OR description LIKE '%smartphone%';
```

*Output:*

id	name	user_id	description	price	stk_quantity	image_path	category_id	created_at	modified_at
PR000001	Smartphone X	SEL00001	High-performance smartphone with advanced f...	799.99	100	phone1.jpg	PC0001	2020-04-13 08:00:00	2020-04-13 08:00:00
PR000042	Portable Charger	SEL000006	Compact portable charger for smartphones	29.99	200	charger1.jpg	PC0003	2020-04-17 10:30:00	2020-04-17 10:30:00

2. User searches for products in a price range

```
SELECT *
FROM product
WHERE price BETWEEN 10 AND 50;
```

*Output:*

id	name	user_id	description	price	stk_quantity	image_path	category_id	created_at	modified_at	total_orders
PR000001	Smartphone X	SEL00001	High-performance smartphone with advanced f...	799.99	100	phone1.jpg	PC0001	2020-04-13 08:00:00	2020-04-13 08:00:00	1
PR000002	Laptop Pro	SEL00002	Powerful laptop for professional use	1499.99	50	laptop1.jpg	PC0001	2020-04-13 10:30:00	2020-04-13 10:30:00	1
PR000003	Smart TV 4K	SEL00003	Ultra HD smart TV with streaming capabilities	899.99	30	tv1.jpg	PC0001	2020-04-13 12:15:00	2020-04-13 12:15:00	1
PR000004	Wireless Headphones	SEL00004	Bluetooth headphones with noise cancellation	199.99	100	headphones1.jpg	PC0001	2020-04-13 14:45:00	2020-04-13 14:45:00	1
PR000005	Men's Shirt	SEL00001	Casual shirt for men	29.99	200	shirt1.jpg	PC0002	2020-04-13 16:20:00	2020-04-13 16:20:00	1
PR000006	Women's Dress	SEL00002	Elegant dress for women	49.99	150	dress1.jpg	PC0002	2020-04-13 09:00:00	2020-04-13 09:00:00	1
PR000007	Kitchen Blender	SEL00003	High-speed blender for kitchen use	79.99	80	blender1.jpg	PC0003	2020-04-13 11:40:00	2020-04-13 11:40:00	1
PR000008	Cookware Set	SEL00004	Complete set of non-stick cookware	149.99	50	cookware1.jpg	PC0003	2020-04-13 13:30:00	2020-04-13 13:30:00	1
PR000009	Novel: "The Adventure"	SEL00001	Bestselling adventure novel	12.99	300	novel1.jpg	PC0004	2020-04-13 15:10:00	2020-04-13 15:10:00	1
PR000010	Educational Toy Set	SEL00001	Toy set for learning and fun	39.99	100	toy1.jpg	PC0005	2020-04-13 17:20:00	2020-04-13 17:20:00	1

3. Display best-selling products

```
SELECT p.*, COUNT(oi.product_id) AS total_orders
FROM product p
LEFT JOIN order_item oi
ON p.id = oi.product_id
GROUP BY p.id
ORDER BY total_orders DESC;
```

Output:

Result Grid										
Filter Rows:										
Export: Wrap Cell Content:										
id	name	user_id	description	price	stk_quantity	image_path	category_id	created_at	modified_at	total_orders
PR000001	Smartphone X	SEL00001	High-performance smartphone with advanced f...	799.99	100	phone1.jpg	PC0001	2020-04-13 08:00:00	2020-04-13 08:00:00	1
PR000002	Laptop Pro	SEL00002	Powerful laptop for professional use	1499.99	50	laptop1.jpg	PC0001	2020-04-13 10:30:00	2020-04-13 10:30:00	1
PR000003	Smart TV 4K	SEL00003	Ultra HD smart TV with streaming capabilities	899.99	30	tv1.jpg	PC0001	2020-04-13 12:15:00	2020-04-13 12:15:00	1
PR000004	Wireless Headphones	SEL00004	Bluetooth headphones with noise cancellation	199.99	100	headphones1.jpg	PC0001	2020-04-13 14:45:00	2020-04-13 14:45:00	1
PR000005	Men's Shirt	SEL00001	Casual shirt for men	29.99	200	shirt1.jpg	PC0002	2020-04-13 16:20:00	2020-04-13 16:20:00	1
PR000006	Women's Dress	SEL00002	Elegant dress for women	49.99	150	dress1.jpg	PC0002	2020-04-13 09:00:00	2020-04-13 09:00:00	1
PR000007	Kitchen Blender	SEL00003	High-speed blender for kitchen use	79.99	80	blender1.jpg	PC0003	2020-04-13 11:40:00	2020-04-13 11:40:00	1
PR000008	Cookware Set	SEL00004	Complete set of non-stick cookware	149.99	50	cookware1.jpg	PC0003	2020-04-13 13:30:00	2020-04-13 13:30:00	1
PR000009	Novel: "The Adventure"	SEL00001	Bestselling adventure novel	12.99	300	novel1.jpg	PC0004	2020-04-13 15:10:00	2020-04-13 15:10:00	1
PR000010	Educational Toy Set	SEL00001	Toy set for learning and fun	30.00	100	toy1.jpg	PC0005	2020-04-13 17:00:00	2020-04-13 17:00:00	1

#### 4. Recommend products of the same category from user's wishlist

```
SELECT * FROM wishlist_item wi
JOIN wishlist w
  ON wi.wishlist_id = w.id
WHERE wi.wishlist_id = (
  SELECT w2.id FROM wishlist w2
  WHERE w2.user_id = 'CUS00001'
);
```

Output:

Result Grid									
Filter Rows:									
Export: Wrap Cell Content:									
	id	wishlist_id	product_id	quantity	created_at	modified_at	id	user_id	total_amt
▶	WLI00001	WL0001	PR000001	1	2024-04-13 08:00:00	2024-04-13 08:00:00	WL0001	CUS00001	150.00
	WLI00002	WL0001	PR000002	1	2024-04-13 10:30:00	2024-04-13 10:30:00	WL0001	CUS00001	150.00

#### 5. Identify repeating customers

```
SELECT user_id, COUNT(*) AS "Order_Count"
FROM 'order'
GROUP BY user_id
ORDER BY COUNT(*) DESC;
```

Output:

Result Grid		
Filter Rows:		
Export: Wrap Cell Content:		
	user_id	Order Count
▶	CUS00008	3
	CUS00001	3
	CUS00006	3
	CUS00004	3
	CUS00009	3
	CUS00002	3
	CUS00007	3
	CUS00005	2

#### 6. Cancel an order

```

UPDATE 'payment'
SET 'status' = FALSE
WHERE 'order_id' = 'ORD0001';

SELECT * FROM payment;

```

Output:

Result Grid						
Filter Rows: <input type="text"/>						
Edit:    Export/Import:   Wrap Cell C						
	id	order_id	account_no	payment_type	status	created_at
▶	PAY00001	ORD0001	ACC00001	Credit Card	0	2020-04-13 08:45:00
	PAY00002	ORD0002	ACC00002	Debit Card	1	2020-04-13 11:00:00
	PAY00003	ORD0003	ACC00003	Cash	1	2020-04-13 13:00:00
	PAY00004	ORD0004	ACC00004	Credit Card	1	2020-04-13 15:30:00
	PAY00005	ORD0005	ACC00005	Cash	1	2020-04-13 18:00:00

7. Show the seller which product is being ordered the most

```

SELECT user_id AS "Seller", COUNT(*) AS "Order_Count"
FROM order_item oi
JOIN product p ON oi.product_id = p.id
GROUP BY p.user_id
ORDER BY "Order_Count" DESC;

```

Output:

Result Grid						
Filter Rows: <input type="text"/>						
Edit:    Export/Import:   Wrap Cell C						
	id	order_id	account_no	payment_type	status	created_at
▶	PAY00001	ORD0001	ACC00001	Credit Card	0	2020-04-13 08:45:00
	PAY00002	ORD0002	ACC00002	Debit Card	1	2020-04-13 11:00:00
	PAY00003	ORD0003	ACC00003	Cash	1	2020-04-13 13:00:00
	PAY00004	ORD0004	ACC00004	Credit Card	1	2020-04-13 15:30:00
	PAY00005	ORD0005	ACC00005	Cash	1	2020-04-13 18:00:00

8. Show orders that were placed in April of each year sorted in descending order of total amount spent for each customer

```

SELECT *, SUM(total_amt) FROM 'order'
WHERE created_at like "____-04-%"
GROUP BY user_id
ORDER BY SUM(total_amt) desc;

```

Output:

Result Grid						
Filter Rows: <input type="text"/>						
Export:  Wrap Cell Content:						
	id	user_id	total_amt	created_at	modified_at	SUM(total_amt)
▶	ORD0007	CUS00007	700.00	2020-04-13 11:00:00	2020-04-13 11:00:00	1450.00
	ORD0001	CUS00001	500.00	2020-04-13 08:30:00	2020-04-13 08:30:00	1250.00
	ORD0004	CUS00004	400.00	2020-04-13 14:15:00	2020-04-13 14:15:00	1100.00
	ORD0009	CUS00009	250.00	2020-04-13 15:30:00	2020-04-13 15:30:00	1050.00
	ORD0005	CUS00005	300.00	2020-04-13 16:30:00	2020-04-13 16:30:00	950.00
	ORD0002	CUS00002	200.00	2020-04-13 10:45:00	2020-04-13 10:45:00	700.00
	ORD0003	CUS00003	150.00	2020-04-13 12:30:00	2020-04-13 12:30:00	650.00

9. Calculate the total revenue of each product of a seller for a given period

```
SELECT p.user_id AS "Seller",
       oi.product_id,
       (oi.quantity * p.price) AS "Revenue",
       (Price - Price * of1.disc_pct/100 * of1.active) AS "Revenue_After_Discount"
FROM order_item oi
JOIN product p ON oi.product_id = p.id
JOIN payment pay ON oi.order_id = pay.order_id
JOIN offer of1 ON oi.product_id = of1.product_id
WHERE p.user_id = 'SEL00003' AND
       pay.status = true AND
       DATE(pay.created_at) >= "2020-04-10" AND DATE(pay.created_at) <= "2020-04-15"
GROUP BY oi.product_id;
```

Output:

Result Grid   Filter Rows:   Export:   Wrap Cell Content:				
	Seller	product_id	Revenue	Revenue After Discount
▶	SEL00003	PR000003	899.99	719.99200000
	SEL00003	PR000007	79.99	67.99150000

10. Show which product is under 1500 taka after a discount

```
SELECT pd.id, pd.price, pd.discounted_price
FROM (
    SELECT p.id, p.price, p.price - (p.price * of1.disc_pct/100 * of1.active) AS "discounted_price"
    FROM product p
    JOIN offer of1 ON p.id = of1.product_id
) AS pd
WHERE pd.discounted_price < 1500
ORDER BY pd.discounted_price DESC;
```

Output:

Result Grid   Filter Rows:   Export:   Wrap Cell Content:			
	id	price	discounted_price
▶	PR000002	1499.99	1274.99150000
	PR000003	899.99	719.99200000
	PR000001	799.99	719.99100000
	PR000004	199.99	149.99250000
	PR000008	149.99	119.99200000
	PR000007	79.99	67.99150000
	PR000006	49.99	44.99100000
	PR000010	39.99	27.99300000
	PR000005	29.99	20.99300000
	PR000009	19.99	17.99150000

## 11. User updates price and stock quantity of a product

```

UPDATE product
SET price = 999.99,
    stk_quantity = 100,
    modified_at = NOW()
WHERE id="PR000007" and user_id="SEL00003";

```

## 12. Show all items in a users cart

```

SELECT ci.id, p.name, ci.quantity, p.price, (ci.quantity * p.price) AS total_price
FROM cart_item ci
JOIN cart c ON ci.cart_id = c.id
JOIN product p ON ci.product_id = p.id
WHERE c.user_id = 'CUS00002'
ORDER BY ci.id;

```

*Output:*

Result Grid     Filter Rows: <input type="text"/>   Export:    Wrap Cell Content:					
	id	name	quantity	price	total_price
▶	CI0003	Smart TV 4K	1	899.99	899.99
	CI0018	Desktop Computer	1	1299.99	1299.99
	CI0019	Gardening Tool Set	3	69.99	209.97

## 13. Get the list of reviews for a specific product

```

SELECT r.id, r.product_id, u.first_name, u.last_name, r.rating, r.reacts
FROM review r
JOIN user u ON r.user_id = u.id
WHERE r.product_id = 'PR000009'
ORDER BY r.rating DESC;

```

*Output:*

Result Grid     Filter Rows: <input type="text"/>   Export:    Wrap Cell Content:						
	id	product_id	first_name	last_name	rating	reacts
▶	REV00009	PR000009	Ava	Bennett	4.9	22

## 14. Display all items in a user's wishlist

```

SELECT wi.id, p.name, p.price
FROM wishlist_item wi
JOIN product p ON wi.product_id = p.id
WHERE wi.wishlist_id = (
    SELECT w.id FROM wishlist w WHERE w.user_id = 'CUS00001'
)
ORDER BY wi.id;

```

Output:

The screenshot shows a database query result grid with the following columns: id, name, and price. The results are as follows:

	id	name	price
▶	WLI00001	Smartphone X	799.99
	WLI00002	Laptop Pro	1499.99

15. Retrieve the total spending of each user on orders

```

SELECT u.id, u.first_name, u.last_name, SUM(o.total_amt) AS total_spending
FROM 'user' u
JOIN 'order' o ON u.id = o.user_id
GROUP BY u.id, u.first_name, u.last_name
ORDER BY total_spending DESC;

```

Output:

The screenshot shows a database query result grid with the following columns: id, first\_name, last\_name, and total\_spending. The results are as follows:

	id	first_name	last_name	total_spending
▶	CUS00007	Isabella	Reyes	1450.00
	CUS00001	Michael	Johnson	1250.00
	CUS00004	Lucas	Nguyen	1100.00
	CUS00009	Ava	Bennett	1050.00
	CUS00005	Olivia	Hill	950.00
	CUS00002	Matthew	Hernandez	700.00
	CUS00003	Sophia	Lee	650.00
	CUS00006	Ethan	Morgan	600.00
	CUS00010	Logan	Carter	560.00
	CUS00008	Mason	Collins	500.00



16. Update the address of a specific user

```
UPDATE user_address
SET address = '123_New_Street', city = 'New_City', postal_code = '12345', country = 'New_Coun
WHERE user_id = 'CUS00001';
```

17. Apply a discount to all products in a specific category

```
UPDATE product
SET price = price * 0.9
WHERE category_id = 'PC0001';
```

18. Update the payment status for a specific order

```
UPDATE payment
SET status = 1
WHERE order_id = 'ORD0002' AND account_no = 'ACC00002';
```

19. Update the offer discount for a specific product

```
UPDATE offer
SET disc_pct = 20
WHERE product_id = 'PR000001';
```

20. Remove an item from a user's cart

```
DELETE FROM cart_item
WHERE cart_id = (
    SELECT id FROM cart WHERE user_id = 'CUS00003' LIMIT 1
) AND product_id = 'PR000004';
```

21. Users search for a particular product's review and sellers

```
select p.name,u.first_name,r.product_id,r.reacts,r.rating
from review r ,product p ,user u
WHERE p.name like "Backpack" AND u.user_type like "Seller"
order by rating desc;
```

*Output:*

Result Grid						Filter Rows:	Search	Export:
name	first_name	product...	reacts	rating				
Backpack	Alice	PR000009	22	4.9				
Backpack	Sophia	PR000009	22	4.9				
Backpack	Isabella	PR000009	22	4.9				
Backpack	Henry	PR000009	22	4.9				
Backpack	Grace	PR000009	22	4.9				
Backpack	Logan	PR000009	22	4.9				
Backpack	Madison	PR000009	22	4.9				
Backpack	Jack	PR000009	22	4.9				
Backpack	Chloe	PR000009	22	4.9				
Backpack	Ryan	PR000009	22	4.9				
Backpack	Lily	PR000009	22	4.9				
Backpack	Emily	PR000009	22	4.9				
Backpack	Mason	PR000009	22	4.9				
Backpack	Avery	PR000009	22	4.9				
Backpack	Evelyn	PR000009	22	4.9				
Backpack	Alice	PR000003	20	4.8				
Backpack	Sophia	PR000003	20	4.8				
Backpack	Isabella	PR000003	20	4.8				
Backpack	Henry	PR000003	20	4.8				
Backpack	Grace	PR000003	20	4.8				
Backpack	Logan	PR000003	20	4.8				
Backpack	Madison	PR000003	20	4.8				

Result 11

Read Only

22. Users search for a particular vendor's lowest price product.

```
select p.name, u.first_name , p.price
from product p , user u
where u.user_type like "Seller" and u.first_name like "Alice" and p.price =(select
min(x.price)
from (select p.name, u.first_name , p.price
from product p , user u
where u.user_type like "Seller" and u.first_name like "Alice" ) x);
```

Output:

name	first_name	price
Novel: "The Adventure"	Alice	12.99

23. User search for a product showing from different sellers and order by price

```
select p.name, u.first_name , p.price
from product p , user u
where p.name like "Laptop_Pro"
order by p.price desc;
```

Output:

name	first_name	price
Laptop Pro	Michael	1499.99
Laptop Pro	Matthew	1499.99
Laptop Pro	Sophia	1499.99
Laptop Pro	Lucas	1499.99
Laptop Pro	Olivia	1499.99
Laptop Pro	Ethan	1499.99
Laptop Pro	Isabella	1499.99
Laptop Pro	Mason	1499.99
Laptop Pro	Ava	1499.99

24. Retrieve all products with their category names

```
SELECT p.id AS product_id ,p.name AS Product_name ,c.name AS Category_name ,
p.price ,p.stk_quantity
FROM product p
JOIN product_category c ON p.category_id = c.id;
```

Output:

product_...	product_name	category_name	price	stk_quanti...	
PR000001	Smartphone X	Electronics	799.99	100	
PR000002	Laptop Pro	Electronics	1499.99	50	
PR000003	Smart TV 4K	Electronics	899.99	30	
PR000004	Wireless Headphones	Electronics	199.99	100	
PR000011	Gaming Console	Electronics	399.99	80	
PR000014	Digital Camera	Electronics	499.99	40	
PR000018	Desktop Computer	Electronics	1299.99	30	
PR000021	Wireless Mouse	Electronics	19.99	200	
PR000024	Bluetooth Speaker	Electronics	59.99	100	
PR000035	Wireless Earbuds	Electronics	89.99	80	
PR000041	Bluetooth Headset	Electronics	69.99	100	
PR000005	Men's Shirt	Clothing	29.99	200	
PR000006	Women's Dress	Clothing	49.99	150	
PR000007	Kitchen Blender	Home & Kitchen	79.99	80	
PR000008	Cookware Set	Home & Kitchen	149.99	50	
PR000013	Coffee Maker	Home & Kitchen	59.99	90	
PR000032	Stainless Steel Cook...	Home & Kitchen	199.99	50	
PR000042	Portable Charger	Home & Kitchen	29.99	200	
Result 2					

25. Find all orders made by a specific user

```
SELECT o.id AS order_id ,concat(u.first_name , "_" ,u.last_name) as name,
o.total_amt ,
o.created_at
FROM 'order' o
JOIN user u ON o.user_id = u.id
WHERE u.email = 'lucas@example.com';
```




Output:

order_id	name	total_amt	created_at	

26. Find all active offers on products

```
select o.id AS Offer_id , p.name AS Product_name,o.name AS
Offer_name, o.disc_pct , o.created_at
from offer o join product p ON o.product_id = p.id
where o.active = 1;
```

Output:

Result Grid   Filter Rows: <input type="text" value="Search"/> Export: 						
	Offer_id	Product_name	Offer_name	disc_pct	created_at	
	OF0001	Smartphone X	Spring Sale	10.00	2024-04-13 08:00:00	
	OF0002	Laptop Pro	Laptop Discount	15.00	2024-04-13 10:30:00	
	OF0003	Smart TV 4K	TV Clearance	20.00	2024-04-13 12:15:00	
	OF0004	Wireless Headphones	Headphones Flash Sale	25.00	2024-04-13 14:45:00	
	OF0005	Men's Shirt	Shirt Promotion	30.00	2024-04-13 16:20:00	
	OF0006	Women's Dress	Dress Sale	10.00	2024-04-13 09:00:00	
	OF0007	Kitchen Blender	Kitchen Appliance Discount	15.00	2024-04-13 11:40:00	
	OF0008	Cookware Set	Cookware Set Offer	20.00	2024-04-13 13:30:00	
	OF0009	Novel: "The Adventure"	Book Clearance	25.00	2024-04-13 15:10:00	
	OF0010	Educational Toy Set	Toy Set Discount	30.00	2024-04-13 17:20:00	
Result 9						

27. Retrieve the top 10 products with the highest sales

```
select p.id AS product_id ,p.name AS product_name ,
sum(oi.quantity) AS Total_Quantity_Sold ,
sum(oi.quantity * p.price) AS Total_Sales
from order_item oi join product p
on oi.product_id = p.id
group by p.id, p.name
order by total_quantity_sold DESC LIMIT 10;
```

Output:

	product_...	product_name	total_quantity_s...	total_sales	
	PR000005	Men's Shirt	5	149.95	
	PR000030	Resistance Bands	3	59.97	
	PR000020	Wristwatch	3	299.97	
	PR000004	Wireless Headphones	3	599.97	
	PR000014	Digital Camera	3	1499.97	
	PR000009	Novel: "The Adventure"	3	38.97	
	PR000026	Camping Tent	3	449.97	
	PR000038	Yoga Blocks	3	44.97	
	PR000016	Baby Stroller	2	299.98	
	PR000006	Women's Dress	2	99.98	




28. List products with their average rating and number of reviews

```

select p.id AS Product_Id, p.name AS Product_Name,
avg(r.rating) AS Average_Rating,
count(r.id) AS Number_of_Reviews
from product p LEFT JOIN review r ON p.id = r.product_id
group by p.id, p.name
order by average_rating DESC;

```

Output:

Result Grid   Filter Rows: <input type="text" value="Search"/> Export: 					
	product_...	product_name	average_rating	number_of_reviews	
	PR000009	Novel: "The Adventure"	4.90000	1	
	PR000003	Smart TV 4K	4.80000	1	
	PR000005	Men's Shirt	4.70000	1	
	PR000014	Digital Camera	4.70000	1	
	PR000007	Kitchen Blender	4.60000	1	
	PR000011	Gaming Console	4.60000	1	
	PR000001	Smartphone X	4.50000	1	
	PR000013	Coffee Maker	4.40000	1	
	PR000008	Cookware Set	4.30000	1	
	PR000004	Wireless Headphones	4.20000	1	
	PR000012	Running Shoes	4.10000	1	
	PR000002	Laptop Pro	4.00000	1	
	PR000010	Educational Toy Set	3.80000	1	
	PR000015	Children's Book Set	3.70000	1	
	PR000006	Women's Dress	3.50000	1	
	PR000016	Baby Stroller	NULL	0	
	PR000017	Yoga Mat	NULL	0	

29. Find products that have not been sold

```

select p.id, p.name
from product p left join order_item oi




```

```

on p.id = oi.product_id
left join order o
on oi.order_id = o.id
where o.created_at IS NULL;

```

Output:

Result Grid   Filter Rows: <input type="text" value="Search"/> Export: 			
	id	name	
	PR000041	Bluetooth Headset	
	PR000042	Portable Charger	
	PR000043	Leather Wallet	
	PR000044	Fitness Gloves	
	PR000045	Cycling Helmet	
	PR000046	Wireless Security Alarm	
	PR000047	Travel Backpack	
	PR000048	Gaming Keyboard	
	PR000049	Digital Drawing Tablet	

30. Find the most popular product categories based on the number of sales



```

select c.id AS Category_Id, c.name AS Category_Name,
count(oi.id) AS total_sales
from product_category c join product p ON c.id = p.category_id
join order_item oi ON p.id = oi.product_id
group by c.id, c.name
order by total_sales DESC;

```

Output:



Result Grid   Filter Rows: <input type="text" value="Search"/> Export: 				
	category_id	category_name	total_sales	
	PC0001	Electronics	10	
	PC0003	Home & Kitchen	4	
	PC0019	Crafts & DIY	3	
	PC0002	Clothing	2	
	PC0004	Books	2	
	PC0006	Beauty & Personal Care	2	
	PC0007	Sports & Outdoors	2	
	PC0008	Automotive	2	
	PC0011	Furniture	2	
	PC0017	Musical Instruments	2	
	PC0018	Baby & Kids	2	
	PC0005	Toys & Games	1	
	PC0009	Health & Wellness	1	
	PC0010	Grocery	1	
	PC0012	Jewelry	1	
	PC0014	Garden & Outdoor	1	
	PC0016	Office Supplies	1	
	PC0020	Fitness & Exercise	1	

## 8 Database Testing

- Structural Testing

- Schema Verification: Schema can be verified through this command.

```
describe product_category;
```

*Output:*

	Field	Type	Null	Key	Default	Extra
▶	id	varchar(20)	NO	PRI	NULL	
	name	varchar(30)	NO		NULL	
	description	text	NO		NULL	
	created_at	datetime	NO		NULL	
	modified_at	datetime	YES		NULL	

- Indexing Verification:

- Functional Testing: CRUD operation can performed properly.

- \* Create

```
CREATE TABLE 'product_category' (
  'id' varchar (20) not null,
  'name' varchar (30) not null,
  'description' text not null,
```

```

        'created_at' datetime not null,
        'modified_at' datetime,
        primary key ('id')
    );

```

\* Read

```
SELECT * FROM product_category;
```

Output:

id	name	description	created_at	modified_at
PC0001	Electronics	Electronics and gadgets	2016-04-13 08:00:00	NULL
PC0002	Clothing	Clothing and apparel	2017-04-13 10:30:00	NULL
PC0003	Home & Kitchen	Home appliances and kitchenware	2018-04-13 12:15:00	NULL
PC0004	Books	Books and literature	2019-04-13 14:45:00	NULL
PC0005	Toys & Games	Toys and games for all ages	2020-04-13 16:20:00	NULL
PC0006	Beauty & Personal Care	Cosmetics and personal care products	2021-04-13 09:00:00	NULL
PC0007	Sports & Outdoors	Sports equipment and outdoor gear	2022-04-13 11:40:00	NULL

\* Update

```

UPDATE 'payment'
SET 'status' = FALSE
WHERE 'order_id' = 'ORD0001';

SELECT * FROM payment;

```

Output:

id	order_id	account_no	payment_type	status	created_at
PAY00001	ORD0001	ACC00001	Credit Card	0	2020-04-13 08:45:00
PAY00002	ORD0002	ACC00002	Debit Card	1	2020-04-13 11:00:00
PAY00003	ORD0003	ACC00003	Cash	1	2020-04-13 13:00:00
PAY00004	ORD0004	ACC00004	Credit Card	1	2020-04-13 15:30:00
PAY00005	ORD0005	ACC00005	Cash	1	2020-04-13 18:00:00

\* Delete

```

DELETE FROM cart_item
WHERE cart_id = (
    SELECT id FROM cart WHERE user_id = 'CUS00003' LIMIT 1
) AND product_id = 'PR000004';

```

Output:

id	cart_id	product_id	quantity	created_at	modified_at
CI0001	CRT0001	PR000001	2	2020-04-13 08:30:00	2020-04-13 08:30:00
CI0002	CRT0001	PR000002	1	2020-04-13 08:30:00	2020-04-13 08:30:00
CI0003	CRT0002	PR000003	1	2020-04-13 10:45:00	2020-04-13 10:45:00
CI0005	CRT0004	PR000005	2	2020-04-13 14:15:00	2020-04-13 14:15:00
CI0006	CRT0005	PR000006	1	2020-04-13 16:30:00	2020-04-13 16:30:00
CI0007	CRT0006	PR000007	2	2020-04-13 18:45:00	2020-04-13 18:45:00

– Key Verification:

\* Throws error if same id is inserted again.

```

-- Trying to insert already existing id
INSERT INTO 'product_category' ('id', 'name', 'description', 'created_at')
VALUES
('PC0001', 'Electronics', 'Electronics_and_gadgets', '2016-04-13_08:00:00');

```



Output:

```
Error Code: 1062. Duplicate entry 'PC0001' for key 'PRIMARY'
```

- Security Testing

If there's lack of proper permission, the queries won't be executed. To check, execute queries without permission. If it's not executed, the database is secure. Example query:

```
DELETE FROM cart_item
WHERE cart_id = (
SELECT id FROM cart WHERE user_id = 'CUS00003' LIMIT 1
) AND product_id = 'PR000004';
```

Output:

The query won't be executed without permission.

- Data Integrity

It ensures that data remains unchanged during operations such as transfer, storage, and retrieval. Data is protected from corruption or unauthorized modifications. Data integrity is crucial for maintaining trustworthiness and usability in any database system.

## 9 Conclusion

Our E-Commerce database project is designed to meet a client's all operational and searching need from an online store database .It is built on several interrelated tables such as users, orders, payments, reviews , wish list's etc. for handling a client's online store. It is basically a comprehensive solution designed to manage the composite operations of an online store. Our designed schema make sure data integrity and allows advanced analytical analysis of the database to drive the client's business to further improvement .At the end it also leaves us with areas we can improve further more to make a much more complicated database while omnimart database works as a base to our self.

## 10 Acknowledgements

We, the team, would like to express our sincerest gratitude towards everyone who contributed to the project.

First and foremost, we express our gratitude towards **Kamruddin Nur**, our faculty and supervisor, for his guidance, support, and insightful feedback. Your direction has been crucial in achieving our goals.

We also would like to express our heartfelt apprecaation towards our fellow group member: Maharun Afroz, Fuwad Hasan and, Al-Amin Rabbi for their dedication and teamwork.