

**Team Members**

**Ali Akbar** – BSE 233016

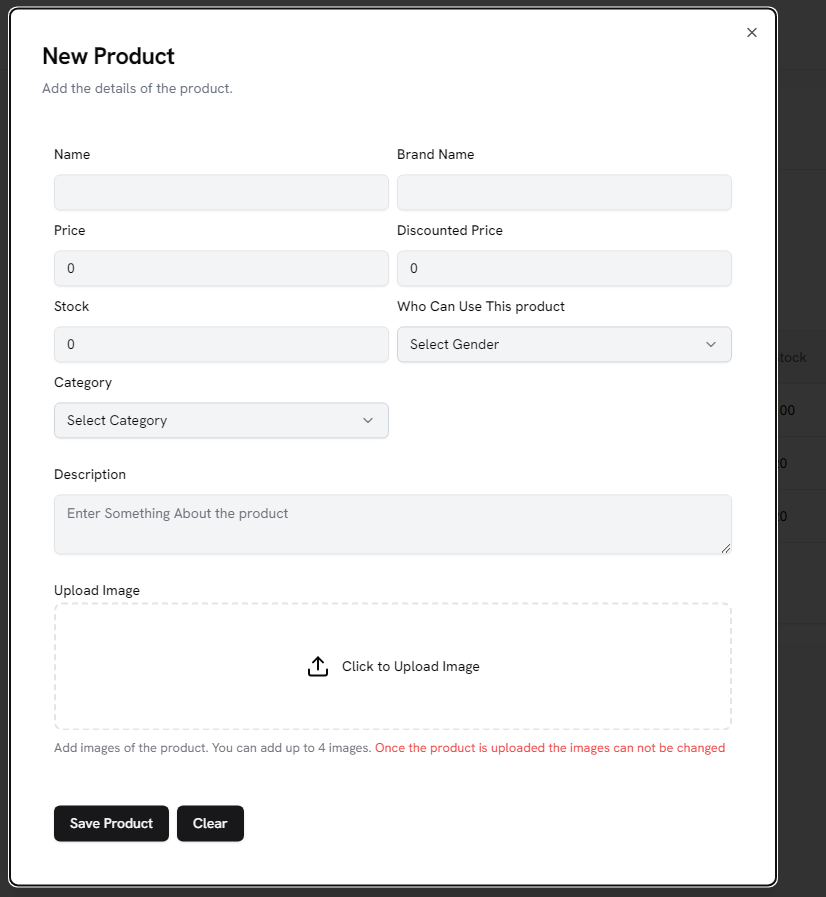
**Laiba Zulfiqar** – BSE233022

**Zainab Fatima** – BSE233008

***Git hub URL*** = https://github.com/ali-akbar-019/db-project.git

**Testing**

Using this UI to add products:



This is the code used to add the product, using prisma ,sql , express and node js

const createProducts = async (req, res) => {

    try {

        const { images, ...productData } = req.body;

        if (!images || !Array.isArray(images) || images.length === 0) {

            return res.status(404).json({

                message: "Images not found or invalid format",

            });

        }

*// Create the product first*

        const product = await prisma.product.create({

            data: {

                ...productData,

            },

        });

*// Create images linked to the product*

        const imagePromises = images.map((img) =>

            prisma.image.create({

                data: {

                    url: img,

                    altText: img.altText || null,

                    productId: product.id,

                },

            })

        );

        const allImages = await Promise.all(imagePromises);

        return res.status(200).json({

            message: "Product created successfully",

            product,

            images: allImages,

        });

    } catch (error) {

        console.error("Error while creating the product:", error.message);

        return res.status(500).json({

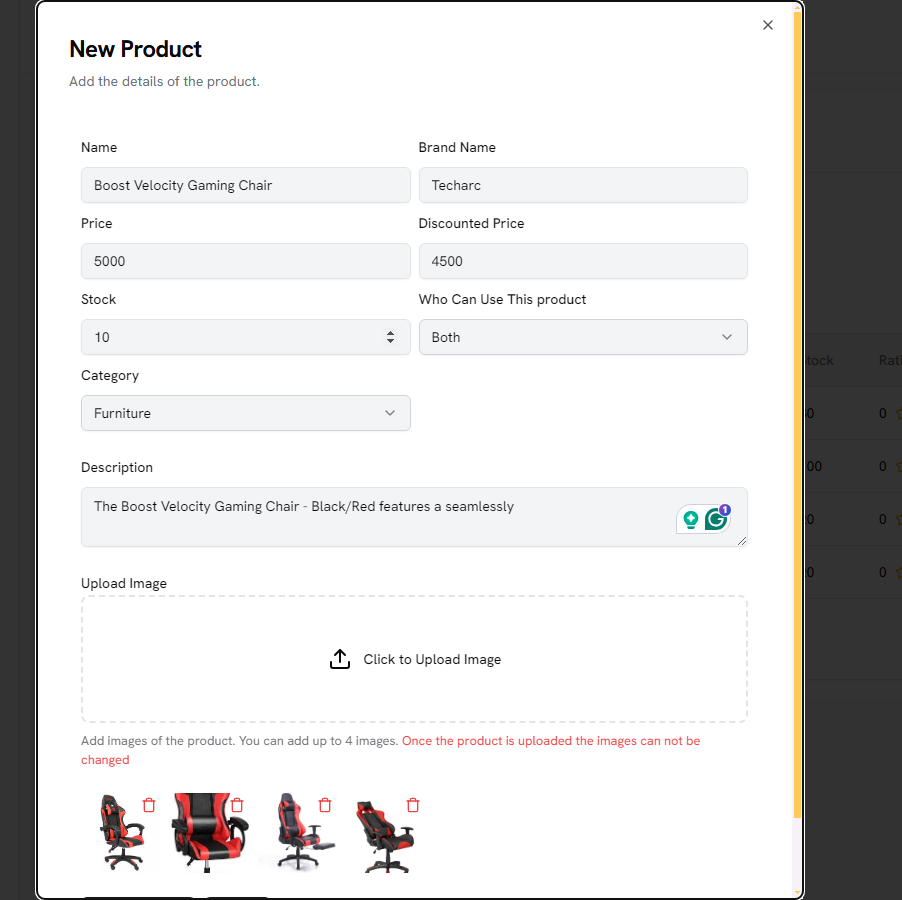
            message: "Error while creating the product",

        });

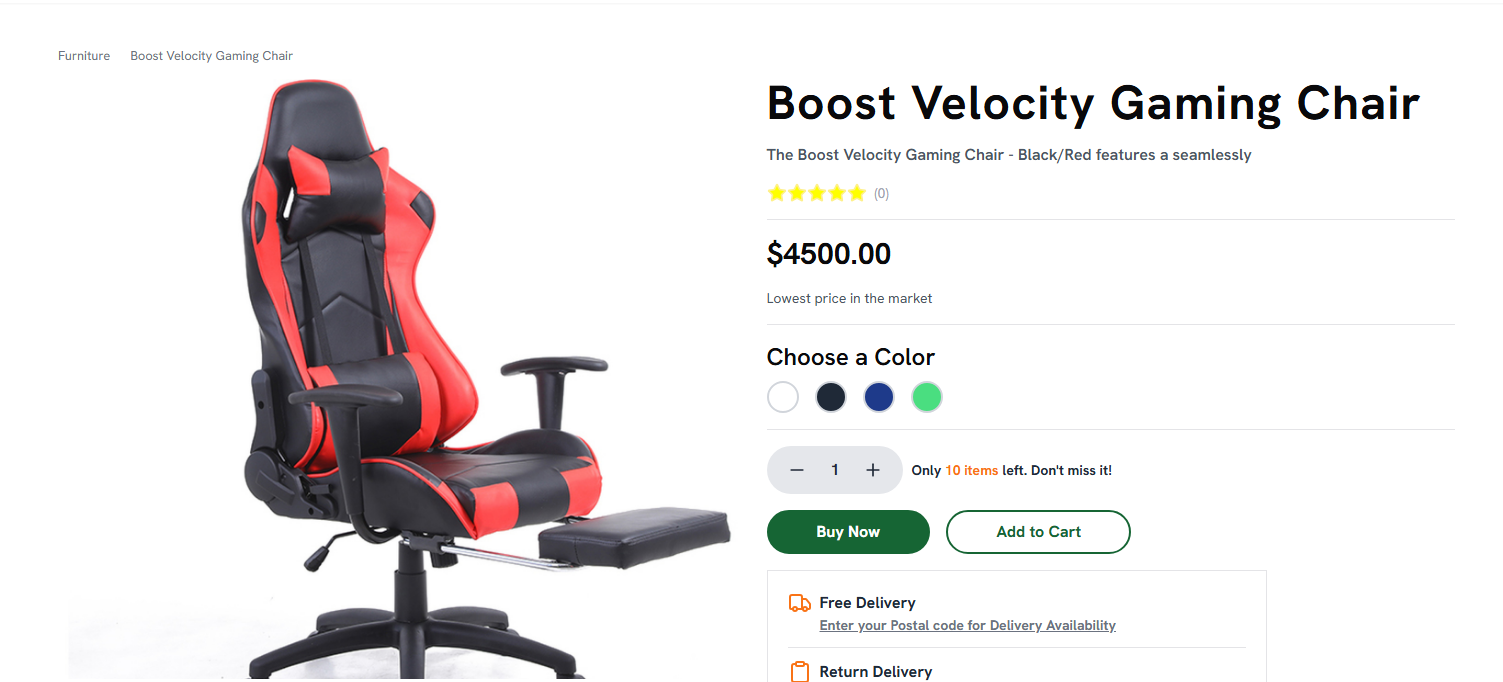
    }

};

**Lets Add A Gaming Chair**



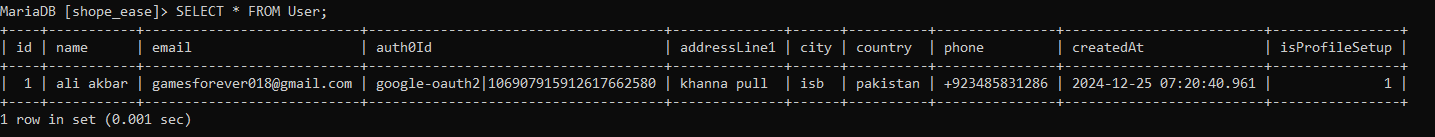
**Added this**



**SQL Queries:**

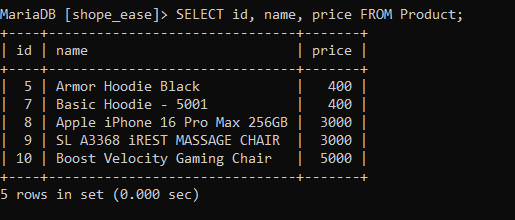
1. **Select all users**

SELECT \* FROM User;



1. **Select specific columns from the Product table**

SELECT id, name, price FROM Product;



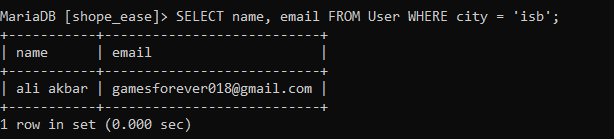
1. **Select products where stock is less than 10**

**Select name, stock from product where stock < 10;**



1. **Select users from a specific city**

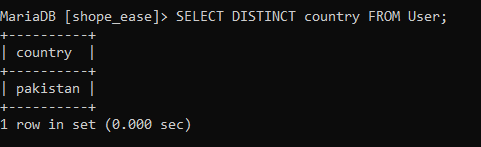
SELECT name, email FROM User WHERE city = 'isb';



**Projection Queries**

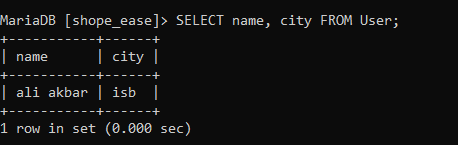
1. **Select distinct countries of users**

SELECT DISTINCT country FROM User;



1. **Select names of users and their cities**

SELECT name, city FROM User;



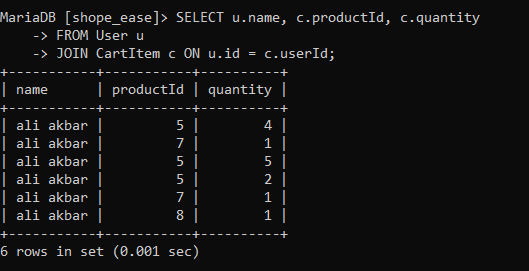
**Joining Queries**

1. **Join User and CartItem to fetch user cart items**

SELECT u.name, c.productId, c.quantity

FROM User u

JOIN CartItem c ON u.id = c.userId;

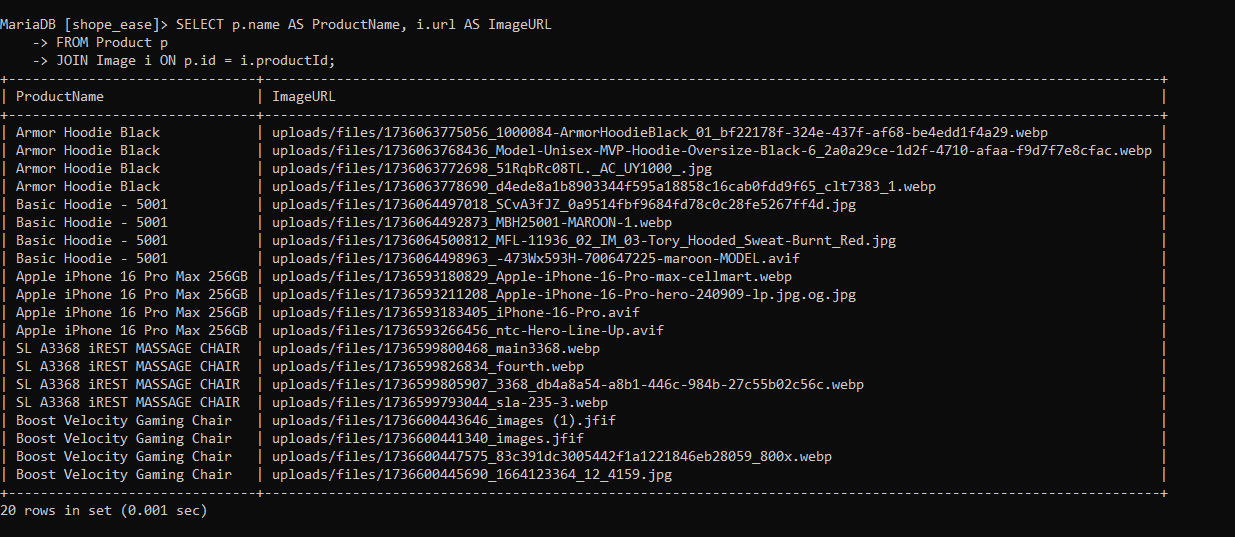


1. **Join Product and Image to fetch product images**

SELECT p.name AS ProductName, i.url AS ImageURL

FROM Product p

JOIN Image i ON p.id = i.productId;



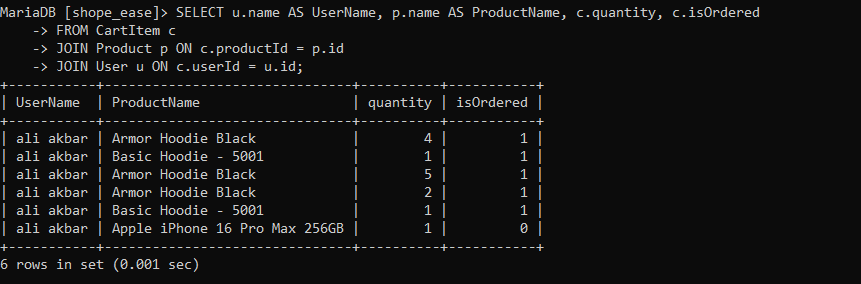
1. **Join CartItem, Product, and User to fetch cart details**

SELECT u.name AS UserName, p.name AS ProductName, c.quantity, c.isOrdered

FROM CartItem c

JOIN Product p ON c.productId = p.id

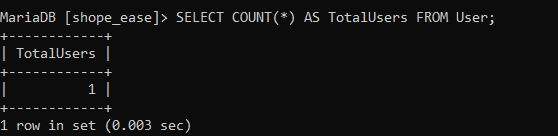
JOIN User u ON c.userId = u.id;



**Aggregation Queries**

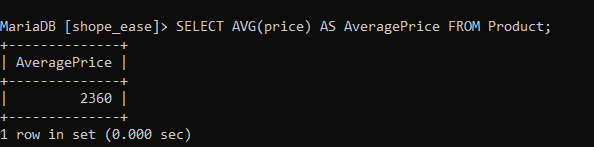
1. **Count the total number of users**

SELECT COUNT(\*) AS TotalUsers FROM User;



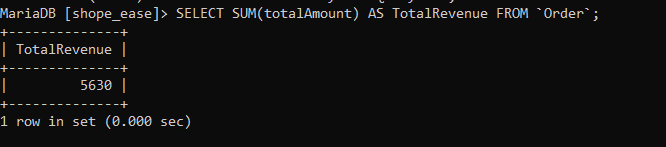
1. **Find the average price of products**

SELECT AVG(price) AS AveragePrice FROM Product;



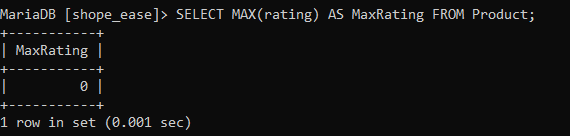
1. **Find the total revenue from all orders**

SELECT SUM(totalAmount) AS TotalRevenue FROM `Order`;



1. **Find the maximum rating of products**

SELECT MAX(rating) AS MaxRating FROM Product;

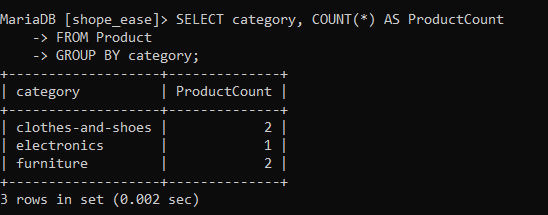


1. **Find the number of products in each category**

SELECT category, COUNT(\*) AS ProductCount

FROM Product

GROUP BY category;



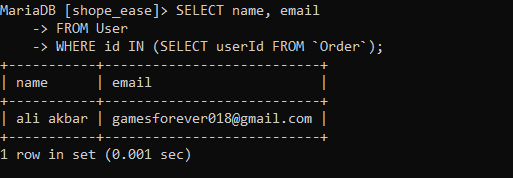
**Subqueries**

1. **Find users who have placed an order**

SELECT name, email

FROM User

WHERE id IN (SELECT userId FROM `Order`);

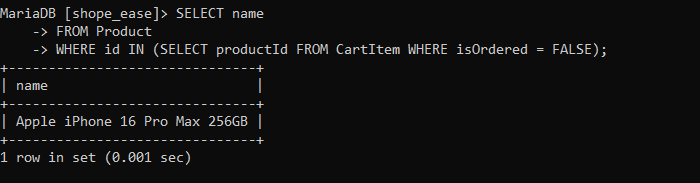


1. **Find products that are in a user's cart but not yet ordered**

SELECT name

FROM Product

WHERE id IN (SELECT productId FROM CartItem WHERE isOrdered = FALSE);

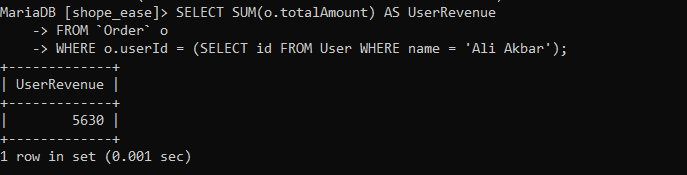


1. **Find the total revenue of a specific user**

SELECT SUM(o.totalAmount) AS UserRevenue

FROM `Order` o

WHERE o.userId = (SELECT id FROM User WHERE name = 'Ali Akbar');



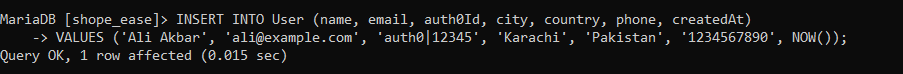
**CRUD Operations**

***Create:***

1. **Insert a new user**

INSERT INTO User (name, email, auth0Id, city, country, phone, createdAt)

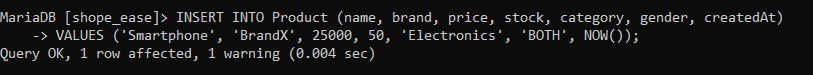
VALUES ('Ali Akbar', 'ali@example.com', 'auth0|12345', 'Karachi', 'Pakistan', '1234567890', NOW());



1. **Insert a new product**

INSERT INTO Product (name, brand, price, stock, category, gender, createdAt)

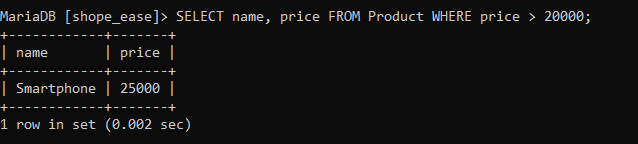
VALUES ('Smartphone', 'BrandX', 25000, 50, 'Electronics', 'BOTH', NOW());



***Read:***

1. **Retrieve products above a specific price**

SELECT name, price FROM Product WHERE price > 20000;



1. **Fetch all users who have more than one cart item**

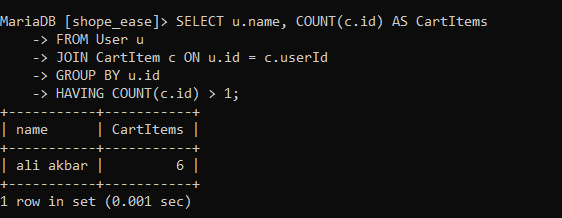
SELECT u.name, COUNT(c.id) AS CartItems

FROM User u

JOIN CartItem c ON u.id = c.userId

GROUP BY u.id

HAVING COUNT(c.id) > 1;



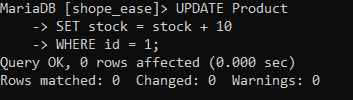
***Update:***

1. **Update stock of a specific product**

UPDATE Product

SET stock = stock + 10

WHERE id = 1;

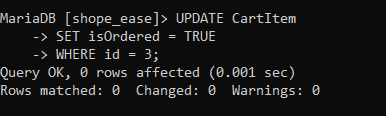


1. **Mark a cart item as ordered**

UPDATE CartItem

SET isOrdered = TRUE

WHERE id = 3;



***Delete:***

1. **Delete User**

DELETE FROM User WHERE id = 5;



1. **Delete Product**

Delete from product where id = 100;

