**ACKNOWLEDGEMENT**

We would like to express our heartfelt gratitude and appreciation to all those who have contributed to the successful completion of our supermarket management system project. First and foremost, we would like to extend our sincere thanks to our instructor, Dr. Hargeet Kaur, for her continuous guidance, support, and encouragement throughout the project. Her valuable insights and feedback have played a crucial role in shaping our ideas and enhancing the overall quality of our work.

We are grateful to our team members, Harsh Shah, Vraj Patel, Tanish Suthar for their unwavering dedication, hard work, and exceptional teamwork in completing this project successfully. Each member has brought their unique skills and expertise to the table, allowing us to create a comprehensive and efficient supermarket management system that we are truly proud of.

Furthermore, we would like to acknowledge the resources and references that have been instrumental in enriching our understanding of the subject matter. Online tutorials, research papers, and other relevant sources have provided invaluable guidance and knowledge throughout the project.

In conclusion, we extend our heartfelt thanks to everyone who has supported and assisted us in completing this project. We are excited to have developed a supermarket management system that will streamline operations and improve the overall shopping experience for customers.

**SUPERMART MANAGEMENT SYSTEM**

**Objective:**

The Objective of our supermarket management system is to create an efficient database that streamlines inventory, sales, customer and employee management, and order processing. The project aims to improve and optimize supermarket operations, and enable better decision-making for management.

**Overview:**

The supermarket management system project is an innovative solution that simplifies the management of a supermarket's daily operations. By consolidating various aspects such as inventory, sales, orders, and staff information into a single system, it provides a cohesive platform for better organization and increased efficiency within the supermarket.

**Description:**

The supermarket management system is a relational database designed to store and manage all pertinent data for a supermarket, with a focus on facilitating effective communication and collaboration among employees and the owner. The database comprises several tables, including CUSTOMER, CATEGORY, PRODUCT, EMPLOYEE, ORDERS, ORDER\_ITEM, PAYMENT, SALES, EMPLOYEE\_MOBILE, and EMPLOYEE\_ORDER. Each table represents a specific aspect of the supermarket's operations and is linked through primary and foreign keys, ensuring data integrity and allowing for complex queries.

**DATABASE SCHEMA**

* **CUSTOMER** – Store’s customer information like name, mobile number, and a unique ID.

CREATE TABLE CUSTOMER (

C\_ID VARCHAR(10) PRIMARY KEY,

C\_NAME VARCHAR(40),

C\_MOBILE VARCHAR(10),

CONSTRAINT CHECK\_PHONE CHECK (C\_MOBILE REGEXP '^[0-9]{10}$')

);

* **CATEGORY** - Stores product categories with a unique ID and category name.

CREATE TABLE CATEGORY (

CATEGORY\_ID VARCHAR(20) PRIMARY KEY,

CATEGORY\_NAME VARCHAR(20)

);

* **PRODUCT** - Stores product details like name, category, cost price, selling price, and stock.

CREATE TABLE PRODUCT (

PRODUCT\_ID VARCHAR(10) PRIMARY KEY, PRODUCT\_NAME,

VARCHAR(40),

CATEGORY\_ID VARCHAR(20),

COST\_PRICE DECIMAL(10,2),

SELLING\_PRICE DECIMAL(10,2),

STOCK INTEGER,

CONSTRAINT CHECK\_STOCK CHECK (STOCK >= 0),

FOREIGN KEY (CATEGORY\_ID) REFERENCES CATEGORY

(CATEGORY\_ID)

);

* **EMPLOYEE** - Stores employee information like name, gender, salary, and a unique ID.

CREATE TABLE EMPLOYEE (

EMP\_ID VARCHAR(10) PRIMARY KEY,

FIRST\_NAME VARCHAR(15),

LAST\_NAME VARCHAR(15),

EMP\_GENDER ENUM('MALE', 'FEMALE'),

EMP\_SALARY DECIMAL(10,2)

);

* **ORDERS** - Stores order details like customer ID, amount, and order date.

CREATE TABLE ORDERS (

ORDER\_ID VARCHAR(10) PRIMARY KEY,

CUSTOMER\_ID VARCHAR(10),

AMOUNT DECIMAL(10,2),

ORDER\_DATE DATE,

FOREIGN KEY (CUSTOMER\_ID) REFERENCES CUSTOMER(C\_ID)

);

* **ORDER\_ITEM** - Stores details of items in an order, including product ID and quantity.

CREATE TABLE ORDER\_ITEM (

ORDER\_ID VARCHAR(10),

PRODUCT\_ID VARCHAR(10),

QUANTITY INTEGER,

PRIMARY KEY (ORDER\_ID, PRODUCT\_ID),

FOREIGN KEY (ORDER\_ID) REFERENCES ORDERS(ORDER\_ID),

FOREIGN KEY (PRODUCT\_ID) REFERENCES PRODUCT(PRODUCT\_ID)

);

* **PAYMENT** - Stores payment details like order ID, payment mode, amount, and payment date.

CREATE TABLE PAYMENT (

PAY\_ID VARCHAR(10) PRIMARY KEY,

ORDER\_ID VARCHAR(10),

PAY\_MODE ENUM('CASH', 'UPI', 'CARD'),

AMOUNT DECIMAL(10,2),

PAY\_DATE TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (ORDER\_ID) REFERENCES ORDERS(ORDER\_ID)

);

* **SALES** - Stores sales information like product ID, quantity sold, and profit.

CREATE TABLE SALES (

PRODUCT\_ID VARCHAR(10),

QUANTITY\_SOLD INTEGER,

PROFIT DECIMAL(10,2),

FOREIGN KEY (PRODUCT\_ID) REFERENCES PRODUCT(PRODUCT\_ID)

);

* **EMPLOYEE\_MOBILE** - Stores employee mobile numbers, allowing multiple numbers for one employee.

CREATE TABLE EMPLOYEE\_MOBILE (

EMP\_ID VARCHAR(10),

EMP\_MOBILE VARCHAR(10),

PRIMARY KEY (EMP\_ID, EMP\_MOBILE),FOREIGN KEY (EMP\_ID) REFERENCES EMPLOYEE(EMP\_ID)

);

* **EMPLOYEE\_ORDER** - Stores the relationship between employees and the orders they process.

CREATE TABLE EMPLOYEE\_ORDER (

EMP\_ID VARCHAR(10),

ORDER\_ID VARCHAR(10),

PRIMARY KEY (EMP\_ID, ORDER\_ID),

FOREIGN KEY (EMP\_ID) REFERENCES EMPLOYEE(EMP\_ID),

FOREIGN KEY (ORDER\_ID) REFERENCES ORDERS(ORDER\_ID)

);

**INSERTED VALUES**

INSERT INTO CUSTOMER VALUES ('C001', 'RAJESH KUMAR', 9876543210);  
INSERT INTO CUSTOMER VALUES ('C002', 'SONIA GUPTA', 9812345678);  
INSERT INTO CUSTOMER VALUES ('C003', 'NEHA SHARMA', 9810987654);  
INSERT INTO CUSTOMER VALUES ('C004', 'AYUSH PATEL', 6587541232);  
  
INSERT INTO CATEGORY VALUES('C101','BAKERY');  
INSERT INTO CATEGORY VALUES('C102','BEVERAGES');  
INSERT INTO CATEGORY VALUES('C103','DAIRY');  
INSERT INTO CATEGORY VALUES('C104','GROCERY');  
INSERT INTO CATEGORY VALUES('C105','PRODUCE');  
INSERT INTO CATEGORY VALUES('C106','CEREALS');  
INSERT INTO CATEGORY VALUES('C107','SNACKS');  
INSERT INTO CATEGORY VALUES('C108','HOUSEHOLD ITEMS');  
INSERT INTO CATEGORY VALUES('C109','BEAUTY');   
  
INSERT INTO ORDER\_ITEM VALUES ('O001', 'P101', 5);  
INSERT INTO ORDER\_ITEM VALUES ('O001', 'P102', 3);  
INSERT INTO ORDER\_ITEM VALUES ('O002', 'P103', 2);  
  
INSERT INTO PRODUCT VALUES('P101','OATS', 'C106',150,200,89);  
INSERT INTO PRODUCT VALUES('P102','MILK', 'C103',45,50,157);  
INSERT INTO PRODUCT VALUES('P103','KETCHUP', 'C104',90,100,50);  
INSERT INTO PRODUCT VALUES('P104','RICE', 'C106',500,550,26);  
INSERT INTO PRODUCT VALUES('P105','BANANA', 'C105',80,100,69);  
INSERT INTO PRODUCT VALUES('P106','COKE', 'C102',100,125,46);  
INSERT INTO PRODUCT VALUES('P107','HAIR OIL', 'C109',80,85,72);  
INSERT INTO PRODUCT VALUES('P108','RICE', 'C101', 350, 400, 100);   
  
INSERT INTO ORDERS VALUES ('O001', 'C004', 4500, '2022-02-23');  
INSERT INTO ORDERS VALUES ('O002', 'C002', 3200, '2023-03-15');  
INSERT INTO ORDERS VALUES ('O003', 'C001', 1200, '2022-01-08');  
INSERT INTO ORDERS VALUES ('O004', 'C003', 1200, '2022-06-11');   
  
INSERT INTO PAYMENT VALUES ('P001', 'O001', 'UPI', 4500, '2022-03-14');  
INSERT INTO PAYMENT VALUES ('P002', 'O002', 'CARD', 3200, '2022-03-15');  
INSERT INTO PAYMENT VALUES ('P003', 'O003', 'CASH', 1200, '2022-03-15');   
  
INSERT INTO EMPLOYEE VALUES ('E001', 'SURESH', 'KUMAR', 'MALE', 30000);  
INSERT INTO EMPLOYEE VALUES ('E002', 'RENU', 'SINGH', 'FEMALE', 25000);  
INSERT INTO EMPLOYEE VALUES ('E003', 'ANIL', 'VERMA', 'MALE', 28000);  
  
INSERT INTO EMPLOYEE\_MOBILE VALUES ('E001', 9879654218);  
INSERT INTO EMPLOYEE\_MOBILE VALUES ('E002', 7845321975);  
INSERT INTO EMPLOYEE\_MOBILE VALUES ('E003', 6548724387);  
  
INSERT INTO EMPLOYEE\_ORDER VALUES ('E001', 'O001');  
INSERT INTO EMPLOYEE\_ORDER VALUES ('E002', 'O002');  
INSERT INTO EMPLOYEE\_ORDER VALUES ('E003', 'O003');   
INSERT INTO SALES VALUES ('P101', 15, 5000);  
INSERT INTO SALES VALUES ('P102', 10, 300);  
INSERT INTO SALES VALUES ('P103', 5, 500);

**FUNCTIONAL DEPENDENCIES**

1. **CUSTOMER:**

C\_ID -> C\_NAME, C\_MOBILE

1. **CATEGORY:**

CATEGORY\_ID -> CATEGORY\_NAME

1. **PRODUCT:**

PRODUCT\_ID -> PRODUCT\_NAME, CATEGORY\_ID, COST\_PRICE, SELLING\_PRICE, STOCK

1. **EMPLOYEE:**

EMP\_ID -> FIRST\_NAME, LAST\_NAME, EMP\_GENDER, EMP\_SALARY, EMP\_MOBILE

1. **ORDERS:**

ORDER\_ID -> CUSTOMER\_ID, AMOUNT, ORDER\_DATE, PRODUCTS

1. **ORDER\_ITEM:**

(ORDER\_ID, PRODUCT\_ID) -> QUANTITY

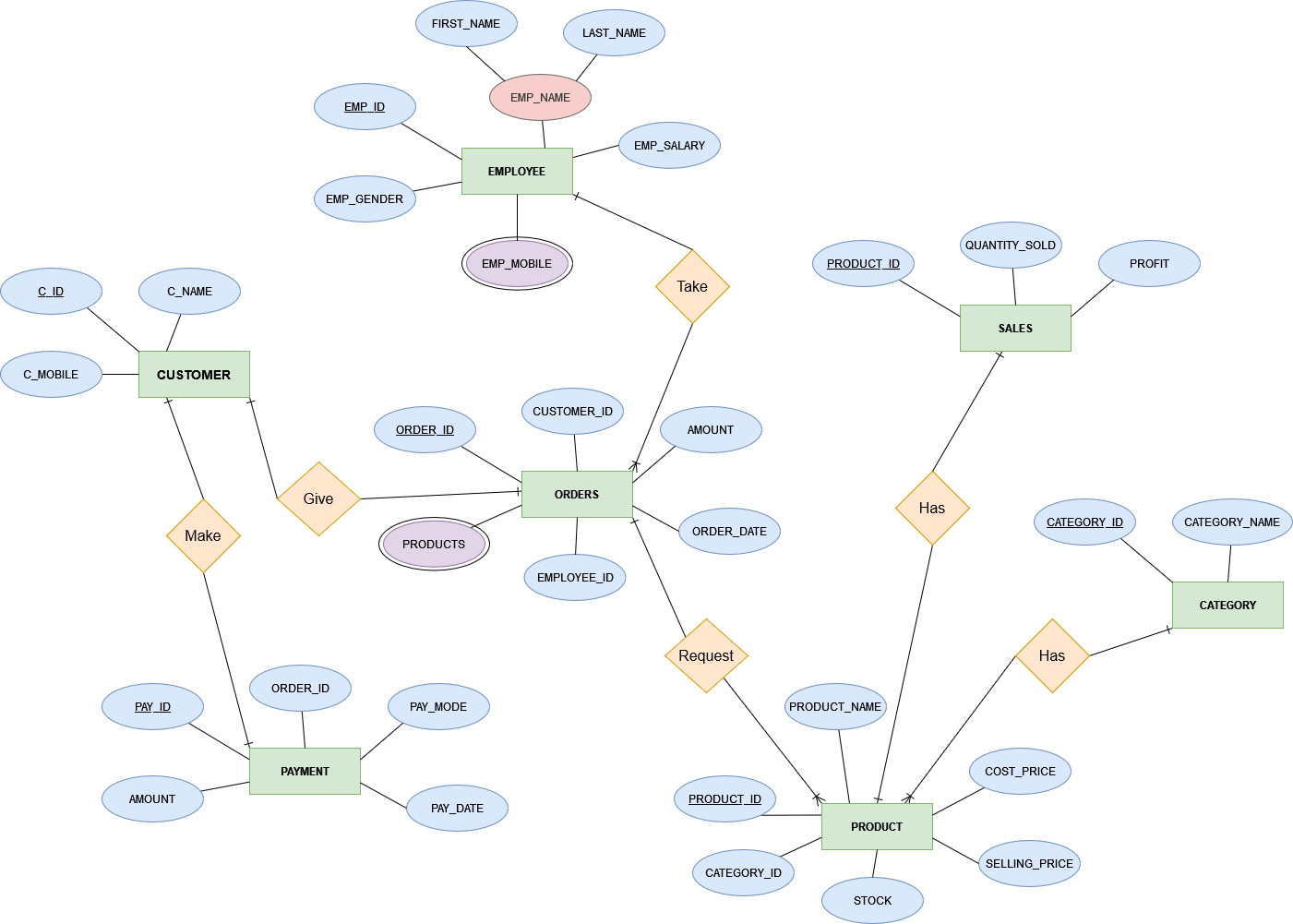
1. **PAYMENT:**

PAY\_ID -> ORDER\_ID, PAY\_MODE, AMOUNT, PAY\_DATE

1. **SALES:**

PRODUCT\_ID -> QUANTITY\_SOLD, PROFIT

**ENTITY RELATIONSHIP DIAGRAM**



**NORMALIZATION**

**1st Normal Form**

First Normal Form (**1NF**) requires that each column in a table should contain atomic values, meaning that a column should only contain one value for each record, and that value should not be divisible further.

Our Schema contained two **multivalued attributes** EMPLOYEE (EMP\_MOBILE) & ORDRS (PRODUCTS). So, we created two more relations EMP\_MOBILE & ORDER\_ITEMS.

Our Schema contained a **composite** **attributes** EMPLOYEE (EMP\_NAME). So, we divided that into two attributes FIRST\_NAME, LAST\_NAME.

**2nd Normal Form**

2nd Normal Form (**2NF**) is a database normalization stage that ensures all non-prime attributes (attributes not part of the primary key) are fully functionally dependent on the primary key and no partial dependencies exist.

Our Schema is already in 2NF, because all tables have a single attribute primary key or a composite primary key, ensuring each row is uniquely identifiable.

Non-prime attributes in each table are fully functionally dependent on the primary key, and no partial dependencies are present. For example, in the PRODUCT table, all **non-prime attributes** (PRODUCT\_NAME, CATEGORY\_ID, COST\_PRICE, SELLING\_PRICE, STOCK) depend on the primary key (PRODUCT\_ID) and not just a part of it.

**3rd Normal Form**

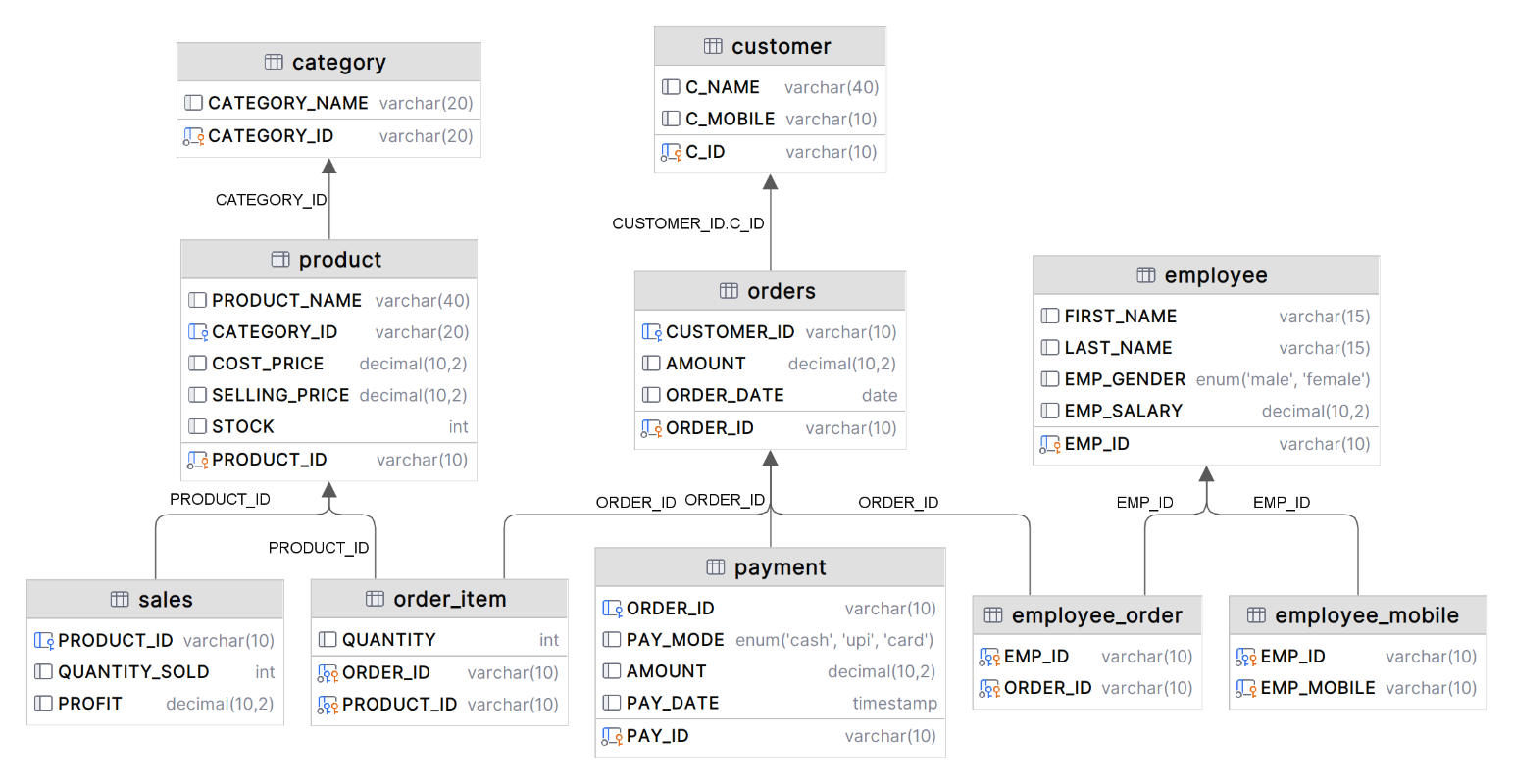
3rd Normal Form (**3NF**) is a database normalization stage that ensures a table is in 2NF and, additionally, all non-prime attributes are non-transitively dependent on the primary key. In other words, there should be no functional dependencies between non-prime attributes.

Our Schema is already in 3NF, because all tables are in 2NF, as previously mentioned.

Non-prime attributes in each table are directly dependent on the primary key, and there are **no transitive dependencies** between non-prime attributes. For example, in the PRODUCT table, non-prime attributes like COST\_PRICE and SELLING\_PRICE depend only on the primary key (PRODUCT\_ID) and not on any other non-prime attribute.

By satisfying these conditions, the schema adheres to the principles of 3rd Normal Form, further reducing redundancy and ensuring that the data is free of insertion, update, and deletion anomalies.

**NORMALIZED RELATIONAL MODEL**



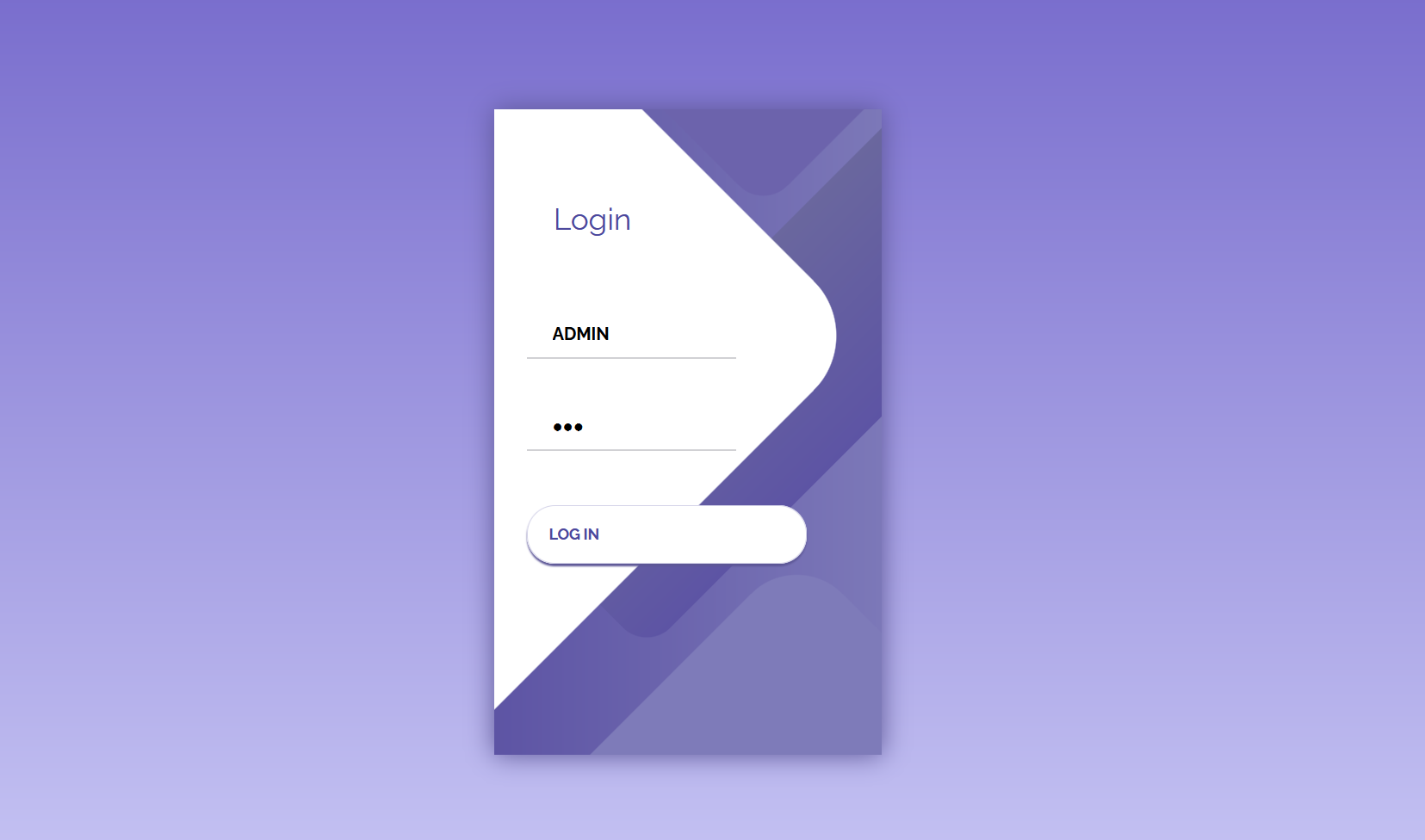
* **Customer** (C\_Id, C\_Name, C\_Mobile)
* **Category** (Category\_Id, Category\_Name)
* **Product** (Product\_Id, Product\_Name, Category\_Id, Cost\_Price, Selling\_Price, Stock)
* **Employee** (Emp\_Id, First\_Name, Last\_Name, Emp\_Gender, Emp\_Salary)
* **Orders** (Order\_Id, Customer\_Id, Amount, Order\_Date)
* **Order\_Item** (Orser\_Id, Product\_Id, Quantity)
* **Payment** (Pay\_Id, Order\_Id, Pay\_Mode, Amount, Pay\_Date)
* **Sales** (Product\_Id, Quantity\_Sold, Profit)
* **Employee\_Mobile** (Emp\_Id, Emp\_Mobile)
* **Employee\_Order** (Emp\_Id, Order\_Id)

**TECH STACKS USED**

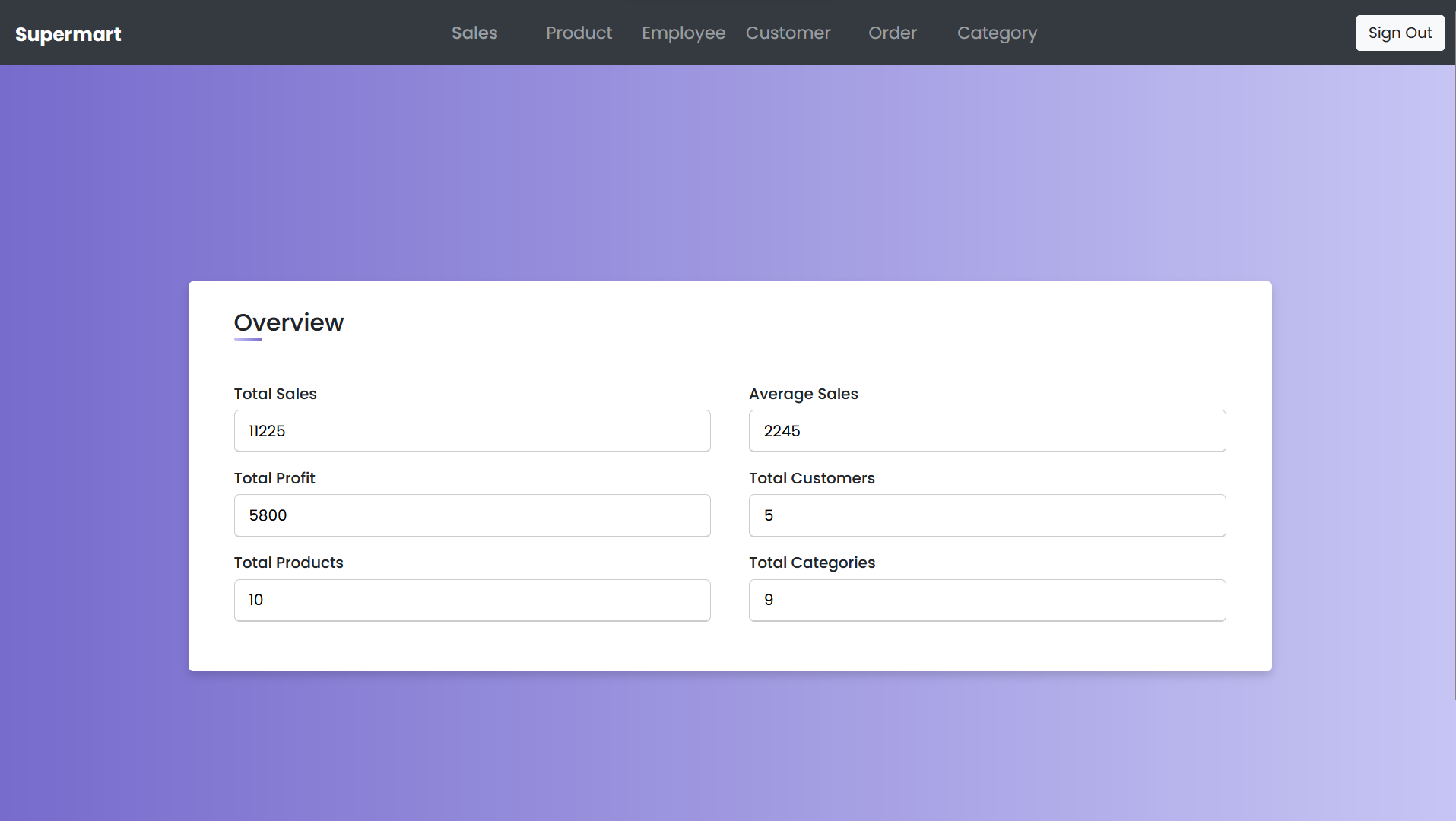
* MySQL
* NodeJS
* EJS
* CSS
* Nodemon
* Express
* Body Parser
* Visual Studio Code
* MySQL Workbench 8.0 CE

**FRONT END INTERFACE**

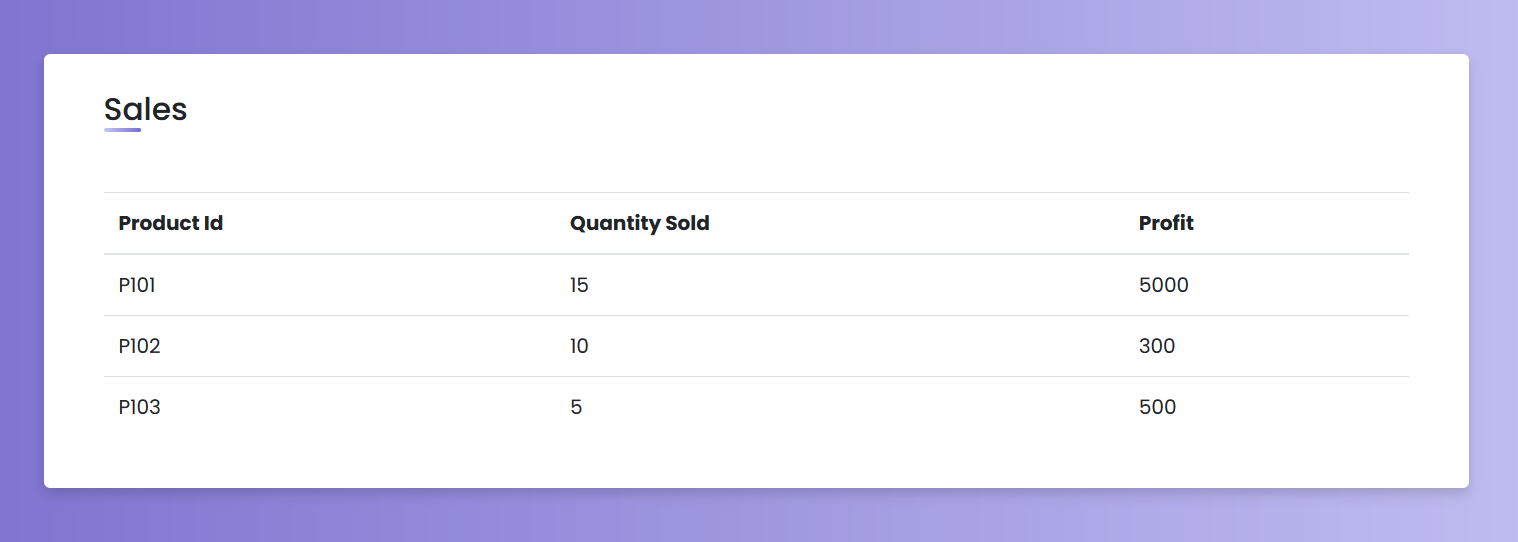
* **Login Page**

****

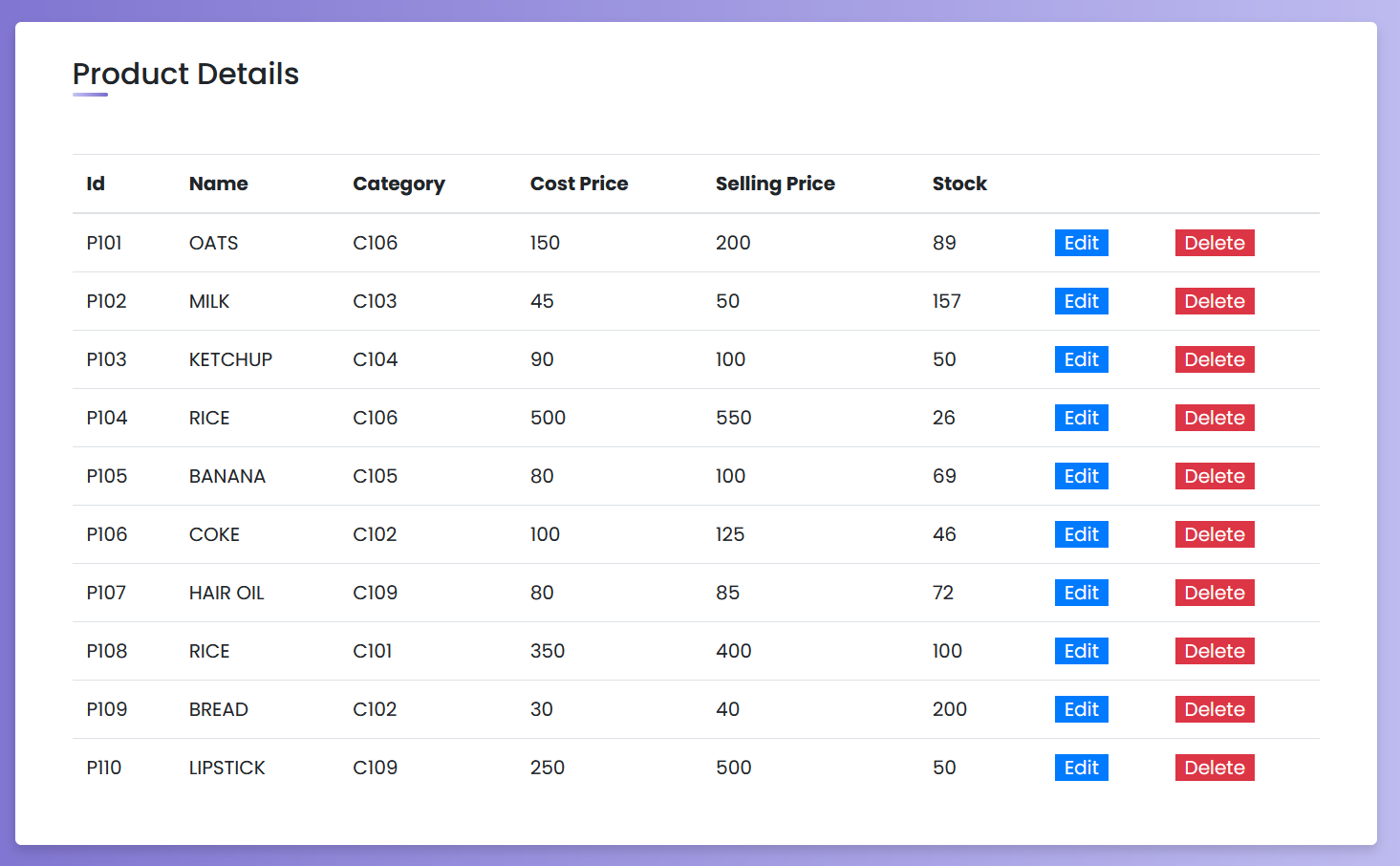
* **Dashboard**



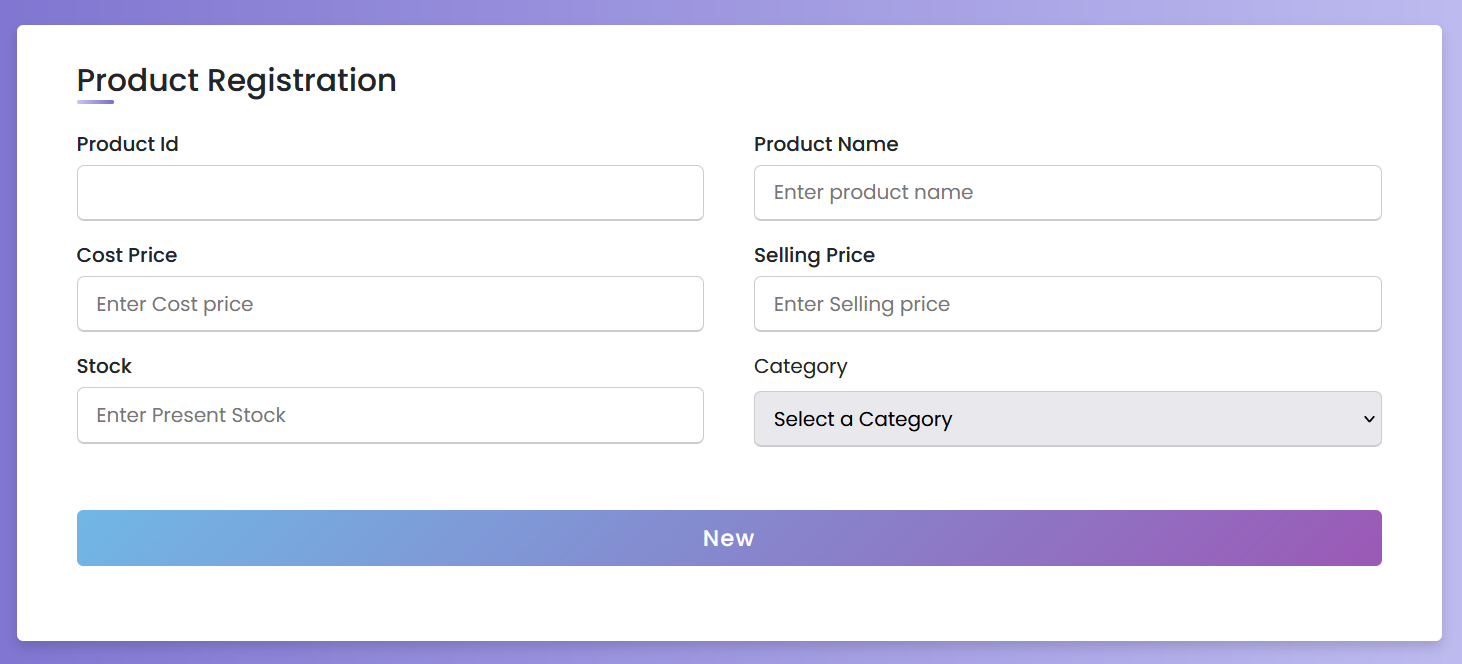
* **Sales**

****

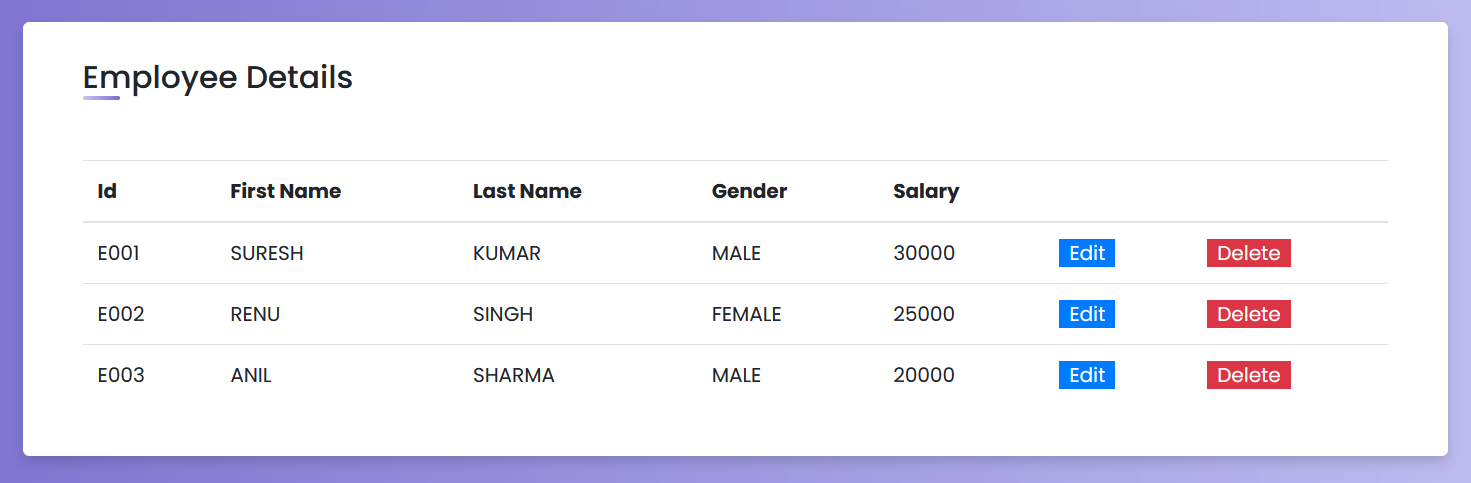
* **Products**

****

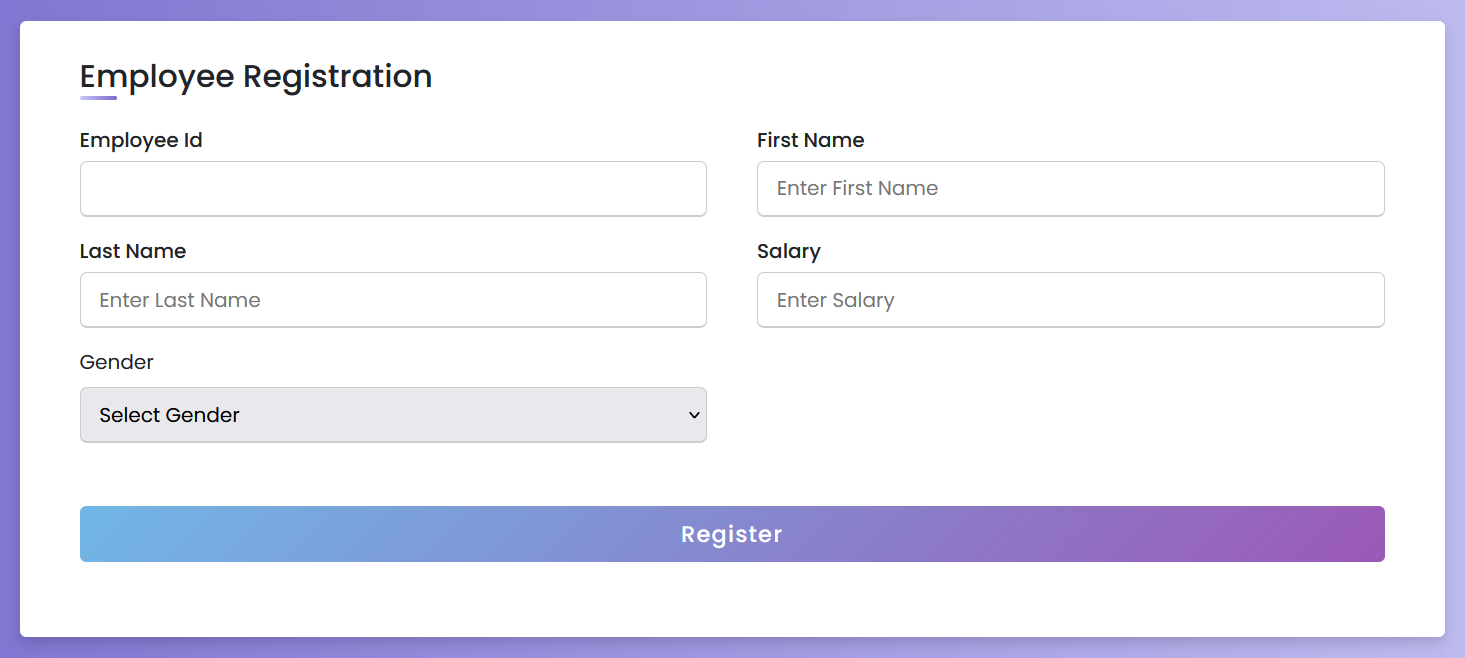
* **New Product**

****

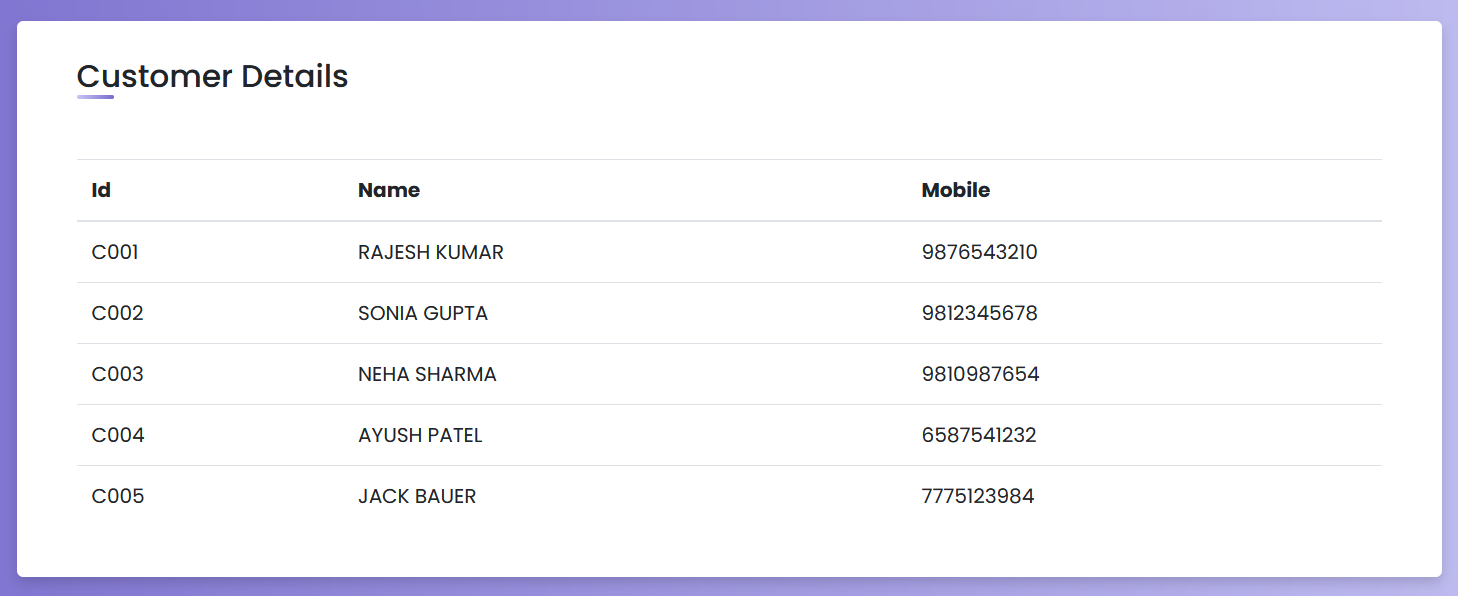
* **Employees**

****

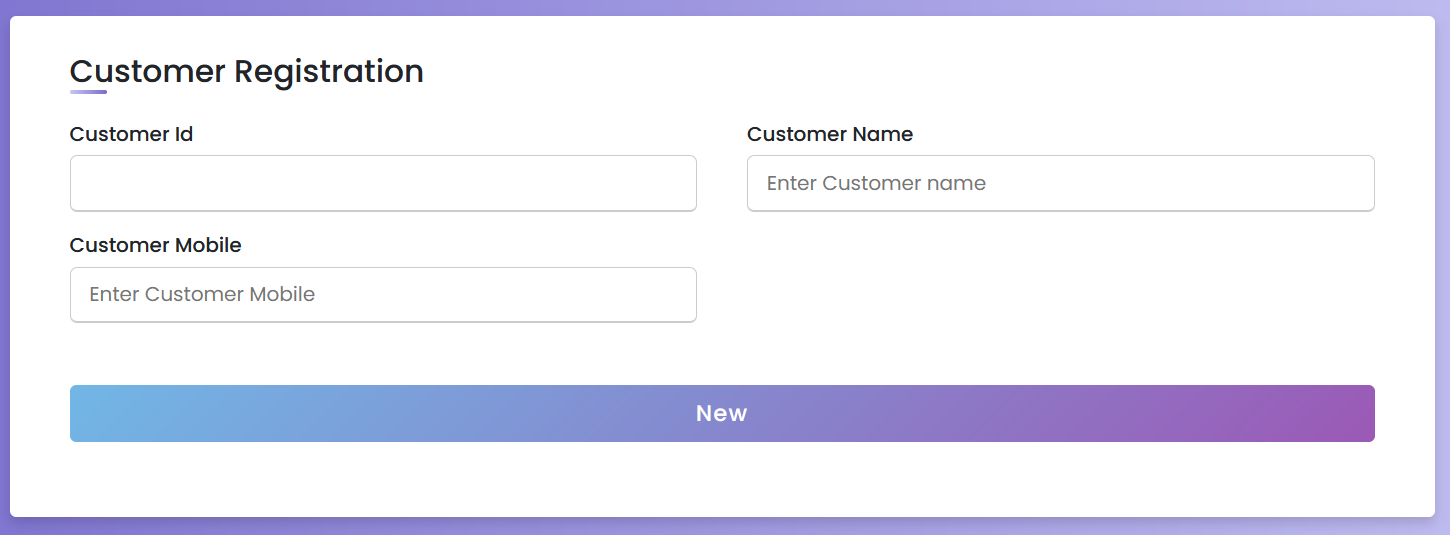
* **New Employee**

****

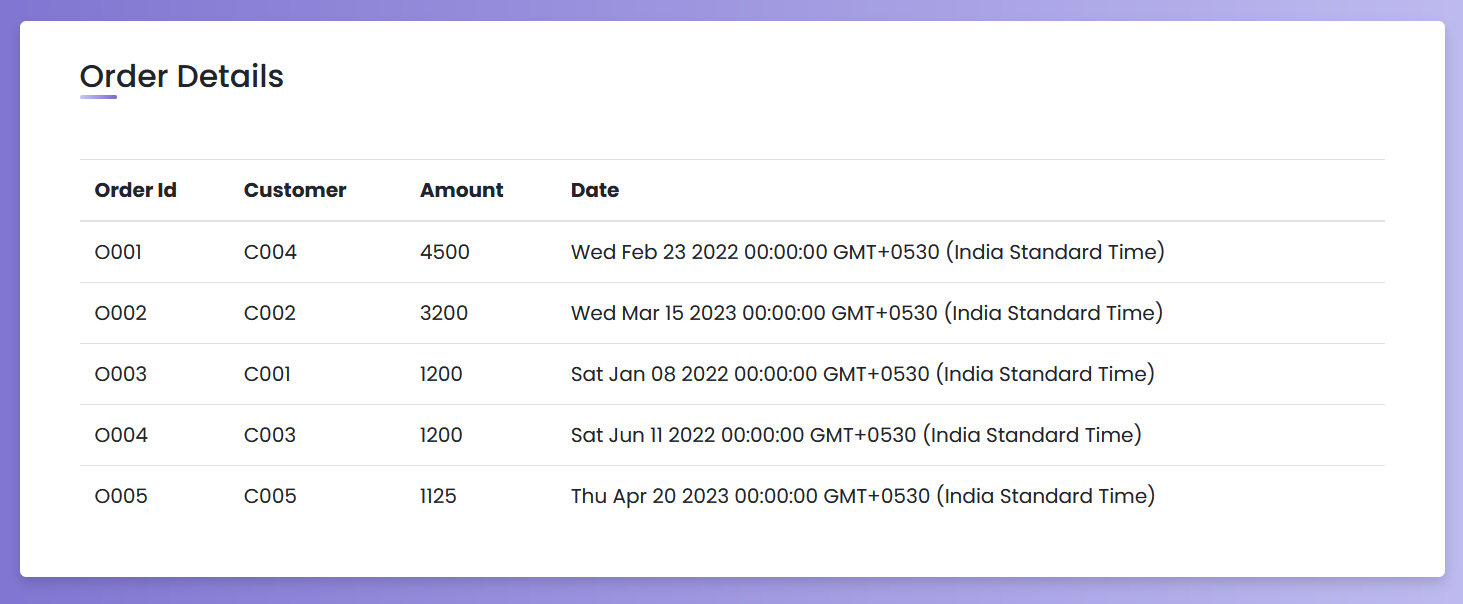
* **Customers**

****

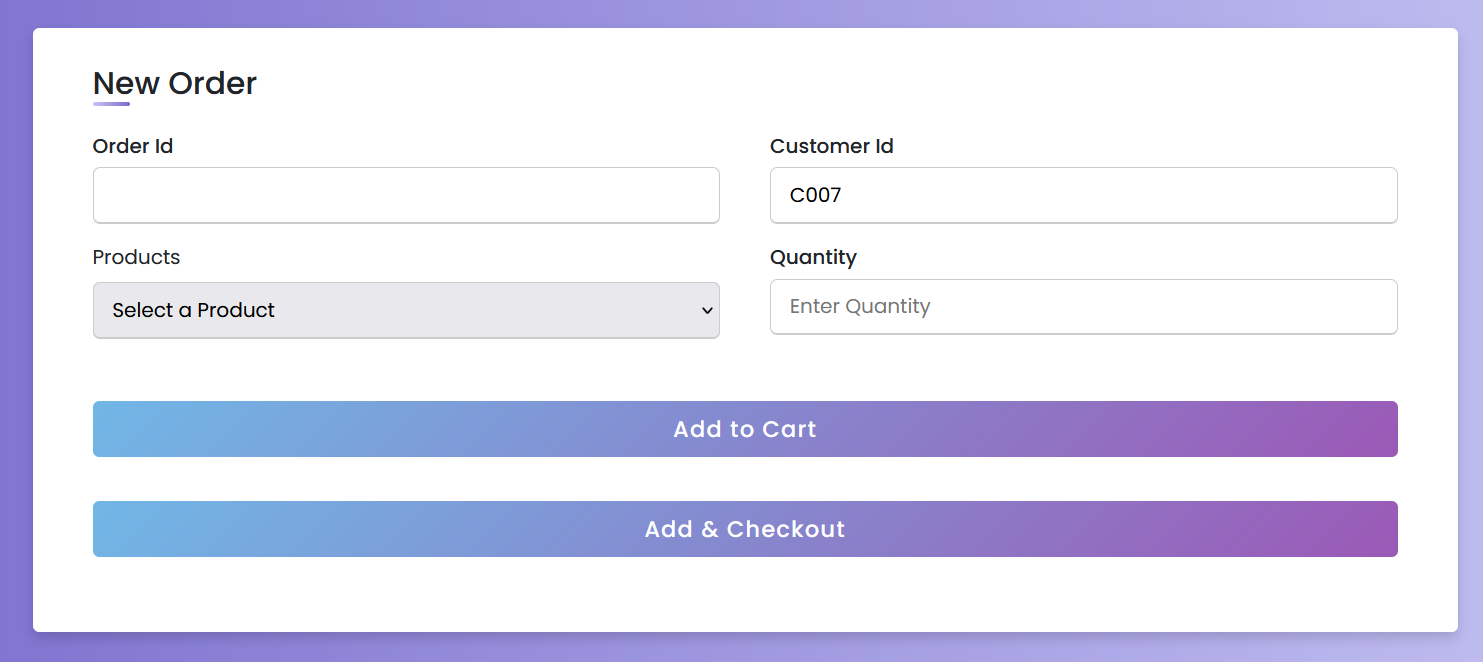
* **New Customer**

****

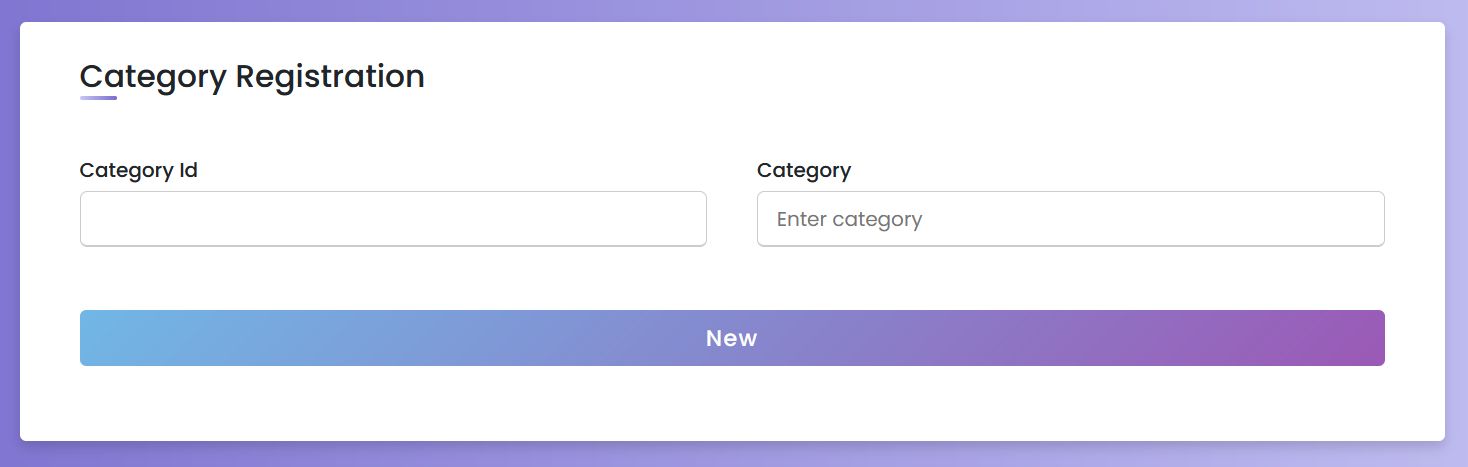
* **Orders**

****

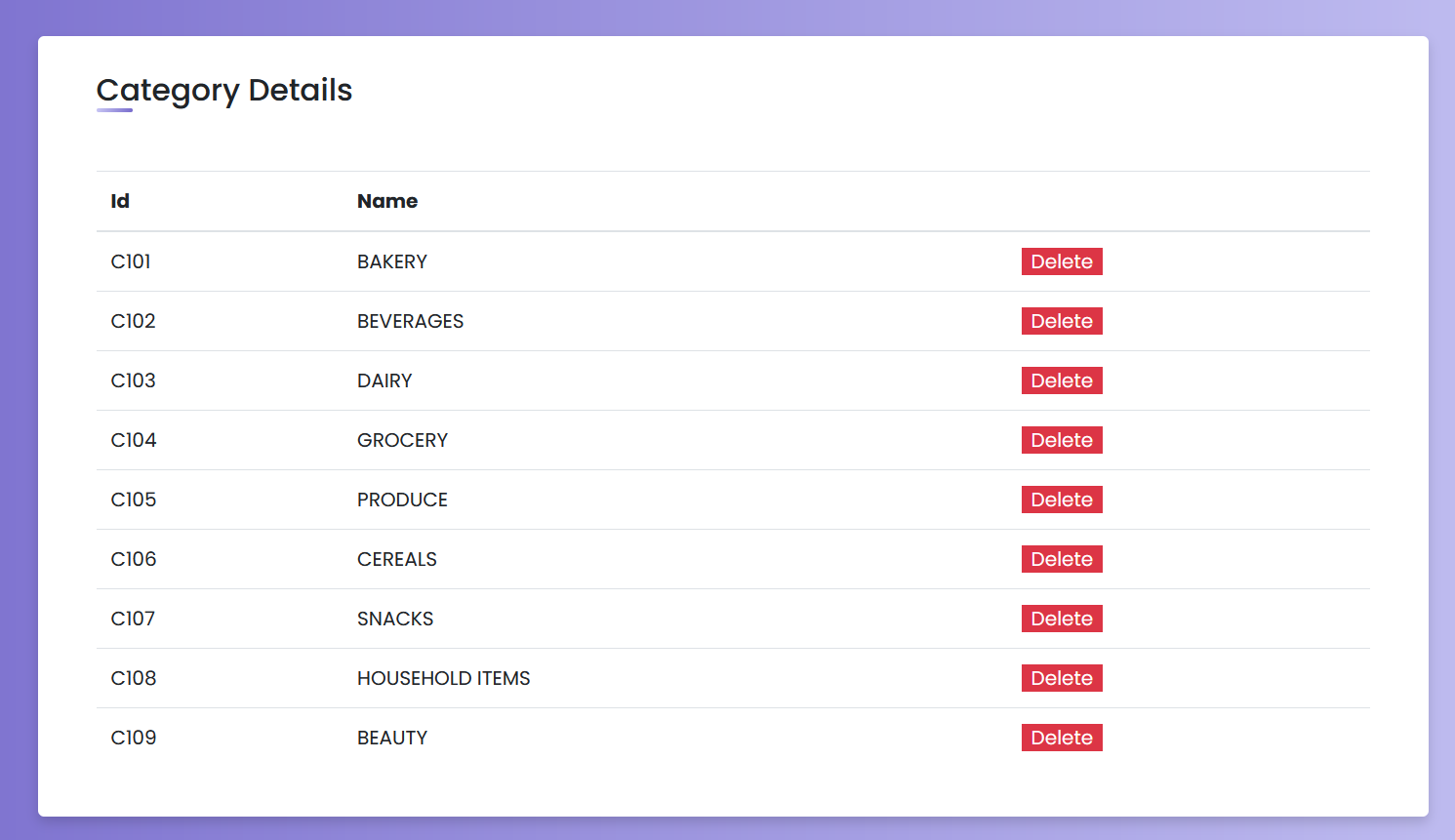
* **New Order**

****

* **New Categories**

****

* **Categories**

****

**BACK-END CONNECTIVITY**

var connection = mysql.createConnection({

  host: "localhost",

  user: "root",

  password: "pavilion",

  database: "SUPERMART",

  insecureAuth: true,

});

connection.connect(function (err) {

  if (err) throw err;

  console.log("MySQL Connected!");

});

**SAMPLE MYSQL QUE RIES IN NODEJS**

1. **INSERT**

var sql =

    "INSERT INTO PRODUCT VALUES (?, ?, ? , " +

    productCostPrice +

    ", " +

    productSellingPrice +

    ", " +

    productStock +

    ");";

  connection.query(

    sql,

    [id, productName, productCategory],

    function (err, data) {

      if (err) throw err;

      console.log(data.affectedRows + " record(s) updated");

      res.redirect("/products");

    }

  );

1. **SELECT**

 var sql = "SELECT \* FROM product ORDER BY PRODUCT\_ID";

  connection.query(sql, function (err, productData, fields) {

    if (err) throw err;

    res.render("products", { title: "Products", productData: productData });

  });

1. **UPDATE**

 var sql =

    "UPDATE PRODUCT SET PRODUCT\_NAME = ?, CATEGORY\_ID = ?, COST\_PRICE = " +

    editedCostPrice +

    " ,SELLING\_PRICE = " +

    editedSellingPrice +

    " ,STOCK = " +

    editedStock +

    " WHERE PRODUCT\_ID = ?;";

  connection.query(sql, [editedName, editedCategory, id], function (err, data) {

    if (err) throw err;

    console.log(data.affectedRows + " record(s) updated");

    res.redirect("/products");

  });

1. **DELETE**

var sql = "DELETE FROM product WHERE PRODUCT\_ID = ?";

  connection.query(sql, [req.params.deleteId], function (err, data) {

    if (err) throw err;

    console.log(data.affectedRows + " record(s) deleted");

    res.redirect("/products");

  });

**CONCLUSION**

In conclusion, the development of a supermarket management system is a significant step towards improving the efficiency and productivity of supermarkets. The project aimed to design and implement a software system that would automate various supermarket operations, such as inventory management, sales tracking, and customer management.

Throughout the project, we identified the key requirements of the system, designed the architecture, implemented the functionality, and tested the system thoroughly to ensure its reliability and accuracy. We also ensured that the system could adapt to the changing needs of the supermarket and its customers.

The successful implementation of the supermarket management system offers several benefits, including improved inventory management, accurate sales tracking, better customer experience, and increased profitability. It also eliminates the need for manual processes, reducing the chances of errors and saving time.

Overall, the supermarket management system represents a significant technological advancement for supermarkets, providing them with a competitive edge in the market. The project has been a great learning experience, allowing us to apply our knowledge and skills in software development, and we are confident that it will bring tangible benefits to supermarket owners, employees, and customers.