SVKM's NMIMS

Mukesh Patel School of Technology Management & Engineering (Mumbai Campus)

Computer Engineering Department (B Tech CSE/CSBS Sem IV/BTI Sem VIII/MBA.Tech-IV)

Database Management System

Project Report

Program	Btech CSBS	
Semester	IV	
Name of the Project	Online Shopping Management System	
Details of Project Members		
Batch	Roll No.	Name
1	E033	Yug Marfatia
2	E047	Aayusman Pati
2	E051	Rajvardhan Rao
Date of Submission: 13-04-25		

Contribution of each project Members:

Roll No.	Name:	Contribution
E033	Yug Marfatia	ER diagram, Relational model, Normalization, database sql queries, frontend design and coding
E047	Aayusman Pati	ER diagram, Relational model, Normalization, database sql queries, frontend design and coding
E051	Rajvardhan Rao	ER diagram, Relational model, Normalization, database sql queries, frontend design and coding

Github link of your project: https://github.com/aayusman05/E-commerce-website

Note:

1. Create a readme file if you have multiple files

2. All files must be properly named (Example:R004_DBMSProject)
3. Submit all relevant files of your work (Report, all SQL files, Any other files)
4. **Plagiarism is highly discouraged (Your report will be checked for plagiarism)**

Rubrics for the Project evaluation:

First phase of evaluation: Innovative Ideas (5 Marks) Design and Partial implementation (5 Marks)	10 marks
Final phase of evaluation Implementation, presentation and viva, Self-Learning and Learning Beyond classroom	10 marks

Project Report

Selected Topic

by

Student 1, Roll number: E033

Student 2, Roll number: E047

Student 3, Roll number: E051

Course: DBMS

AY: 2024-25

I. Storyline

Online shopping has been the most popular way to buy products in the current digital era because of its convenience and wide selection. Building a database system for an e-commerce website is the main goal of our project.

This project aims to demonstrate fundamental DBMS principles like normalisation, relationships, and complex SQL queries while designing and implementing a fully functional database that replicates an actual online shopping experience.

II. Components of Database Design

Entities and their attributes:

1) Customer: cust_id, cust_name, cust_email, cust_contact_no, user_id

2) Order: order_id, order_date, cust_id, cust_add, total_amt

3) OrderReturn: order_return_id, order_id, cust_id, return_date, prod_id, return_amount

4) ShoppingCart: prod_id, cust_id, cart_id

5) Product: prod_id, prod_name, prod_qty, prod_price, prod_color

6) OrderDetail: order_id, prod_id, prod_qty, prod_price, prod_total_price

7) Bill_detail: bill_id, bill_date, cust_id, order_id, cust_name, prod_id, prod_qty, prod_price, prod_total_price, order_total_price, offer_disc, bill_total_price, pay_status

8) Payment: pay_id, pay_mode, pay_date, pay_disc, pay_amount, order_id, cust_id, pay_status

9) User: User_id, Username, Password_hash

10) Employee: employee_id, employee_name, employee_email, employee_contact_no, Designation, User_id

11) Final Bill Detail: order_id, order_total_price, Offer_discount, Bill_total_price, Bill_id

Relationships and Cardinality:

1) Customer → Places → Order

Cardinality: 1:N
A customer can place multiple orders, but each order is placed by only one customer.

Participation: Total for Order (Each order must be placed by a customer), Optional for Customer (A customer may not place an order).

2) Customer → Adds in → Shopping Cart

Cardinality: 1:N
A customer can add multiple products to the shopping cart, but each cart item belongs to only one customer.

Participation: Total for Shopping Cart (Each item in the shopping cart must be associated with a customer), Optional for Customer (A customer may not have an item in the cart).

3) Customer → Initiates → OrderReturn

Cardinality: 1:N
A customer can initiate multiple returns, but each return is initiated by only one customer.

Participation: Total for OrderReturn (Each return must be initiated by a customer), Optional for Customer (A customer may not initiate any return).

4) Order → Contains → OrderDetail

Cardinality: 1:N
An order can contain multiple order details (representing each product in the order), but each order detail belongs to only one order.

Participation: Total for OrderDetail (Each order must have at least one product listed in the order details), Total for Order (An order must contain at least one order detail).

5) OrderDetail → Refers → Product

Cardinality: N:1
Multiple order details can refer to the same product (e.g., if multiple quantities of the same product are ordered), but each order detail refers to only one product.

Participation: Total for OrderDetail (Each order detail must reference a product), Optional for Product (A product may not be in an order detail if not ordered).

6) Order → Paid via → Payment

Cardinality: 1:1
Each order is paid via a single payment, and each payment corresponds to only one order.

Participation: Total for both Order and Payment (An order must have a corresponding payment, and a payment must be linked to an order).

7) Payment → Recorded in → BillDetail

Cardinality: 1:1
Each payment is recorded in a single bill detail, and each bill detail records only one payment.

Participation: Total for both Payment and BillDetail (Every payment should be recorded in a bill detail, and each bill detail must correspond to a payment).

8) Order → Generates → BillDetail

Cardinality: 1:N

An order generates multiple bill details (e.g., if there are multiple products or services being billed), but each bill detail is associated with only one order.

Participation: Total for BillDetail (Each bill detail must correspond to an order), Optional for Order (An order may not have a corresponding bill detail yet).

9) Product → Added to → ShoppingCart

Cardinality: 1:N
A product can be added to multiple shopping carts, but each cart item refers to only one product.

Participation: Total for ShoppingCart (Each item in the cart must reference a product), Optional for Product (A product may not be in any cart).

10) Product → Returned in → OrderReturn

Cardinality: 1:N
A product can be returned multiple times in different order returns, but each order return refers to only one product.

Participation: Total for OrderReturn (Each return must reference a product), Optional for Product (A product may not be returned).

11) OrderReturn → Associated with → Order

- Cardinality: N:1
- A return is tied to an order. Each return must be for an order.
- Participation: Total for OrderReturn, Optional for Order

12) Customer → Has → User

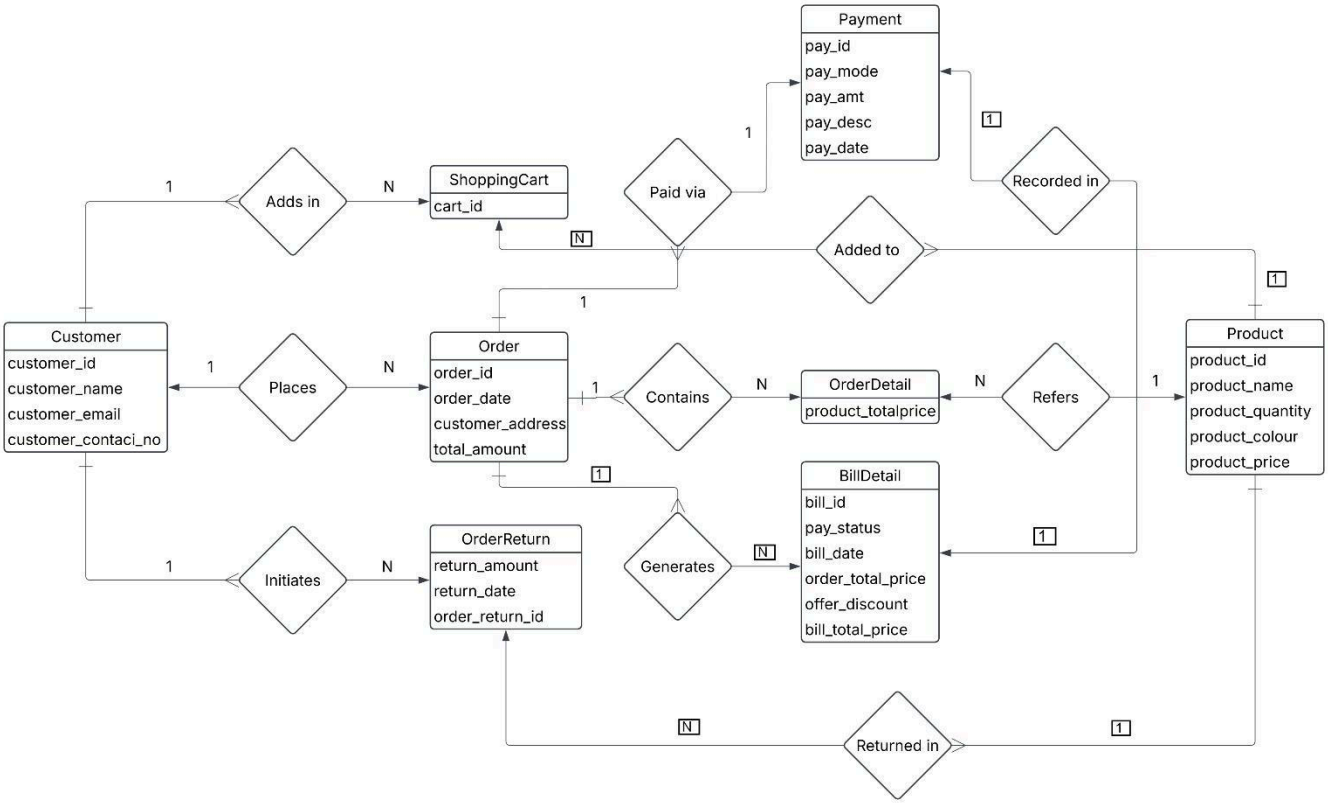
- Cardinality: 1 : 1
- Participation: Total for Customer, Total for User (from schema constraints)

13) Employee → Has → User

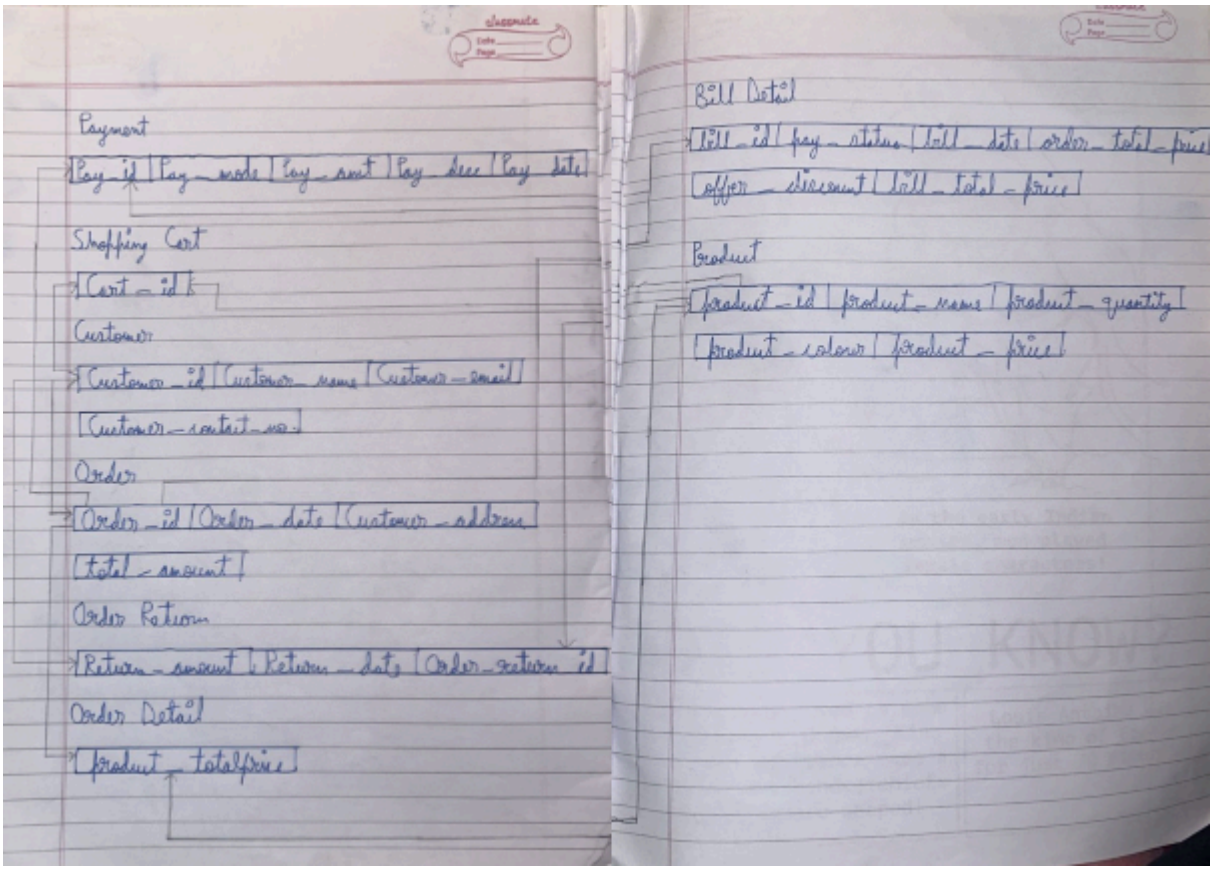
- Cardinality: 1:1 (each employee is a user)

- Participation: Total for both (assuming login access is mandatory)

III. Entity Relationship Diagram



IV. Relational Model



1. Payment

Attributes:

- Pay_id (INT)
- Pay_mode (VARCHAR(50))
- Pay_amt (DECIMAL(10,2))
- Pay_desc (VARCHAR(255))
- Pay_date (DATE)

Primary Key: Pay_id

2. Customer

Attributes:

- Customer_id (INT)
- Customer_name (VARCHAR(100))
- Customer_email (VARCHAR(100))
- Customer_contact_no (VARCHAR(15))
- User_id (INT)

Primary Key: Customer_id
Foreign Key: User_id → User(User_id)

3. Order

Attributes:

- Order_id (INT)
- Order_date (DATE)
- Customer_address (VARCHAR(255))
- Total_amt (DECIMAL(10,2))
- Customer_id (INT)

Primary Key: Order_id
Foreign Key: Customer_id → Customer(Customer_id)

4. Order_Return

Attributes:

- Return_amt (DECIMAL(10,2))
- Return_date (DATE)
- Order_id (INT)

Primary Key: Order_id
Foreign Key: Order_id → Orders(Order_id)

5. Bill_Detail

Attributes:

- Bill_id (INT)
- Pay_status (VARCHAR(50))
- Bill_date (DATE)
- Order_total_price (DECIMAL(10,2))
- Order_id (INT)

Primary Key: Bill_id
Foreign Key: Order_id → Orders(Order_id)

6. Final_Bill_Detail

Attributes:

- Order_tatal_price (DECIMAL(10,2)) — typo: should be Order_total_price
- Offer_discount (DECIMAL(10,2))
- Bill_total_price (DECIMAL(10,2))
- Bill_id (INT)

Primary Key: Bill_id
Foreign Key: Bill_id → Bill_Detail(Bill_id)

7. Product

Attributes:

- Product_id (INT)
- Product_name (VARCHAR(100))

- Product_qty (INT)
- Product_colour (VARCHAR(50))
- Product_price (DECIMAL(10,2))

Primary Key: Product_id

8. User

Attributes:

- User_id (INT)
- Username (VARCHAR(100))
- Password_hash (VARCHAR(255))
- Role (VARCHAR(50))

Primary Key: User_id

9. Employee

Attributes:

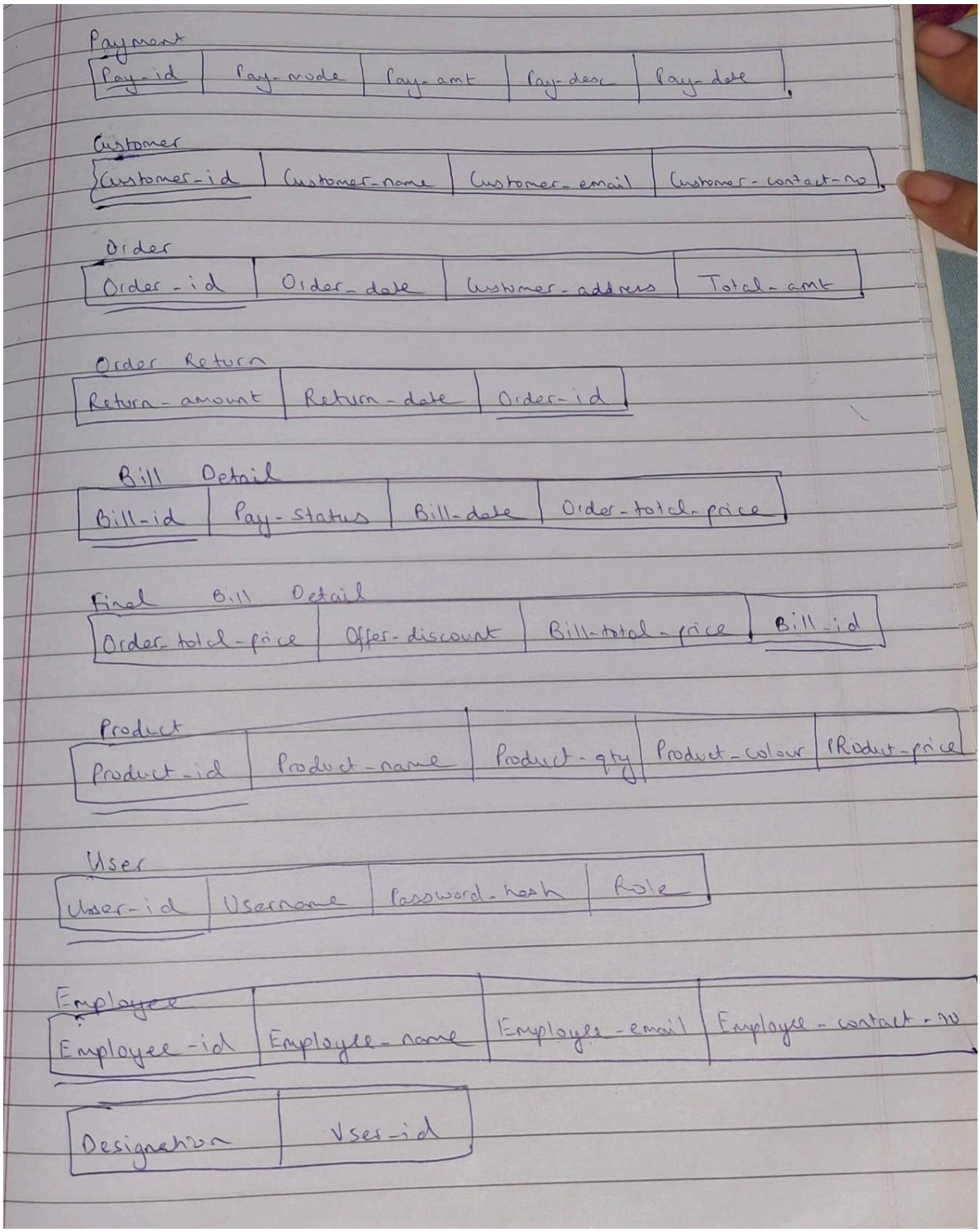
- Employee_id (INT)
- Employee_name (VARCHAR(100))
- Employee_email (VARCHAR(100))
- Employee_contact_no (VARCHAR(15))
- Designation (VARCHAR(100))
- User_id (INT)

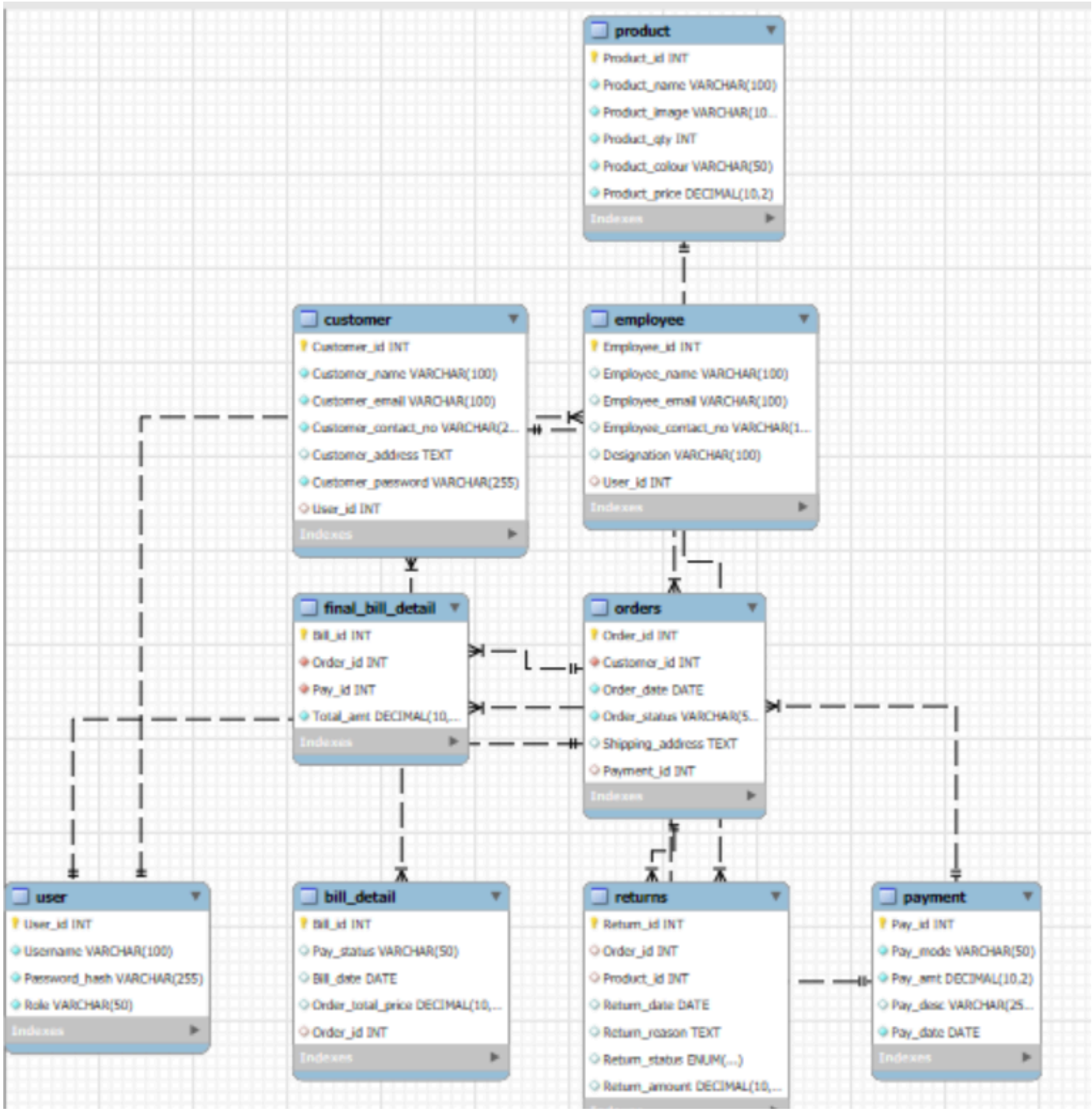
Primary Key: Employee_id
Foreign Key: User_id → User(User_id)

V. Normalization

The original relational model already satisfies 1NF and 2NF conditions.

3NF:




product
Product_id INT
Product_name VARCHAR(100)
Product_image VARCHAR(10...
Product_qty INT
Product_colour VARCHAR(50)
Product_price DECIMAL(10,2)
Indexes
customer
Customer_id INT
Customer_name VARCHAR(100)
Customer_email VARCHAR(100)
Customer_contact_no VARCHAR(2...
Customer_address TEXT
Customer_password VARCHAR(255)
User_id INT
Indexes
employee
Employee_id INT
Employee_name VARCHAR(100)
Employee_email VARCHAR(100)
Employee_contact_no VARCHAR(1...
Designation VARCHAR(100)
User_id INT
Indexes
final_bill_detail
Bill_id INT
Order_id INT
Pay_id INT
Total_amt DECIMAL(10,...
Indexes
orders
Order_id INT
Customer_id INT
Order_date DATE
Order_status VARCHAR(5...
Shipping_address TEXT
Payment_id INT
Indexes
user
User_id INT
Username VARCHAR(100)
Password_hash VARCHAR(255)
Role VARCHAR(50)
Indexes
bill_detail
Bill_id INT
Pay_status VARCHAR(50)
Bill_date DATE
Order_total_price DECIMAL(10,...
Order_id INT
Indexes
returns
Return_id INT
Order_id INT
Product_id INT
Return_date DATE
Return_reason TEXT
Return_status ENUM(...)
Return_amount DECIMAL(10,...
payment
Pay_id INT
Pay_mode VARCHAR(50)
Pay_amt DECIMAL(10,2)
Pay_desc VARCHAR(25...
Pay_date DATE
Indexes

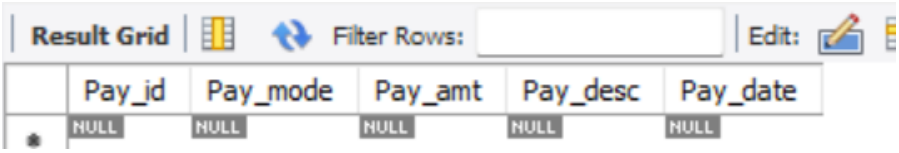
VI. SQL Queries

Using a DBMS software (SQLite3 or MySQL or any other of your choice):

- Create the tables
- Populate the tables (insert some meaningful data, at least 10 tuples for each relation)
- Run SQL queries (minimum 20) covering **all concepts** learned in the class

This section should contain the question, SQL code, and the output snapshot for each query.

```
CREATE TABLE Payment (
   Pay_id INT PRIMARY KEY,
   Pay_mode VARCHAR(50),
   Pay_amt DECIMAL(10, 2),
   Pay_desc VARCHAR(255),
   Pay_date DATE
);
```




```
CREATE TABLE Customer (
   Customer_id INT PRIMARY KEY,
   Customer_name VARCHAR(100),
   Customer_email VARCHAR(100),
   Customer_contact_no VARCHAR(15),
   User_id INT,
   FOREIGN KEY (User_id) REFERENCES User(User_id)
);
```


```
CREATE TABLE Orders (
   Order_id INT PRIMARY KEY,
   Order_date DATE,
   Customer_address VARCHAR(255),
   Total_amt DECIMAL(10, 2),
   Customer_id INT,
   FOREIGN KEY (Customer_id) REFERENCES Customer(Customer_id)
);
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

Order_id	Order_date	Customer_address	Total_amount	Customer_id	Payment_method	Order_status	Shipping_address
1	2025-04-13	NULL	1299.99	2	upi	Processing	Aayusman Pati 202/3, Raheja Tipco H
2	2025-04-13	NULL	1599.98	2	cod	Pending	Aayusman Pati 202/3, Raheja Tipco H
3	2025-04-13	NULL	849.98	4	netbanking	Processing	Aayusman Pati 202/3, Raheja Tipco H
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

```
CREATE TABLE Order_Return (
    Return_amt DECIMAL(10, 2),
    Return_date DATE,
    Order_id INT,
    FOREIGN KEY (Order_id) REFERENCES Orders(Order_id)
);
```

Result Grid | Filter Rows:

Return_amt	Return_date	Order_id

```
CREATE TABLE Bill_Detail (
    Bill_id INT PRIMARY KEY,
    Pay_status VARCHAR(50),
    Bill_date DATE,
    Order_total_price DECIMAL(10, 2),
    Order_id INT,
    FOREIGN KEY (Order_id) REFERENCES Orders(Order_id)
);
```

Result Grid | Filter Rows: | Edit:

Bill_id	Pay_status	Bill_date	Order_total_price	Order_id
NULL	NULL	NULL	NULL	NULL

```
CREATE TABLE Final_Bill_Detail (
    Order_tatal_price DECIMAL(10, 2),
    Offer_discount DECIMAL(10, 2),
    Bill_total_price DECIMAL(10, 2),
    Bill_id INT,
    FOREIGN KEY (Bill_id) REFERENCES Bill_Detail(Bill_id)
);
```

Result Grid | Filter Rows: | Export:

Order_tatal_price	Offer_discount	Bill_total_price	Bill_id

```
CREATE TABLE Product (
    Product_id INT PRIMARY KEY,
```

```
    Product_name VARCHAR(100),
    Product_qty INT,
    Product_colour VARCHAR(50),
    Product_price DECIMAL(10, 2)
);
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

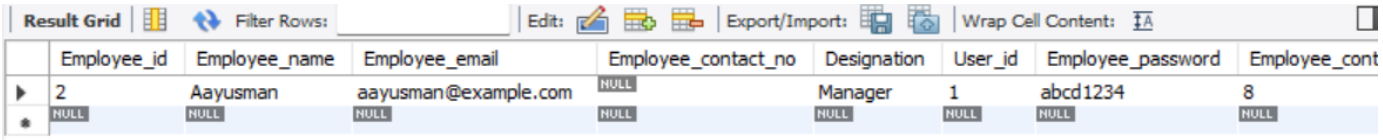
Product_id	Product_name	Product_qty	Product_colour	Product_price	Product_image	Product_stock
1	iPhone X	45	White	999.99	/static/images/products/smartphone.jpeg	100
2	Macbook Pro	28	Silver	1299.99	/static/images/products/laptop.jpeg	100
3	AirPods Pro	99	White	149.99	/static/images/products/earbuds.jpeg	100
4	Apple Watch Series 10	44	Black	299.99	/static/images/products/smartwatch.jpeg	100
5	PS5	24	White	499.99	/static/images/products/console.jpeg	100
6	Homepod	59	White	79.99	/static/images/products/speaker.jpeg	100
7	iPad Air	40	Blue	699.99	/static/images/products/tablet.jpeg	100
8	Canon	20	Black	799.99	/static/images/products/camera.jpeg	100
9	Apple Studio Display	75	White	349.99	/static/images/products/monitor.jpeg	100
10	Magic Mouse	120	Black	49.99	/static/images/products/mouse.jpeg	100
11	Magic Keyboard	50	White	129.99	/static/images/products/keyboard.jpeg	100
12	Samsung External SS...	35	Blue	159.99	/static/images/products/ssd.jpeg	100
NULL	NULL	NULL	NULL	NULL	NULL	NULL

```
CREATE TABLE User (
    User_id INT PRIMARY KEY,
    Username VARCHAR(100) UNIQUE NOT NULL,
    Password_hash VARCHAR(255) NOT NULL,
    Role VARCHAR(50) NOT NULL -- e.g., 'Customer', 'Employee'
);
```

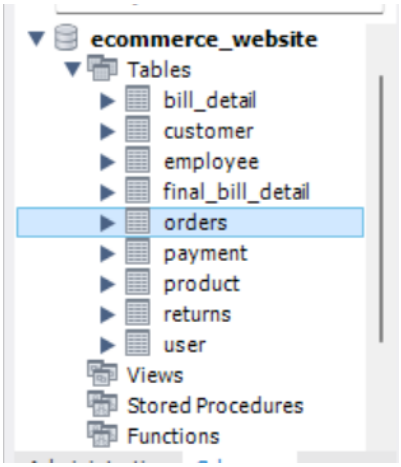
Result Grid | Filter Rows: | Edit:

User_id	Username	Password_hash	Role
1	Aayusman	abcd1234	Manager
NULL	NULL	NULL	NULL

```
CREATE TABLE Employee (
    Employee_id INT PRIMARY KEY,
    Employee_name VARCHAR(100),
    Employee_email VARCHAR(100),
    Employee_contact_no VARCHAR(15),
    Designation VARCHAR(100),
    User_id INT UNIQUE,
    FOREIGN KEY (User_id) REFERENCES User(User_id)
);
```



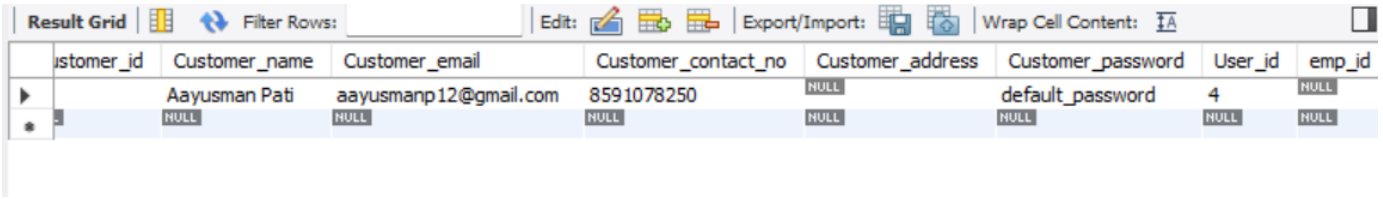
Employee_id	Employee_name	Employee_email	Employee_contact_no	Designation	User_id	Employee_password	Employee_cont
2	Aayusman	aayusman@example.com	NULL	Manager	1	abcd1234	8
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL



ALTER TABLE Customers ADD COLUMN emp_id INT;

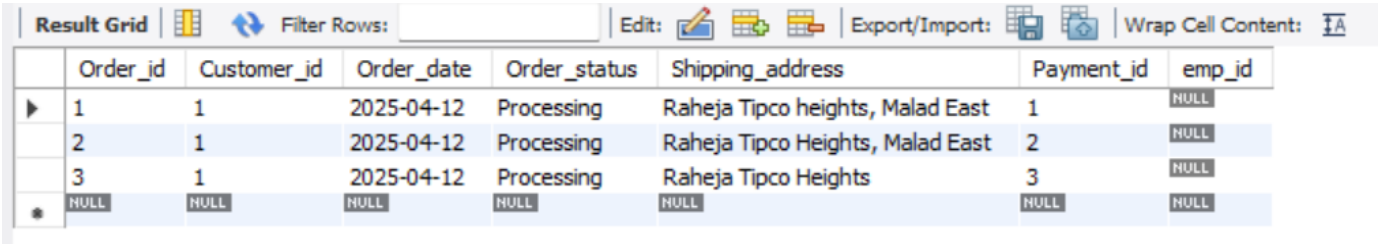
ALTER TABLE Customers ADD CONSTRAINT fk_customers_emp

FOREIGN KEY (emp_id) REFERENCES Employees(emp_id);



ustomer_id	Customer_name	Customer_email	Customer_contact_no	Customer_address	Customer_password	User_id	emp_id
	Aayusman Pati	aayusmanp12@gmail.com	8591078250	NULL	default_password	4	NULL
	NULL	NULL	NULL	NULL	NULL	NULL	NULL

ALTER TABLE orders ADD COLUMN emp_id INT;
ALTER TABLE orders ADD CONSTRAINT fk_orders_emp
FOREIGN KEY (emp_id) REFERENCES employee(Employee_id);



Order_id	Customer_id	Order_date	Order_status	Shipping_address	Payment_id	emp_id
1	1	2025-04-12	Processing	Raheja Tipco heights, Malad East	1	NULL
2	1	2025-04-12	Processing	Raheja Tipco Heights, Malad East	2	NULL
3	1	2025-04-12	Processing	Raheja Tipco Heights	3	NULL
NULL	NULL	NULL	NULL	NULL	NULL	NULL

ALTER TABLE returns ADD COLUMN emp_id INT;
ALTER TABLE returns ADD CONSTRAINT fk_returns_emp
FOREIGN KEY (emp_id) REFERENCES employee(Employee_id);

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

Return_id	Order_id	Product_id	Return_date	Return_reason	Return_status	Return_amount	emp_id
1	2	1	2025-04-12	not satisfied with the product	Rejected	999.99	NULL
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

```
ALTER TABLE bill_detail ADD COLUMN emp_id INT;
ALTER TABLE bill_detail ADD CONSTRAINT fk_bill_detail_emp
   FOREIGN KEY (emp_id) REFERENCES employee(Employee_id);
```

Result Grid | Filter Rows: | Edit: | Export/Im

Bill_id	Pay_status	Bill_date	Order_total_price	Order_id	emp_id
NULL	NULL	NULL	NULL	NULL	NULL

```
ALTER TABLE final_bill_detail ADD COLUMN emp_id INT;
ALTER TABLE final_bill_detail ADD  CONSTRAINT fk_final_bill_detail_emp
   FOREIGN KEY (emp_id) REFERENCES employee(Employee_id);
```

Result Grid | Filter Rows: | Edit:

Bill_id	Order_id	Pay_id	Total_amt	emp_id
6	1	1	79.99	NULL
7	2	2	2299.98	NULL
8	3	3	999.99	NULL
NULL	NULL	NULL	NULL	NULL

```
ALTER TABLE payment ADD COLUMN emp_id INT;
ALTER TABLE payment ADD CONSTRAINT fk_payment_emp
   FOREIGN KEY (emp_id) REFERENCES employee(Employee_id);
```

Result Grid | Filter Rows: | Edit: | Export/Import:

Pay_id	Pay_mode	Pay_amt	Pay_desc	Pay_date	emp_id
1	UPI	79.99	Order payment	2025-04-12	NULL
2	Net Banking	2299.98	Order payment	2025-04-12	NULL
3	Net Banking	999.99	Order payment	2025-04-12	NULL
NULL	NULL	NULL	NULL	NULL	NULL

```
ALTER TABLE product ADD COLUMN emp_id INT;
ALTER TABLE product ADD CONSTRAINT fk_product_emp
   FOREIGN KEY (emp_id) REFERENCES Employee(Employee_id)
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

Product_id	Product_name	Product_image	Product_qty	Product_colour	Product_price	emp_id
1	Smartphone X Pro	smartphone.jpeg	45	Black	999.99	NULL
2	Laptop Ultra	laptop.jpeg	28	Silver	1299.99	NULL
3	Wireless Earbuds	earbuds.jpeg	99	White	149.99	NULL
4	Smart Watch	smartwatch.jpeg	44	Black	299.99	NULL
5	Gaming Console	console.jpeg	24	Black	499.99	NULL

product 14

INSERT INTO employee (Employee_id, Employee_name, Employee_email, Employee_contact_no, Designation, User_id)
VALUES (4, 'Raj', 'rvrr1107@gmail.com', 8850004042, 'Clerk', 4);

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

Employee_id	Employee_name	Employee_email	Employee_contact_no	Designation	User_id
1	Admin User	admin@example.com	1234567890	Administrator	1
2	Aayusman	aayusman@example.com	9876543210	Manager	2
3	Yug	yug@example.com	9876543211	Supervisor	3
4	Raj	rvrr1107@gmail.com	8850004042	Clerk	4
NULL	NULL	NULL	NULL	NULL	NULL

INSERT INTO user (User_id, Username, Password_hash, Role)
VALUES
(5, 'rvrr1107@gmail.com', 'abcd1234', 'Employee');

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

User_id	Username	Password_hash	Role
1	admin@example.com	admin123	Employee
2	aayusman@example.com	scrypt:32768:8:1$fCqNQZl8j0uSfNla$eb8f65b7...	Employee
3	yug@example.com	scrypt:32768:8:1$HBwOKoz7bqYm8Kiq$96de5d...	Employee
4	aayusmanp12@gmail.com	yug@example.com /KGMpSlXBiZJaZD$9cb5cd3...	Customer
5	rvrr1107@gmail.com	abcd1234	Employee

INSERT INTO Payment (Pay_id, Pay_mode, Pay_amt, Pay_desc, Pay_date, emp_id)
VALUES
(4, 'Credit Card', 1500.00, 'Order payment', '2025-04-13', NULL),
(5, 'Debit Card', 250.75, 'Order payment', '2025-04-13', NULL),
(6, 'UPI', 349.50, 'Order payment', '2025-04-13', NULL),
(7, 'Net Banking', 1875.00, 'Order payment', '2025-04-13', NULL),
(8, 'UPI', 499.99, 'Order payment', '2025-04-13', NULL);

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap C

Pay_id	Pay_mode	Pay_amt	Pay_desc	Pay_date	emp_id
1	UPI	79.99	Order payment	2025-04-12	NULL
2	Net Banking	2299.98	Order payment	2025-04-12	NULL
3	Net Banking	999.99	Order payment	2025-04-12	NULL
4	Credit Card	1500.00	Order payment	2025-04-13	NULL
5	Debit Card	250.75	Order payment	2025-04-13	NULL
6	UPI	349.50	Order payment	2025-04-13	NULL
7	Net Banking	1875.00	Order payment	2025-04-13	NULL
8	UPI	499.99	Order payment	2025-04-13	NULL
NULL	NULL	NULL	NULL	NULL	NULL

INSERT INTO returns (Return_id, Order_id, Product_id, Return_date, Return_reason, Return_status, Return_amount, emp_id)
VALUES
(2, 3, 2, '2025-04-13', 'Damaged item received', 'Approved', 2299.98, NULL),
(3, 1, 3, '2025-04-13', 'Wrong item delivered', 'Pending', 79.99, NULL);

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

Return_id	Order_id	Product_id	Return_date	Return_reason	Return_status	Return_amount	emp_id
1	2	1	2025-04-12	not satisfied with the product	Rejected	999.99	NULL
2	3	2	2025-04-13	Damaged item received	Approved	2299.98	NULL
3	1	3	2025-04-13	Wrong item delivered	Pending	79.99	NULL
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

INSERT INTO bill_detail (Bill_id, Bill_date, Order_total_price)
VALUES (1, '2023-01-01', 100.00);

UPDATE customer
SET total_purchases = total_purchases + 100.00
WHERE customer_id = 1;

COMMIT;

Result Grid | Filter Rows: | Edit: | Export/Import:

Bill_id	Pay_status	Bill_date	Order_total_price	Order_id	emp_id
1	NULL	2023-01-01	100.00	NULL	NULL
NULL	NULL	NULL	NULL	NULL	NULL

SELECT SUM(Pay_amt) AS total_sales
FROM payment;

Result Grid

	total_sales
▶	7855.20

```
SELECT Pay_id, SUM(Pay_amt) AS total_sales
FROM payment
GROUP BY Pay_id;
```

Result Grid | Filter Rows:

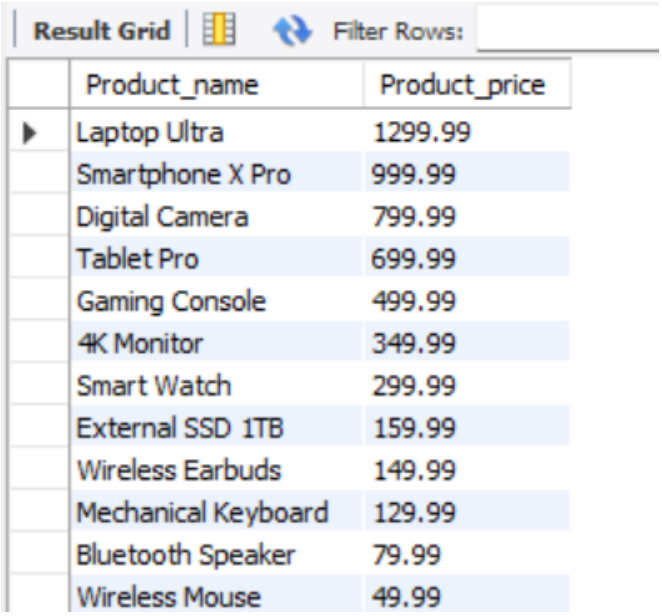
	Pay_id	total_sales
▶	1	79.99
	2	2299.98
	3	999.99
	4	1500.00
	5	250.75
	6	349.50
	7	1875.00
	8	499.99

```
SELECT c.Customer_name
FROM customer c
WHERE c.Customer_id IN (
   SELECT o.Customer_id
   FROM orders o
   JOIN final_bill_detail fbd ON o.Order_id = fbd.Order_id
   WHERE fbd.Total_amt > (
      SELECT AVG(fbd2.Total_amt)
      FROM final_bill_detail fbd2
   )
);
```

Result Grid | Filter Row

	Customer_name
▶	Aayusman Pati

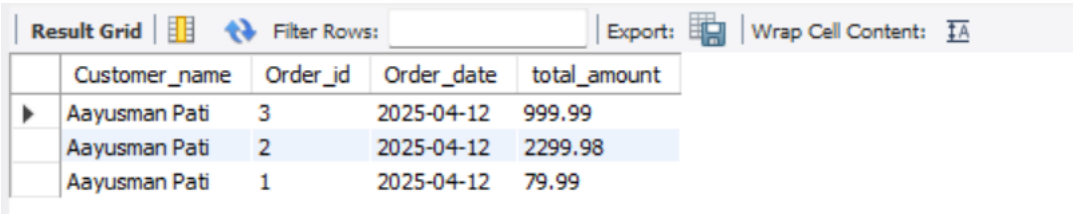
```
SELECT Product_name, Product_price
FROM product
ORDER BY Product_price DESC;
```



Result Grid | Filter Rows:

Product_name	Product_price
Laptop Ultra	1299.99
Smartphone X Pro	999.99
Digital Camera	799.99
Tablet Pro	699.99
Gaming Console	499.99
4K Monitor	349.99
Smart Watch	299.99
External SSD 1TB	159.99
Wireless Earbuds	149.99
Mechanical Keyboard	129.99
Bluetooth Speaker	79.99
Wireless Mouse	49.99

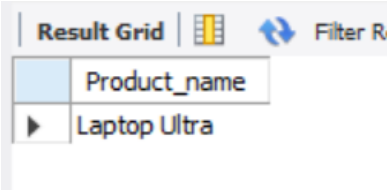
```
SELECT c.Customer_name, o.Order_id, o.Order_date, fbd.Total_amt AS total_amount
FROM customer c
JOIN orders o ON c.Customer_id = o.Customer_id
JOIN final_bill_detail fbd ON o.Order_id = fbd.Order_id;
```



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Customer_name	Order_id	Order_date	total_amount
Aayusman Pati	3	2025-04-12	999.99
Aayusman Pati	2	2025-04-12	2299.98
Aayusman Pati	1	2025-04-12	79.99

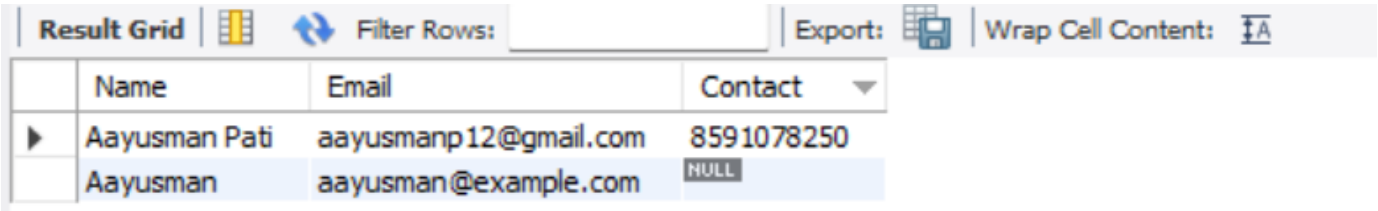
```
SELECT Product_name
FROM product
WHERE Product_name LIKE 'L%';
```



Result Grid | Filter R

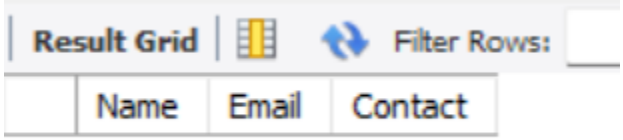
Product_name
Laptop Ultra

```
SELECT Customer_name AS Name, Customer_email AS Email, Customer_contact_no AS Contact
FROM Customer
UNION
SELECT Employee_name AS Name, Employee_email AS Email, Employee_contact_no AS Contact
FROM employee;
```


Result Grid | Filter Rows: | Export: | Wrap Cell Content:

	Name	Email	Contact
▶	Aayusman Pati	aayusmanp12@gmail.com	8591078250
	Aayusman	aayusman@example.com	NULL

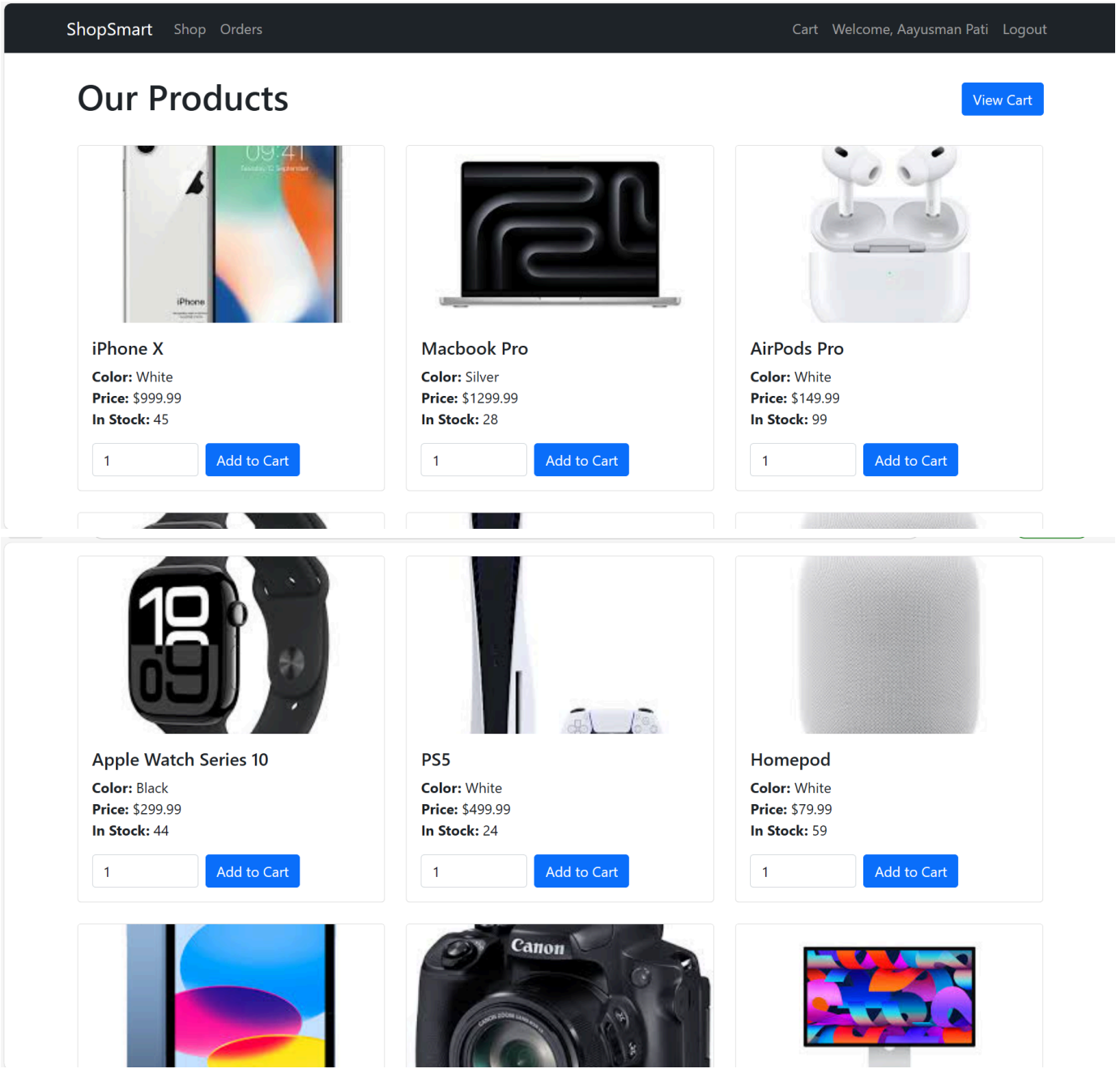
```
SELECT c.Customer_name AS Name, c.Customer_email AS Email, c.Customer_contact_no AS Contact
FROM Customer c
JOIN Employee e
  ON c.Customer_name = e.Employee_name
 AND c.Customer_email = e.Employee_email
 AND c.Customer_contact_no = e.Employee_contact_no;
```


Result Grid | Filter Rows:

	Name	Email	Contact

VII. Project demonstration

- Tools/software/ libraries used
- Screenshot and Description of the Demonstration of project (If GUI is made)



iPad Air
Color: Blue
Price: $699.99
In Stock: 40
1 Add to Cart

Canon
Color: Black
Price: $799.99
In Stock: 20
1 Add to Cart

Apple Studio Display
Color: White
Price: $349.99
In Stock: 75
1 Add to Cart

Magic Mouse
Color: Black
Price: $49.99
In Stock: 120
1 Add to Cart

Magic Keyboard
Color: White
Price: $129.99
In Stock: 50
1 Add to Cart

Samsung External SSD 1TB
Color: Blue
Price: $159.99
In Stock: 35
1 Add to Cart

ShopSmart Login Register

Login to ShopSmart

Login As
Customer

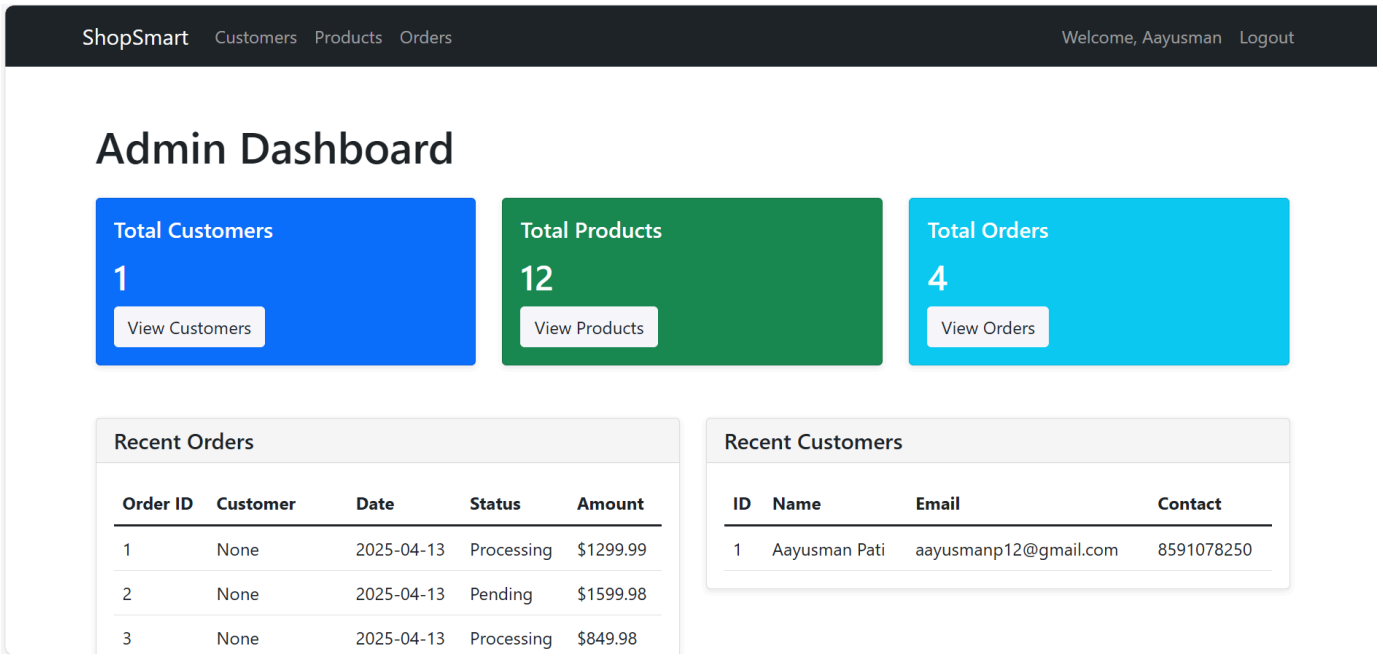
Email
aayusmanp12@gmail.com

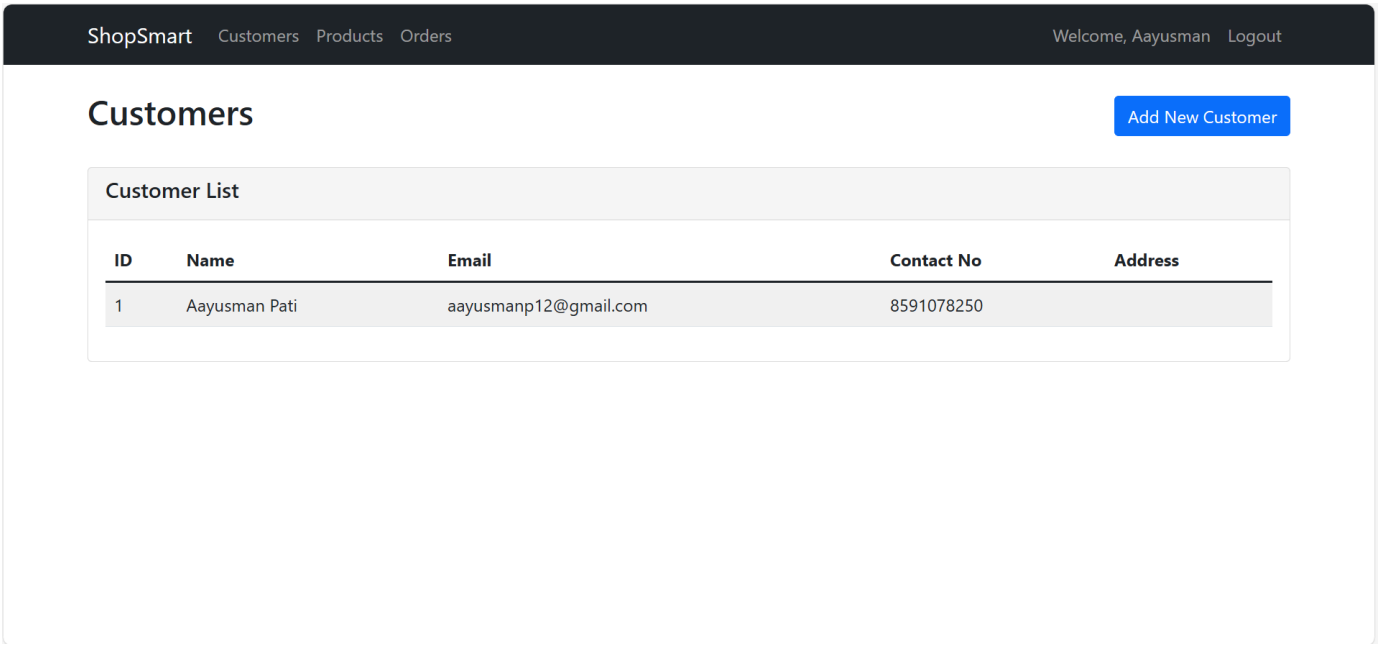
Password
••••••••

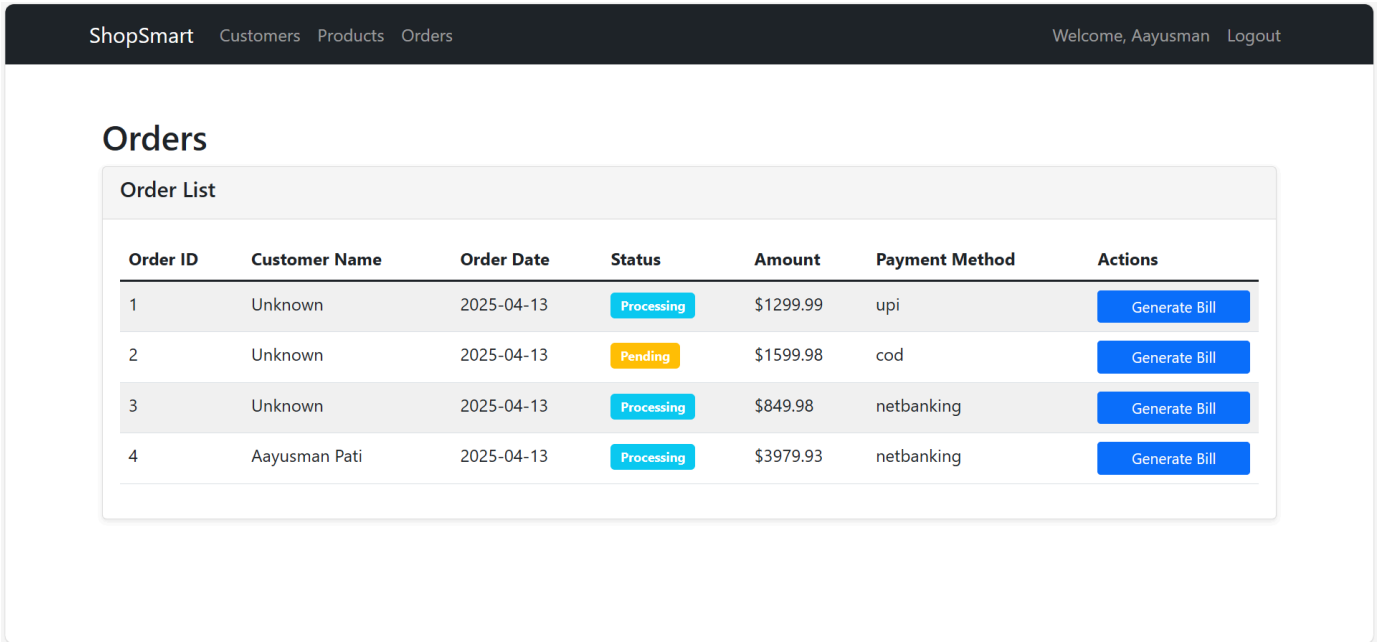
Login

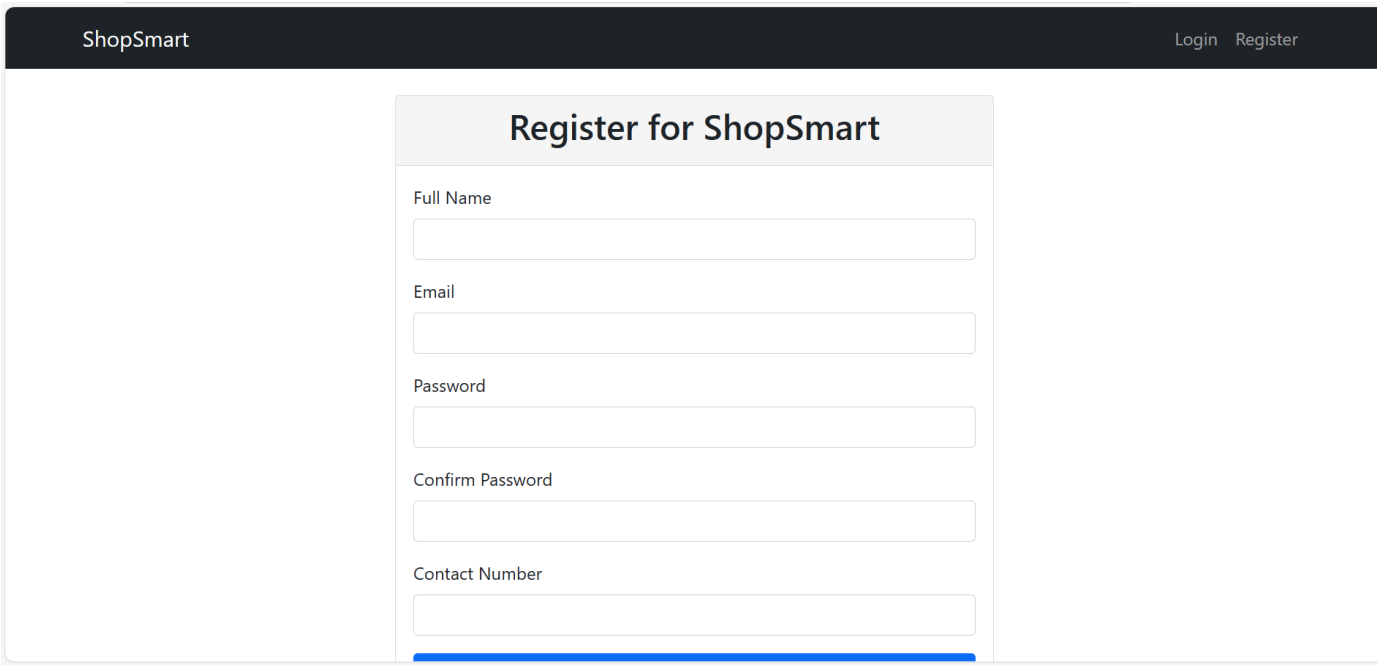
Don't have an account? Register here

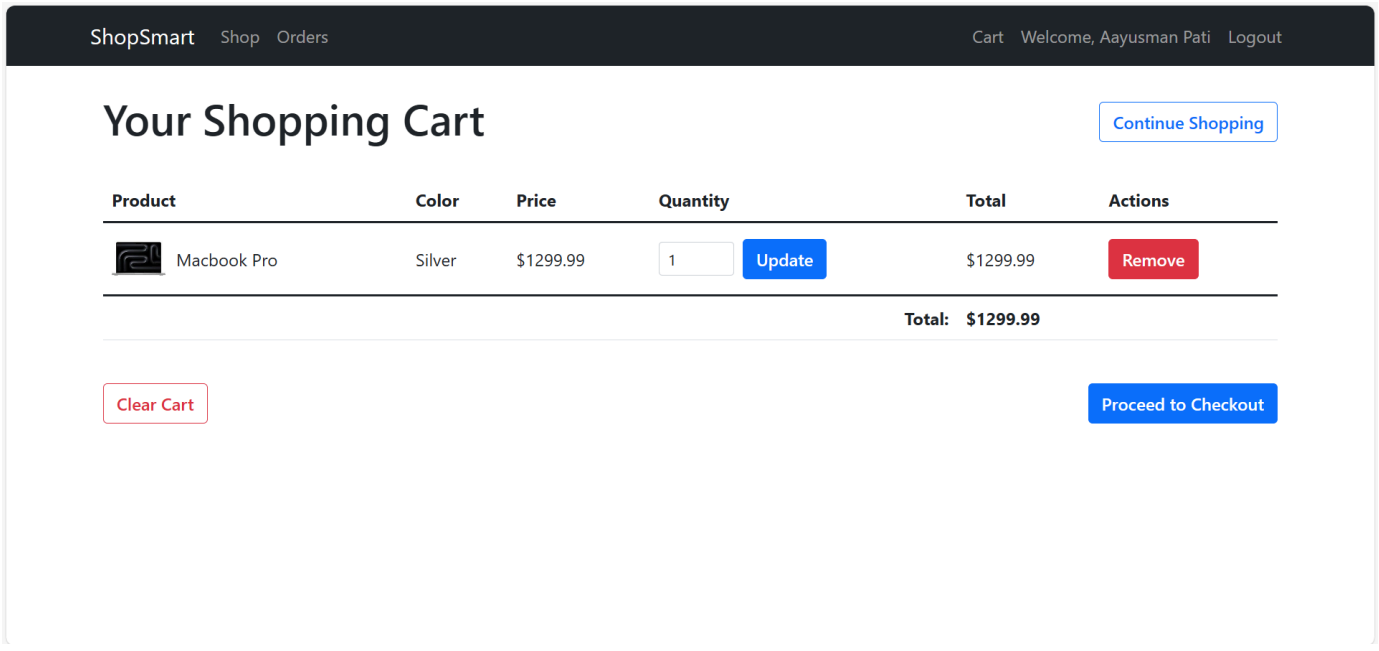
127.0.0.1:8000/login

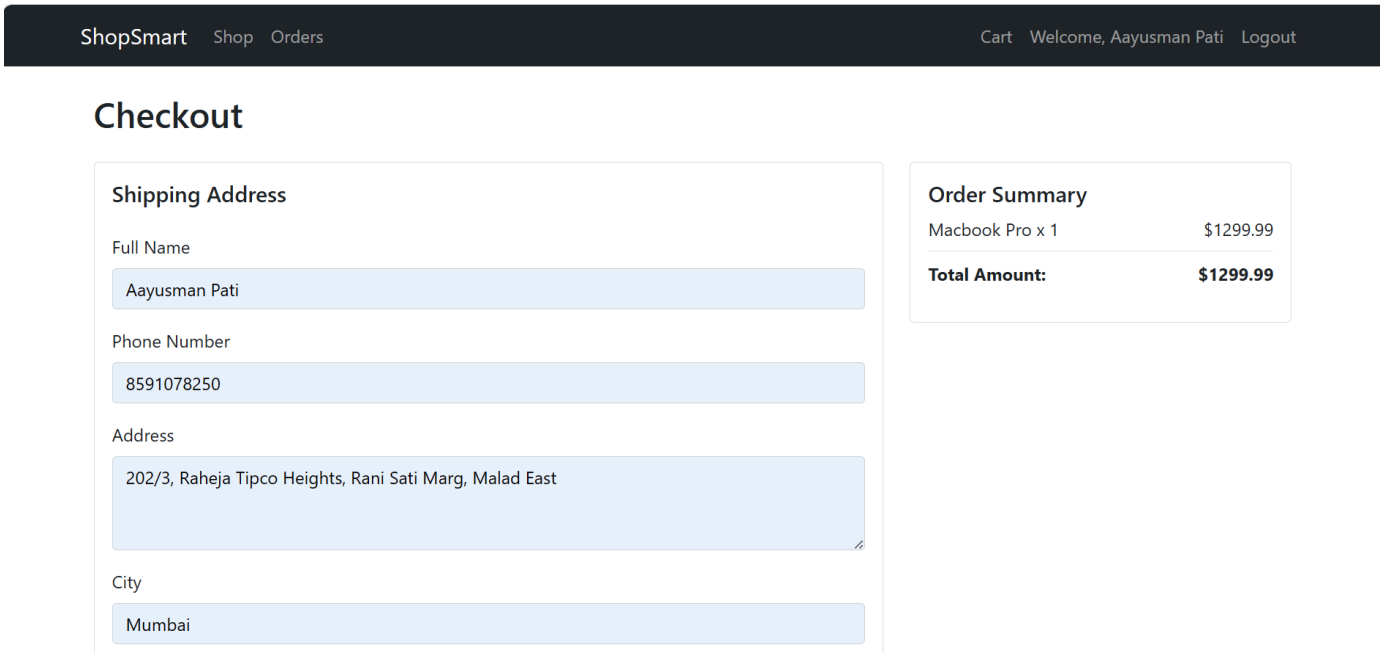

ShopSmart
Customers
Products
Orders
Welcome, Aayusman
Logout
Admin Dashboard
Total Customers
1
View Customers
Total Products
12
View Products
Total Orders
4
View Orders
Recent Orders
Order ID
Customer
Date
Status
Amount
1
None
2025-04-13
Processing
$1299.99
2
None
2025-04-13
Pending
$1599.98
3
None
2025-04-13
Processing
$849.98
Recent Customers
ID
Name
Email
Contact
1
Aayusman Pati
aayusmanp12@gmail.com
8591078250


ShopSmart
Customers
Products
Orders
Welcome, Aayusman
Logout
Customers
Add New Customer
Customer List
ID
Name
Email
Contact No
Address
1
Aayusman Pati
aayusmanp12@gmail.com
8591078250


ShopSmart
Customers
Products
Orders
Welcome, Aayusman
Logout
Orders
Order List
Order ID
Customer Name
Order Date
Status
Amount
Payment Method
Actions
1
Unknown
2025-04-13
Processing
$1299.99
upi
Generate Bill
2
Unknown
2025-04-13
Pending
$1599.98
cod
Generate Bill
3
Unknown
2025-04-13
Processing
$849.98
netbanking
Generate Bill
4
Aayusman Pati
2025-04-13
Processing
$3979.93
netbanking
Generate Bill


ShopSmart
Login
Register
Register for ShopSmart
Full Name
Email
Password
Confirm Password
Contact Number


ShopSmart
Shop
Orders
Cart
Welcome, Aayusman Pati
Logout
Your Shopping Cart
Continue Shopping
Product
Color
Price
Quantity
Total
Actions
Macbook Pro
Silver
$1299.99
1
Update
$1299.99
Remove
Total:
$1299.99
Clear Cart
Proceed to Checkout


ShopSmart
Shop
Orders
Cart
Welcome, Aayusman Pati
Logout
Checkout
Shipping Address
Full Name
Aayusman Pati
Phone Number
8591078250
Address
202/3, Raheja Tipco Heights, Rani Sati Marg, Malad East
City
Mumbai
Order Summary
Macbook Pro x 1
$1299.99
Total Amount:
$1299.99

8591078250

Address

202/3, Raheja Tipco Heights, Rani Sati Marg, Malad East

City

Mumbai

State

Maharashtra

PIN Code

400097

Payment Method

- UPI Payment
- Cash on Delivery (COD)
- Net Banking

Back to Cart

Place Order

ShopSmart Shop Orders Cart Welcome, Aayusman Pati Logout

Orders

Order List

Order ID	Customer Name	Order Date	Status	Amount	Payment Method	Actions
4	Aayusman Pati	2025-04-13	Processing	$3979.93	netbanking	Generate Bill
5	Aayusman Pati	2025-04-13	Pending	$1299.99	cod	Generate Bill

VIII. Self -Learning beyond classroom

:

- What new aspects did you learn on your own? You have to mention learning beyond the classroom

Through this project, we had the opportunity to learn some things that are not included in the general classroom curriculum. Some of these are:

Frontend-Backend Connectivity:
We explored how to establish a seamless connection between the user interface and the backend database, enabling dynamic interaction within a web application. Through this process, we gained practical insight into how real world websites perform essential operations such as adding items to a shopping cart, updating product quantities, and processing orders. Understanding this interaction helped us understand the underlying logic and data flow that supports responsive and user driven functionality on modern platforms.

Troubleshooting and Debugging Skills:
While testing various features, we encountered several bugs, logical inconsistencies, and unexpected behaviors. Addressing these issues required a methodical approach to debugging, analyzing the code, identifying root causes, and implementing effective solutions. This hands on experience strengthened our troubleshooting skills.

IX. Learning from the Project

Include learning from the project:

- How this project helped you?

The project provided us with a clear understanding of how concepts in a database are used in actual systems. It facilitated linking learning in the classroom to actual practice and enabled us to learn a lot of new things in the process. Some of the most important learnings are:

Real-world use case understanding:
We learned how online shopping platforms handle complex data like customer orders, payments, and returns. This enabled us to understand how databases work in actual businesses.

Query construction:
We constructed queries involving multiple tables and conditions. Constructing queries that would answer particular business-related questions assisted in improving our approach towards solving problems.

Schema evolution and adaptability:
As we made progress, we realized the need to make changes in the schema. This taught us how to make databases flexible and future-proof.

User roles and access control:
By incorporating various user types such as customers and employees, we understood how role-based access functions and why it is essential in multi-user systems.

Debugging and error resolution:
During the process, we encountered problems such as wrong outputs, query errors, or data mismatches. Correcting them assisted us in sharpening our debugging skills and concentration.

X. Challenges Faced

We faced a number of difficulties while working on this DBMS project that put our knowledge of database concepts and project development to the test. The primary challenges we faced are as follows:

Creating the Database Schema: At first, it was challenging to determine the required entities and accurately define the relationships. To prevent duplication and guarantee data consistency, it needed to be revised several times.

Making the ER Diagram: It was difficult to turn our concept into an accurate ER diagram. When mapping real-world activities like placing an order and returning a product, we had to make sure that the proper cardinalities and participation constraints were maintained.

Normalisation Complexity: At first, it was difficult to apply the normalisation principles.. Making sure our tables were correctly normalised up to 3NF needed more work.

Backend Integration: One of the biggest challenges was integrating the MySQL database with the front-end interface so that dynamic tasks like order placement and real-time cart item changes could be carried out.

XI. Conclusion

- What are the key takeaways from the project?

Through this project, we successfully bridged the gap between classroom knowledge and real-world application by designing a comprehensive database system for an online marketplace. We transitioned from conceptual ER models to their practical implementation using real SQL queries, allowing us to solidify our understanding and enhance our SQL proficiency. This hands-on experience enabled us to confidently write complex queries and understand the critical role of constraints and normalization in maintaining data integrity and eliminating redundancy. Additionally, the project exposed us to real-world technical challenges, such as team collaboration and system integration issues, which ultimately strengthened both our technical skills and project management capabilities.