CHANDIGARH UNIVERSITY

Discover. Learn. Empower.

Chandigarh University

University Institute of Computing

Report

On

E-COMMERCE SYSTEM

SUBMITTED BY: RIDHI

UID: 24BCD10027

SUBMITTED TO: Mr. SURAJ PRAKASH

SUBJECT: DATABASE MANAGEMENT SYSTEM

SUBJECT-CODE:24CAP-204

ACKNOWEDGMENT

We deem it a great pleasure to express our heartfelt gratitude to our project guide, **Mr. Suraj**, under whose constant guidance, encouragement, and insightful suggestions we were able to carry out our project work successfully. His incisive observations, objective feedback, and timely advice provided us with the motivation and clarity we needed at every stage of the project. His unwavering support instilled in us a continuous flow of energy and enthusiasm to strive for excellence.

We also extend our sincere thanks to **Dr. Kavita Gupta (H.O.D., University Institute of Computing)**, whose valuable insights and constructive suggestions played a crucial role in shaping the direction and quality of our work. Her constant inspiration and supportive demeanour created a nurturing environment that fuelled our academic endeavours.

Our deepest gratitude goes to **Dr. Manisha Malhotra, Director, University Institute of Technology**, for fostering a strong academic culture, and for providing us with a disciplined and intellectually stimulating atmosphere. Her leadership and commitment to academic rigor encouraged us to approach the project with utmost seriousness, dedication, and professionalism.

We are also immensely thankful to all the faculty members and staff of the University Institute of Computing for their kind cooperation and constant encouragement throughout the course of our project.

Finally, we would be remiss if we did not acknowledge the unconditional support of our **parents and friends**. No achievement is possible without sacrifices and moral strength provided by loved ones. Their patience, care, and unwavering belief in us were the pillars upon which we built our efforts. To them, we owe our deepest gratitude and love.

ABSTRACT

1. Aim:

The aim of this DBMS project is to design and develop a structured database system for managing various aspects of an online shopping platform, such as product details, categories, customers, orders, and payments.
The database ensures accurate, fast, and secure data storage and retrieval—supporting all operations that occur in a typical e-commerce environment.

This project also aims to explore the principles of relational database design, normalization, and query operations to ensure efficiency and data integrity.

2. Objective:

The objectives of this project are:

1. To create a relational database that stores and manages e-commerce information systematically.
2. To identify entities and relationships in an online shopping scenario and represent them using an E-R diagram.
3. To normalize the data up to Third Normal Form (3NF) to remove redundancy.
4. To implement SQL operations (DDL, DML, DQL) to manage and query the database.
5. To apply database constraints such as PRIMARY KEY, FOREIGN KEY, UNIQUE, and CHECK to maintain integrity.
6. To perform queries using JOIN, GROUP BY, HAVING, and VIEWS for meaningful business insights.

3. Introduction:

An E-Commerce Product Management System is a database-based application designed to maintain the complete life cycle of an online shopping system.

The database acts as the core of the system, managing data related to products, customers, orders, and transactions.
The project uses SQL as the primary query language and MySQL / Oracle as the database management tool.

Key Aspects:

- Efficient product storage and classification
- Easy order tracking and invoice generation
- Real-time payment record management
- Customer information maintenance
- Simplified querying and reporting

A properly designed database helps the e-commerce business to improve speed, data consistency, and analytical capabilities, while also supporting scalability and future growth.

PHASE 1 – PROBLEM IDENTIFICATION

1.1 Problem Definition

E-commerce companies handle large volumes of data daily — including customer information, order details, payments, and product inventory. Without a structured database, problems arise such as:

- Data duplication and inconsistency
- Delayed response time in searches and updates
- Difficulty in generating sales reports

- Lack of secure payment and transaction records

To solve these issues, this project proposes a relational database model that systematically handles all the above elements.

1.2 Identification of Entities and Relationships

The main entities involved in this system are:

Entity Name	Attributes	Description
Customer	Customer_ID, Name, Email, Phone, Address, City	Stores registered customer details
Category	Category_ID, Category_Name, Description	Classifies products into various categories
Product	Product_ID, Name, Category_ID, Price, Stock	Contains all information about available products
Orders	Order_ID, Customer_ID, Order_Date, Total_Amount, Status	Represents each purchase transaction
Order_Details	Detail_ID, Order_ID, Product_ID, Quantity, Subtotal	Holds detailed order information (multi-item orders)
Payment	Payment_ID, Order_ID, Payment_Mode, Payment_Status, Payment_Date	Maintains payment transaction details

1.3 Relationships Among Entities

1. Customer → Orders: One-to-Many
    - A single customer can place multiple orders.
2. Orders → Payment: One-to-One
    - Each order has one payment record.
3. Category → Product: One-to-Many
    - Each category may contain several products.
4. Orders → Order_Details → Product: Many-to-Many (resolved using Order_Details)
    - An order can contain multiple products, and a product can belong to multiple orders.

1.4 E-R Diagram:

Description of E-R Diagram:

- Customer entity connects to Orders via a 1-to-many relationship.
- Orders are connected to Payment through a 1-to-1 link.
- Product connects to Category with a 1-to-many relationship.
- Order_Details acts as a bridge between Orders and Products, resolving their many-to-many association.

1.5 Data Storage and Retrieval

- Data will be stored in normalized tables to avoid redundancy.
- Indexes will be created on primary keys for fast retrieval.
- Queries like JOIN, GROUP BY, and VIEWS will extract summarized data for reports (e.g., total sales, top customers, payment summary).

PHASE 2 – DATABASE DESIGN:

2.1 Relational Schema

Table Name	Attributes	Primary Key	Foreign Key
Customer	Customer_ID, Name, Email, Phone, Address, City	Customer_ID	—
Category	Category_ID, Category_Name, Description	Category_ID	—
Product	Product_ID, Name, Category_ID, Price, Stock	Product_ID	Category_ID
Orders	Order_ID, Customer_ID, Order_Date, Total_Amount, Status	Order_ID	Customer_ID
Order_Details	Detail_ID, Order_ID, Product_ID, Quantity, Subtotal	Detail_ID	Order_ID, Product_ID
Payment	Payment_ID, Order_ID, Payment_Mode, Payment_Status, Payment_Date	Payment_ID	Order_ID

2.2 Conversion from ER to Relational Model

Each strong entity (like Customer, Product, Category) becomes an independent table.

Relationships (like Order-Product) are represented through linking tables such as Order_Details.

2.3 Normalization:

First Normal Form (1NF):

- All fields hold atomic values (no repeating groups).

Second Normal Form (2NF):

- All non-key attributes depend fully on the primary key.

Third Normal Form (3NF):

- No transitive dependencies: all attributes depend only on the primary key.

After normalization, the database maintains minimal redundancy and maximum consistency.

2.4 Constraints Applied:

1. PRIMARY KEY: Ensures unique identification of each record.
2. FOREIGN KEY: Enforces referential integrity across tables.
3. NOT NULL: Ensures mandatory fields are filled.
4. UNIQUE: Prevents duplicate email IDs or order numbers.
5. CHECK: Ensures logical limits, such as price > 0, quantity > 0.
6. DEFAULT: Assigns default status like “Pending” for new orders.

PHASE 3 – IMPLEMENTATION AND QUERIES

3.1 DDL Commands

```sql
CREATE TABLE Customer (
    Customer_ID INT PRIMARY KEY,
    Name VARCHAR(50) NOT NULL,
    Email VARCHAR(100) UNIQUE,
    Phone VARCHAR(15),
    Address VARCHAR(150),
    City VARCHAR(50)
);

CREATE TABLE Category (
    Category_ID INT PRIMARY KEY,
    Category_Name VARCHAR(50),
    Description VARCHAR(100)
);
```

```sql
CREATE TABLE Product (
    Product_ID INT PRIMARY KEY,
    Name VARCHAR(50),
    Category_ID INT,
    Price DECIMAL(10,2) CHECK(Price > 0),
    Stock INT DEFAULT 0,
    FOREIGN KEY(Category_ID) REFERENCES Category(Category_ID)
);

CREATE TABLE Orders (
    Order_ID INT PRIMARY KEY,
    Customer_ID INT,
    Order_Date DATE,
    Total_Amount DECIMAL(10,2),
    Status VARCHAR(30) DEFAULT 'Pending',
    FOREIGN KEY(Customer_ID) REFERENCES Customer(Customer_ID)
);
```

```sql
CREATE TABLE Order_Details (
    Detail_ID INT PRIMARY KEY,
    Order_ID INT,
    Product_ID INT,
    Quantity INT CHECK(Quantity > 0),
    Subtotal DECIMAL(10,2),
    FOREIGN KEY(Order_ID) REFERENCES Orders(Order_ID),
    FOREIGN KEY(Product_ID) REFERENCES Product(Product_ID)
);
```

```sql
CREATE TABLE Payment (
  Payment_ID INT PRIMARY KEY,
  Order_ID INT,
  Payment_Mode VARCHAR(30),
  Payment_Status VARCHAR(20),
  Payment_Date DATE,
  FOREIGN KEY(Order_ID) REFERENCES Orders(Order_ID)
);
```

3.2 DML Commands (Data Manipulation)

```sql
INSERT INTO Customer VALUES (1, 'Ridhi', 'ridhi@cu.edu', '9876543210', 'Sector 14', 'Chandigarh');
INSERT INTO Category VALUES (10, 'Electronics', 'Electronic items and gadgets');
INSERT INTO Product VALUES (101, 'Smartphone', 10, 14999.00, 50);
INSERT INTO Orders VALUES (201, 1, '2025-10-24', 14999.00, 'Delivered');
INSERT INTO Order_Details VALUES (301, 201, 101, 1, 14999.00);
INSERT INTO Payment VALUES (501, 201, 'Credit Card', 'Success', '2025-10-25');
```

3.3 DQL Queries (Data Retrieval)

1. Display all products

```sql
SELECT * FROM Product;
```

2. Display orders with customer names

```sql
SELECT c.Name, o.Order_ID, o.Total_Amount, o.Status
FROM Customer c
JOIN Orders o ON c.Customer_ID = o.Customer_ID;
```

3. Find total sales by category

```sql
SELECT ca.Category_Name, SUM(od.Subtotal) AS Total_Sales
FROM Order_Details od
JOIN Product p ON od.Product_ID = p.Product_ID
JOIN Category ca ON p.Category_ID = ca.Category_ID
GROUP BY ca.Category_Name;
```

4. Retrieve successful payments

```sql
SELECT c.Name, o.Order_ID, p.Payment_Mode, p.Payment_Status
FROM Payment p
JOIN Orders o ON p.Order_ID = o.Order_ID
JOIN Customer c ON o.Customer_ID = c.Customer_ID
WHERE p.Payment_Status = 'Success';
```

5. Create a View for Top Products

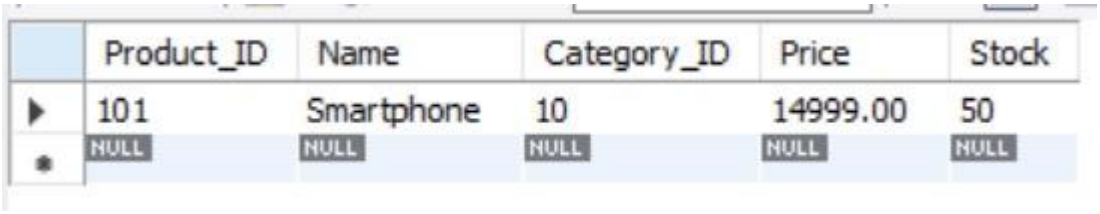
```sql
CREATE VIEW Top_Products AS
SELECT p.Name, SUM(od.Quantity) AS Total_Sold
FROM Order_Details od
JOIN Product p ON od.Product_ID = p.Product_ID
GROUP BY p.Name
ORDER BY Total_Sold DESC;

SELECT * FROM Top_Products;
```

PHASE 4 – PRESENTATION

4.1 Results and Output

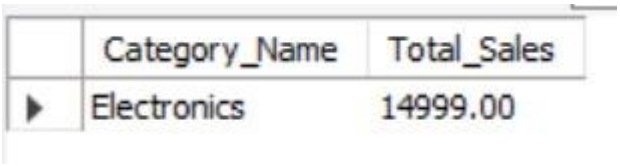
After executing the above queries:

- Product, order, and payment data were retrieved successfully.
- Grouping and aggregation queries generated total category-wise sales.
- Views simplified data access for business insights.

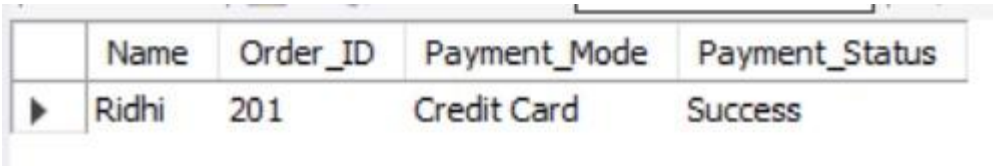


	Product_ID	Name	Category_ID	Price	Stock
▶	101	Smartphone	10	14999.00	50
*	NULL	NULL	NULL	NULL	NULL

	Name	Order_ID	Total_Amount	Status
▶	Ridhi	201	14999.00	Delivered



	Category_Name	Total_Sales
▶	Electronics	14999.00



	Name	Order_ID	Payment_Mode	Payment_Status
▶	Ridhi	201	Credit Card	Success

4.2 Design Decisions

- The relational structure was chosen for consistency and normalization.
- Primary and foreign key relationships ensure data integrity.
- Using MySQL allowed efficient query execution and easy visualization.

4.3 Learning Outcomes

1. Practical understanding of database design and normalization.
2. Ability to write SQL queries to manipulate and retrieve data.
3. Understanding the importance of relationships in maintaining consistency.
4. Awareness of constraints and views for integrity and abstraction.

4. Case Study

Consider an example where a customer, *Ridhi*, orders a *Smartphone* under the *Electronics* category.
The order is stored in the Orders table, product details in Product, and the payment record in Payment.
When a query is executed, all related details can be fetched using JOIN, showing how integrated and consistent the data remains.

5. Conclusion

The E-Commerce Product Management System successfully demonstrates the role of DBMS in organizing and managing digital marketplace data.

The project ensures:

- Accurate and efficient handling of large data volumes
- Logical data relationships using foreign keys
- Reliable transaction management
- Reduction of redundancy through normalization
- Ease of report generation using SQL queries

6. Future Enhancements

- Integration with a front-end shopping website using HTML/CSS/JavaScript.
- Addition of modules for admin analytics and dashboards.
- Implementation of stored procedures and triggers for automation.
- Enhanced security features for transactions and user data.
- Cloud deployment using MySQL on AWS or Azure for scalability.

REFERENCES

1. **Tutorials Point Database Tutorial. (2024).**
   Retrieved from: https://www.tutorialspoint.com/sql/
   → Served as a practical guide for learning database constraints and DDL/DML operations.
2. **W3Schools SQL Tutorial. (2024).**
   Retrieved from: https://www.w3schools.com/sql/
   → Provided examples of SQL queries such as JOIN, GROUP BY, and HAVING, applied during implementation.
3. **MySQL Documentation. (2024).** *MySQL 8.0 Reference Manual.* Oracle Corporation.
   Retrieved from: https://dev.mysql.com/doc/
   → Official reference for MySQL syntax, constraints, and data definition commands used in this project.