Inventory Management System

A PROJECT REPORT

Submitted by

ADITYA GUPTA[RA2211026010434]

SATYAM SINHA[RA2211026010444]

Under the Guidance of

Dr. USHARANI R

Assistant Professor , Department of Computing Technologies

in partial fulfillment of the requirements for the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

(SPL IN AI & ML)

DEPARTMENT OF COMPUTING TECHNOLOGIES
COLLEGE OF ENGINEERING AND TECHNOLOGY
SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
KATTANKULATHUR- 603203

MAY 2024

SRM
INSTITUTE OF SCIENCE & TECHNOLOGY
(Deemed to be University u/s 3 of UGC Act, 1956)

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
KATTANKULATHUR–603 203

BONAFIDE CERTIFICATE

Register no. RA2211026010434, RA2211026010444 Certified to be the bonafide work done by **ADITYA GUPTA, SATYAM SINHA** of II year/IV sem B.Tech Degree Course in the Project Course – **21CSC205P Database Management Systems** in **SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**, Kattankulathur for the academic year 2023-2024.

Date:03/05/2024

FACULTY INCHARGE
Dr Usharani R
Assistant Professor
Computing Technologies
SRM Institute of Science and Technology -KTR

HEAD OF THE DEPARTMENT
Dr. M. Pushpalatha
Professor and Head
Computing Technologies
SRM Institute of Science and Technology - KTR

TABLE OF CONTENTS

ABSTRACT

An inventory management system is a comprehensive software solution designed to help businesses effectively organize, track, and manage their inventory levels and related operations. At its core, it facilitates the monitoring of stock levels, sales, orders, and deliveries. By providing real-time visibility into inventory data, these systems enable businesses to optimize their stock levels, reduce carrying costs, prevent stockouts, and streamline their supply chain operations. One of the primary functions of an inventory management system is inventory tracking. This involves keeping a detailed record of all items in stock, including their quantities, locations, and movement history. This allows businesses to quickly locate items, monitor stock levels, and track inventory turnover rates. Additionally, inventory management systems typically include features for order management, allowing businesses to efficiently process orders, track order status, and manage customer inquiries. These systems can automate order processing tasks, such as order entry, invoicing, and order fulfillment, saving businesses time and reducing the likelihood of errors. Another key aspect of inventory management systems is purchasing management. These systems help businesses optimize their purchasing processes by providing tools for managing supplier information.

CHAPTER 1

Problem understanding, Identification of Entity and Relationships, Construction of DB using ER Model for the project

I. Problem Understanding

The inventory management system described by the provided SQL queries designed address several key challenges in managing products, orders, customers, employees, shipments, and inventory within an organization. Firstly, the system facilitates the efficient organization of products by categorizing them into different categories such as essentials, furniture, electronics, etc. This enables easy navigation and management of the product inventory .Secondly, the system handles customer management by storing essential customer information including their name, contact details, address, and login credentials.

This ensures a personalized experience for customers and allows for effective communication. Thirdly, the system manages customer orders effectively, recording order details such as order ID, order date, products ordered, quantity, and order status. This ensures accurate tracking of orders throughout the fulfillment process.Fourthly, the system manages employee information, including their name, department ,contact details, commission, and salary.

I. Identification of Entity and Relationships

Entities

1. Agent

- Attributes: agent_id (PK), agent_name

2. Category

- Attributes: cat_id (PK), category

3. Customer

- Attributes: custid (PK), c_name, phoneno, c_address, email_id, password

4. Department

- Attributes: dept_id (PK), departname

5. Employee

- Attributes: emp_id (PK), e_name, dept_id (FK), qualification, dob, e_address, e_phno, comm, salary

6. Order

- Attributes: order_id (PK), o_date, prod_count, cust_id (FK), o_status

7. Product

- Attributes: p_id (PK), p_name, quantity, catid (FK), discount, price

8. Shipment

- Attributes: sh_id (PK), s_date, s_status, agent_id (FK), emp_id (FK), est_delivery

Relationships

1. Agent-shipment Relationship

- Agent (1) ---- (0 or more) Shipment
- Foreign Key: agent_id (Agent) -> agent_id (Shipment)

2. Category-product Relationship

- Category (1)----- (0 or more) Product
- Foreign Key: cat_id (Category) -> catid (Product)

3. Customer-order Relationship

- Customer (1) -----(0 or more) Order
- Foreign Key: custid (Customer) -> cust_id (Order)

4. Department-employee Relationship

- Department (1)----- (0 or more) Employee
- Foreign Key: dept_id (Department) -> dept_id (Employee)

5. Employee-shipment Relationship

- Employee (1) -----(0 or more) Shipment

- Foreign Key: emp_id (Employee) -> emp_id (Shipment)

6. Order-product Relationship

- Order (1) -----(1 or more) Product

- Foreign Key: order_id (Order) -> order_id (Order Reference), p_id (Product) -> p_id (Order Reference)

7. Product-category Relationship

- Product (1)----- (1) Category

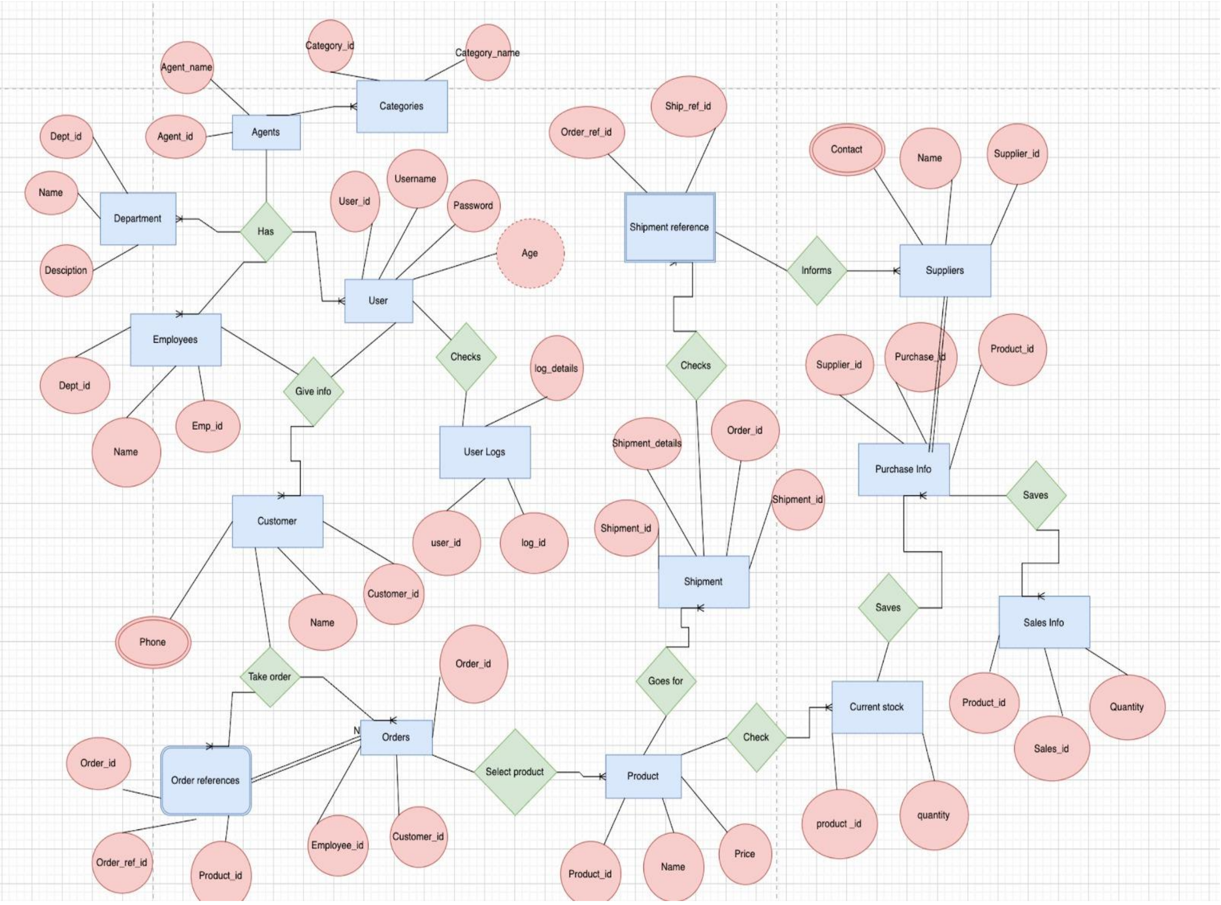
- Foreign Key: catid (Product) -> cat_id (Category)

8. Shipment-order Relationship

- Shipment (1)-----(1 or more) Order

- Foreign Key: sh_id (Shipment) -> sh_id (Shipref), order_id (Order) -> order_id (Shipref)

II. Entity and Relationships Model



In an inventory management system, the Entity-Relationship Model (ERM) defines several key entities and their relationships. The primary entities include "Product," "Supplier," "Order," and "Customer." Each product has attributes such as ProductID, Name, Description, Price, and Quantity. Suppliers provide products and have attributes like SupplierID, Name, Contact, and Address. Customers place orders, which are related to products through the OrderDetails entity, detailing attributes such as OrderID, ProductID, Quantity, and TotalPrice. This model ensures efficient management of inventory, tracking product availability, supplier information, and customer orders.

CHAPTER 2

Design of Relational Schemas, Creation of Database Tables for the project

I.Relational Schemas

1. Agent table

- agent_id (PK, varchar(5)): Primary key representing the unique ID of the agent.
- agent_name (varchar(10)): Name of the agent.

This table stores information about different agents involved in the inventory management system.

Agent table

agent_id (PK, varchar(5))

agent_name (varchar(10))

2. Category table

- cat_id (PK, char(20)): Primary key representing the unique ID of the category.
- category (char(20)): Name of the category.

This table stores different categories of products available in the inventory.

Category table:

cat_id (PK, char(20))

category (char(20))

3. Customer table

- custid (PK, char(10)): Primary key representing the unique ID of the customer.
- c_name (char(15)): Name of the customer.
- phoneno (varchar(10)): Phone number of the customer.
- c_address (char(20)): Address of the customer.
- email_id (char(100)): Email ID of the customer.
- password (char(20)): Password of the customer.

This table stores information about the customers who purchase products.

Customer table

custid (PK, char(10))

c_name (char(15))

phoneno (varchar(10))

c_address (char(20))

email_id (char(100))

password (char(20))

4. Department table

- dept_id (PK, varchar(5)): Primary key representing the unique ID of the department.
- departname (varchar(20)): Name of the department.

This table stores information about different departments in the organization.

Department table

dept_id (PK, varchar(5))

departname (varchar(20))

5. Employee table

- emp_id (PK, varchar(10)): Primary key representing the unique ID of the employee.
- e_name (varchar(15)): Name of the employee.
- dept_id (varchar(5)): Foreign key referencing the department ID.
- qualification (varchar(20)): Qualification of the employee.
- dob (date): Date of birth of the employee.
- e_address (varchar(20)): Address of the employee.
- e_phno (int(11)): Phone number of the employee.
- comm (int(3)): Commission percentage of the employee.
- salary (int(6)): Salary of the employee.

This table stores information about the employees working in different departments.

Employee table

emp_id (PK, varchar(10))

e_name (varchar(15))

dept_id (varchar(5))

qualification (varchar(20))

dob (date)

e_address (varchar(20))

e_phno (int(11))

comm (int(3))

salary (int(6))

6. Order Reference table

- order_id (varchar(10)): Foreign key referencing the order ID.
- p_id (varchar(10)): Foreign key referencing the product ID.

This table stores the reference between orders and products.

Order Reference table

order_id (varchar(10))

p_id (varchar(10))

7. Order table

- order_id (PK, varchar(10)): Primary key representing the unique ID of the order.
- o_date (date): Date of the order.
- prod_count (int(10)): Number of products in the order.
- cust_id (varchar(10)): Foreign key referencing the customer ID.
- o_status (varchar(10)): Status of the order.

This table stores information about the orders placed by customers.

Order table

order_id (PK, varchar(10))

o_date (date)

prod_count (int(10))

cust_id (varchar(10))

o_status (varchar(10))

8. Product table

- p_id (PK, char(10)): Primary key representing the unique ID of the product.
- p_name (char(10)): Name of the product.
- quantity (int(5)): Quantity of the product available in the inventory.
- catid (char(20)): Foreign key referencing the category ID.
- discount (int(3)): Discount percentage applicable to the product.
- price (int(5)): Price of the product.

This table stores information about the products available in the inventory.

Product table

p_id (PK, char(10))

p_name (char(10))

quantity (int(5))

catid (char(20))

discount (int(3))

price (int(5))

9. Shipment table

- sh_id (PK, varchar(10)): Primary key representing the unique ID of the shipment.
- s_date (date): Date of the shipment.
- s_status (varchar(10)): Status of the shipment.
- agent_id (varchar(5)): Foreign key referencing the agent ID.
- emp_id (varchar(10)): Foreign key referencing the employee ID.
- est_delivery (date): Estimated delivery date of the shipment.

This table stores information about the shipments of products.

Shipment table

sh_id (PK, varchar(10))

s_date (date)

s_status (varchar(10))

agent_id (varchar(5))

emp_id (varchar(10))

est_delivery (date)

10. Shipment Reference table

- sh_id (varchar(10)): Foreign key referencing the shipment ID.
- order_id (varchar(10)): Foreign key referencing the order ID.

This table stores the reference between shipments and orders.

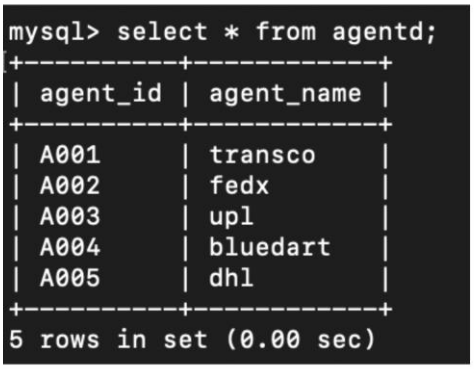
Shipment Reference table

sh_id (varchar(10))

order_id (varchar(10))

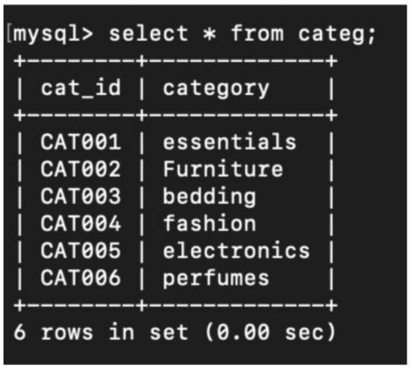
II. Database Tables

1. agentd



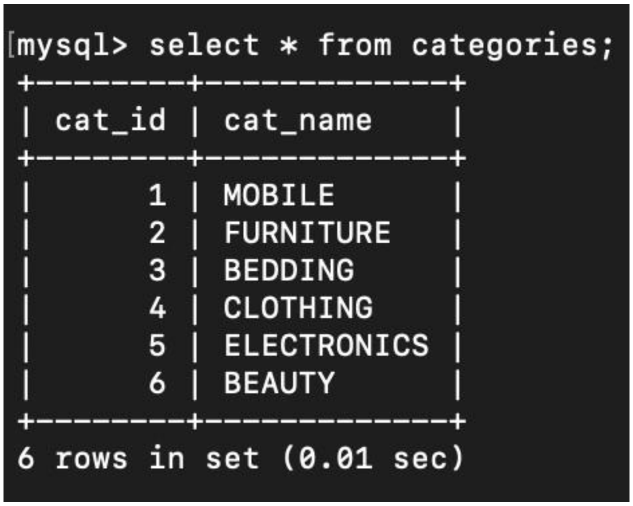
```
mysql> select * from agentd;
[+----------+------------+
| agent_id | agent_name |
+----------+------------+
| A001     | transco    |
| A002     | fedx       |
| A003     | upl        |
| A004     | bluedart   |
| A005     | dhl        |
+----------+------------+
5 rows in set (0.00 sec)
```

2.categ



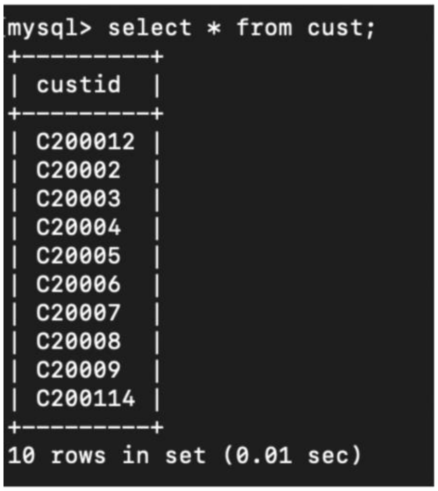
```
[mysql> select * from categ;
+--------+-------------+
| cat_id | category    |
+--------+-------------+
| CAT001 | essentials  |
| CAT002 | Furniture   |
| CAT003 | bedding     |
| CAT004 | fashion     |
| CAT005 | electronics |
| CAT006 | perfumes    |
+--------+-------------+
6 rows in set (0.00 sec)
```

3.categories



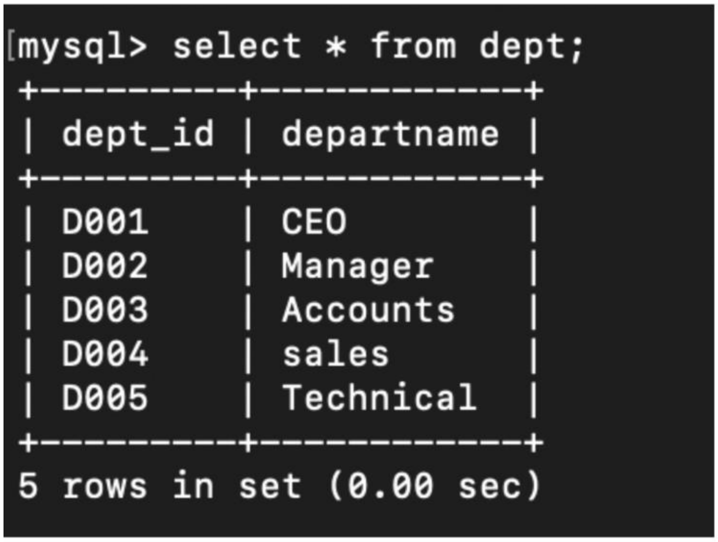
```
[mysql> select * from categories;
+--------+-------------+
| cat_id | cat_name    |
+--------+-------------+
|      1 | MOBILE      |
|      2 | FURNITURE   |
|      3 | BEDDING     |
|      4 | CLOTHING    |
|      5 | ELECTRONICS |
|      6 | BEAUTY      |
+--------+-------------+
6 rows in set (0.01 sec)
```

4.cust



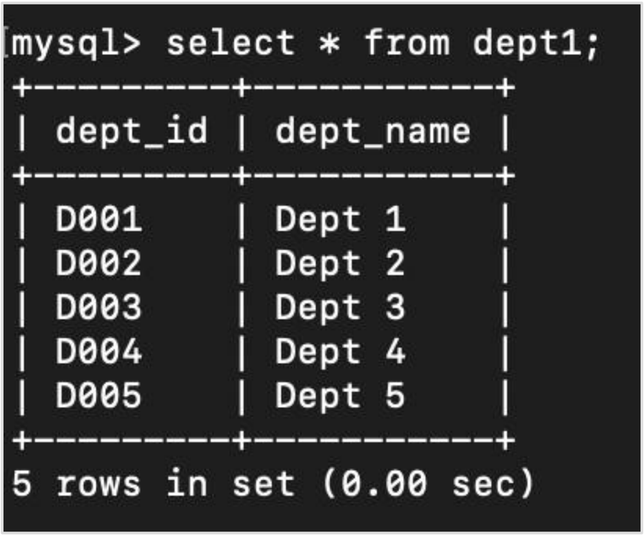
```
mysql> select * from cust;
+---------+
| custid  |
+---------+
| C200012 |
| C20002  |
| C20003  |
| C20004  |
| C20005  |
| C20006  |
| C20007  |
| C20008  |
| C20009  |
| C200114 |
+---------+
10 rows in set (0.01 sec)
```

5.dept



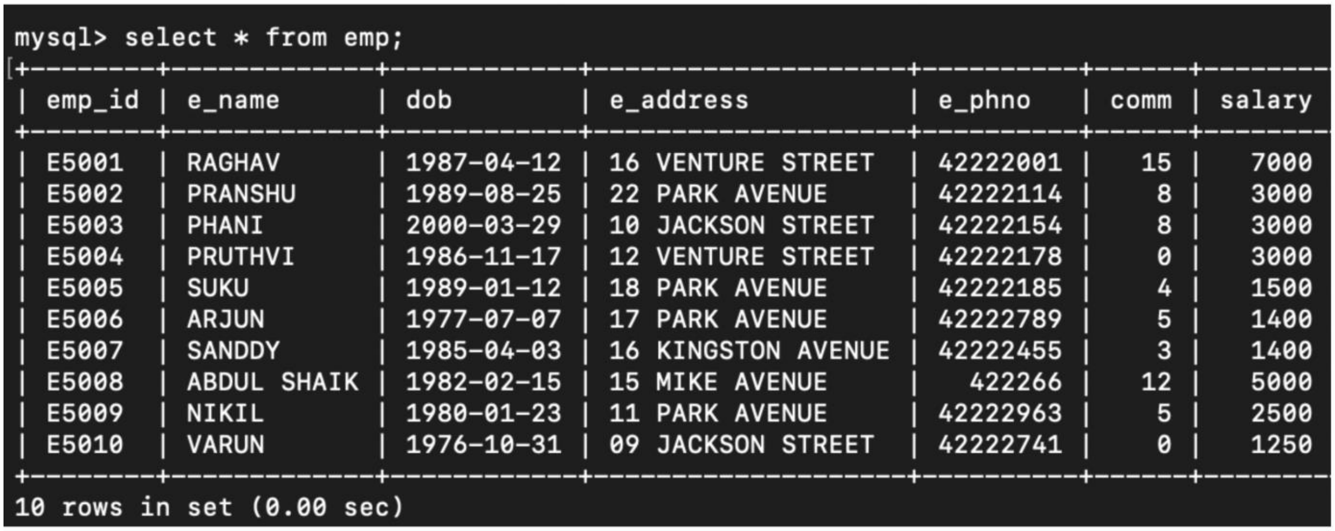
```
[mysql> select * from dept;
+---------+------------+
| dept_id | departname |
+---------+------------+
| D001    | CEO        |
| D002    | Manager    |
| D003    | Accounts   |
| D004    | sales      |
| D005    | Technical  |
+---------+------------+
5 rows in set (0.00 sec)
```

6.dept1



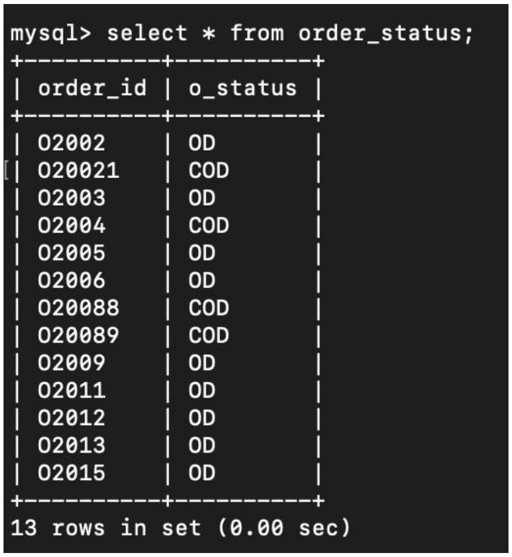
```
[mysql> select * from dept1;
+---------+-----------+
| dept_id | dept_name |
+---------+-----------+
| D001    | Dept 1    |
| D002    | Dept 2    |
| D003    | Dept 3    |
| D004    | Dept 4    |
| D005    | Dept 5    |
+---------+-----------+
5 rows in set (0.00 sec)
```

7.emp



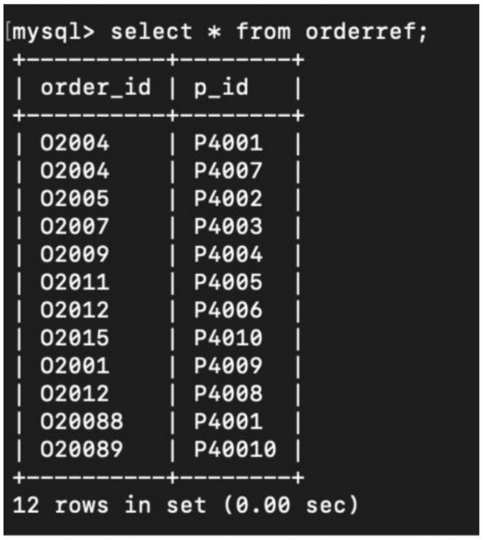
```
mysql> select * from emp;
[+--------+-------------+------------+--------------------+----------+------+--------
| emp_id | e_name      | dob        | e_address          | e_phno   | comm | salary
+--------+-------------+------------+--------------------+----------+------+--------
| E5001  | RAGHAV      | 1987-04-12 | 16 VENTURE STREET  | 42222001 |   15 |   7000
| E5002  | PRANSHU     | 1989-08-25 | 22 PARK AVENUE     | 42222114 |    8 |   3000
| E5003  | PHANI       | 2000-03-29 | 10 JACKSON STREET  | 42222154 |    8 |   3000
| E5004  | PRUTHVI     | 1986-11-17 | 12 VENTURE STREET  | 42222178 |    0 |   3000
| E5005  | SUKU        | 1989-01-12 | 18 PARK AVENUE     | 42222185 |    4 |   1500
| E5006  | ARJUN       | 1977-07-07 | 17 PARK AVENUE     | 42222789 |    5 |   1400
| E5007  | SANDDY      | 1985-04-03 | 16 KINGSTON AVENUE | 42222455 |    3 |   1400
| E5008  | ABDUL SHAIK | 1982-02-15 | 15 MIKE AVENUE     |   422266 |   12 |   5000
| E5009  | NIKIL       | 1980-01-23 | 11 PARK AVENUE     | 42222963 |    5 |   2500
| E5010  | VARUN       | 1976-10-31 | 09 JACKSON STREET  | 42222741 |    0 |   1250
+--------+-------------+------------+--------------------+----------+------+--------
10 rows in set (0.00 sec)
```

8.order status



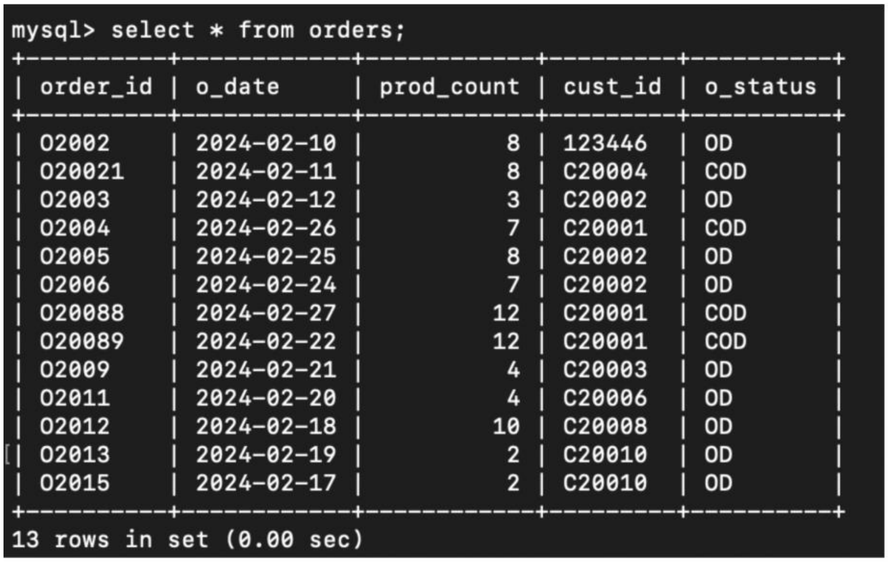
```
mysql> select * from order_status;
+----------+----------+
| order_id | o_status |
+----------+----------+
| O2002    | OD       |
[| O20021   | COD      |
| O2003    | OD       |
| O2004    | COD      |
| O2005    | OD       |
| O2006    | OD       |
| O20088   | COD      |
| O20089   | COD      |
| O2009    | OD       |
| O2011    | OD       |
| O2012    | OD       |
| O2013    | OD       |
| O2015    | OD       |
+----------+----------+
13 rows in set (0.00 sec)
```

9.orderref



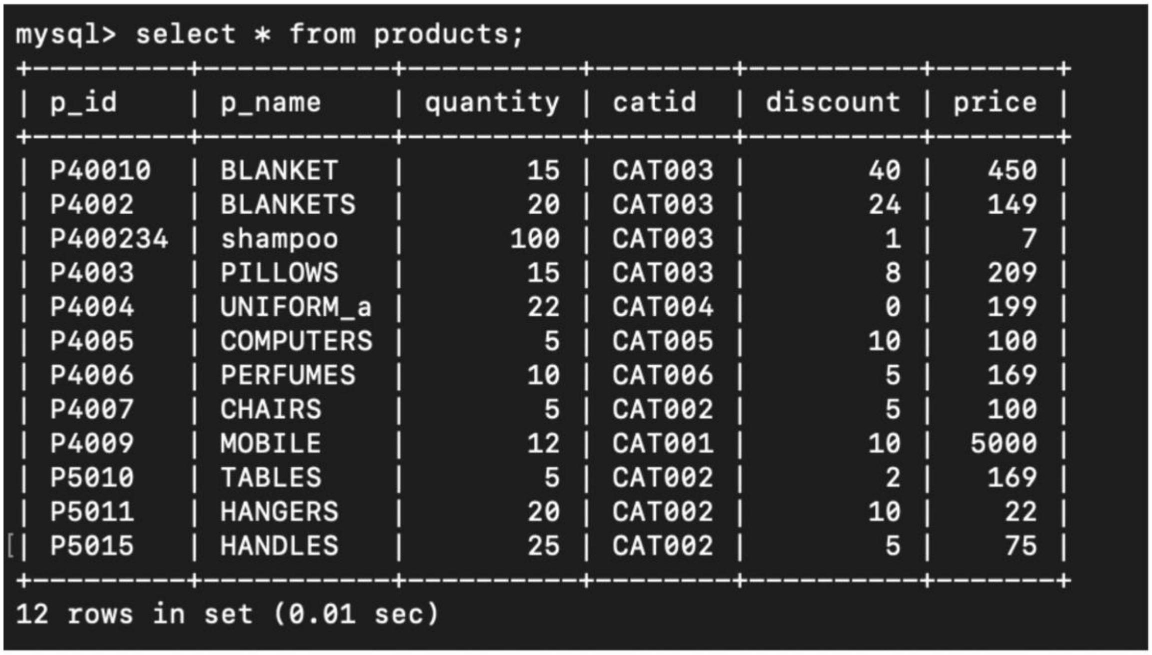
```
[mysql> select * from orderref;
+----------+--------+
| order_id | p_id   |
+----------+--------+
| O2004    | P4001  |
| O2004    | P4007  |
| O2005    | P4002  |
| O2007    | P4003  |
| O2009    | P4004  |
| O2011    | P4005  |
| O2012    | P4006  |
| O2015    | P4010  |
| O2001    | P4009  |
| O2012    | P4008  |
| O20088   | P4001  |
| O20089   | P40010 |
+----------+--------+
12 rows in set (0.00 sec)
```

10. orders



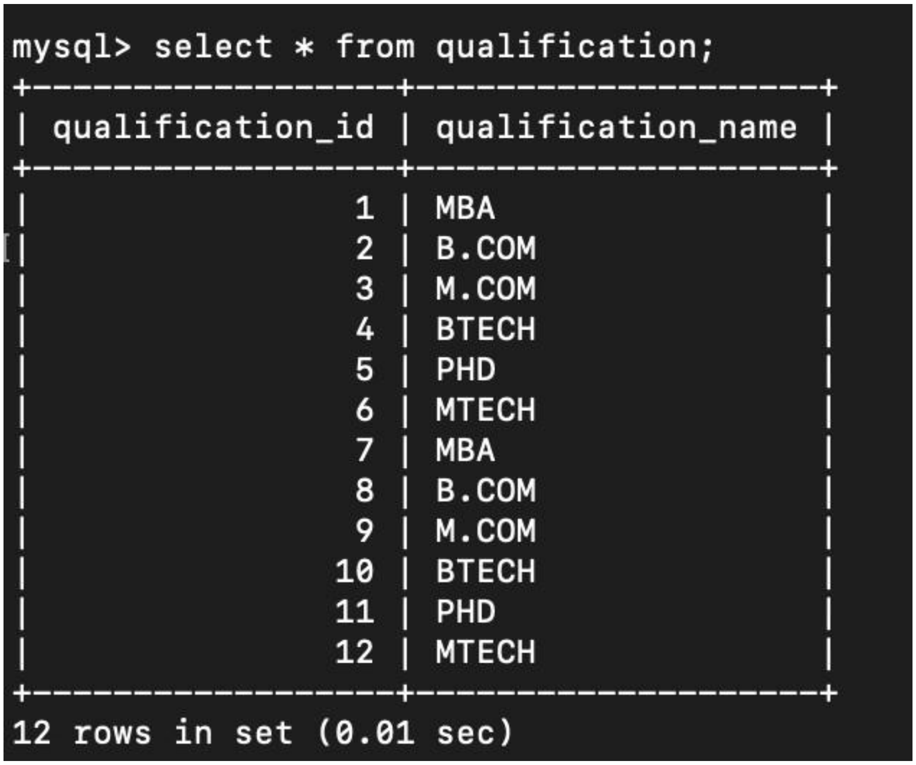
```
mysql> select * from orders;
+----------+------------+------------+---------+----------+
| order_id | o_date     | prod_count | cust_id | o_status |
+----------+------------+------------+---------+----------+
| O2002    | 2024-02-10 |          8 | 123446  | OD       |
| O20021   | 2024-02-11 |          8 | C20004  | COD      |
| O2003    | 2024-02-12 |          3 | C20002  | OD       |
| O2004    | 2024-02-26 |          7 | C20001  | COD      |
| O2005    | 2024-02-25 |          8 | C20002  | OD       |
| O2006    | 2024-02-24 |          7 | C20002  | OD       |
| O20088   | 2024-02-27 |         12 | C20001  | COD      |
| O20089   | 2024-02-22 |         12 | C20001  | COD      |
| O2009    | 2024-02-21 |          4 | C20003  | OD       |
| O2011    | 2024-02-20 |          4 | C20006  | OD       |
| O2012    | 2024-02-18 |         10 | C20008  | OD       |
[| O2013    | 2024-02-19 |          2 | C20010  | OD       |
| O2015    | 2024-02-17 |          2 | C20010  | OD       |
+----------+------------+------------+---------+----------+
13 rows in set (0.00 sec)
```

11. products



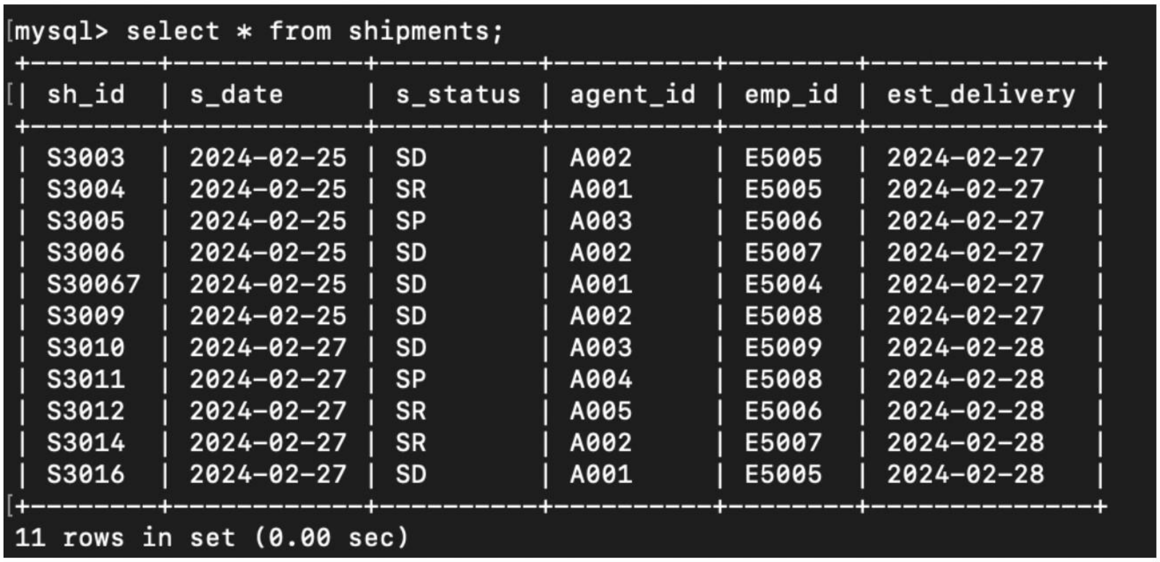
```
mysql> select * from products;
+---------+-----------+----------+--------+----------+-------+
| p_id    | p_name    | quantity | catid  | discount | price |
+---------+-----------+----------+--------+----------+-------+
| P40010  | BLANKET   |       15 | CAT003 |       40 |   450 |
| P4002   | BLANKETS  |       20 | CAT003 |       24 |   149 |
| P400234 | shampoo   |      100 | CAT003 |        1 |     7 |
| P4003   | PILLOWS   |       15 | CAT003 |        8 |   209 |
| P4004   | UNIFORM_a |       22 | CAT004 |        0 |   199 |
| P4005   | COMPUTERS |        5 | CAT005 |       10 |   100 |
| P4006   | PERFUMES  |       10 | CAT006 |        5 |   169 |
| P4007   | CHAIRS    |        5 | CAT002 |        5 |   100 |
| P4009   | MOBILE    |       12 | CAT001 |       10 |  5000 |
| P5010   | TABLES    |        5 | CAT002 |        2 |   169 |
| P5011   | HANGERS   |       20 | CAT002 |       10 |    22 |
[| P5015   | HANDLES   |       25 | CAT002 |        5 |    75 |
+---------+-----------+----------+--------+----------+-------+
12 rows in set (0.01 sec)
```

12. qualification



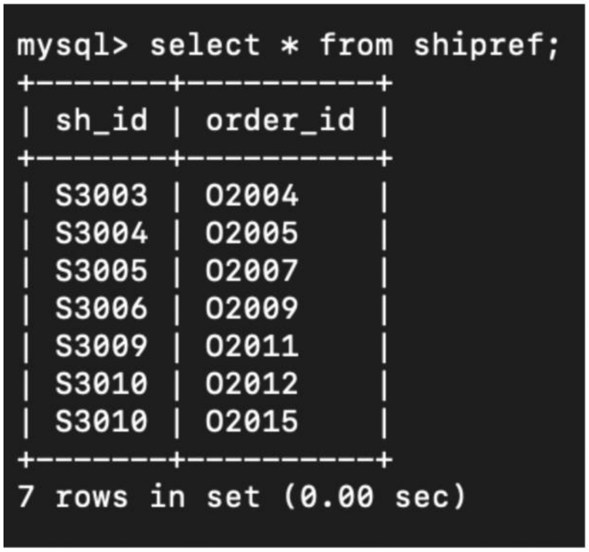
```
mysql> select * from qualification;
+------------------+--------------------+
| qualification_id | qualification_name |
+------------------+--------------------+
|                1 | MBA                |
[|                2 | B.COM              |
|                3 | M.COM              |
|                4 | BTECH              |
|                5 | PHD                |
|                6 | MTECH              |
|                7 | MBA                |
|                8 | B.COM              |
|                9 | M.COM              |
|               10 | BTECH              |
|               11 | PHD                |
|               12 | MTECH              |
+------------------+--------------------+
12 rows in set (0.01 sec)
```

13. shipments



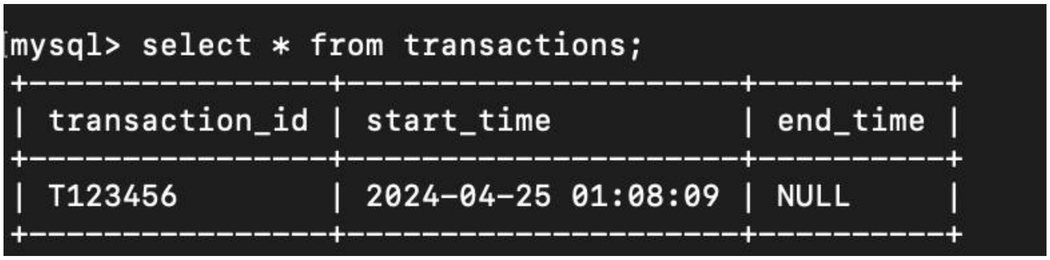
```
[mysql> select * from shipments;
+--------+------------+----------+----------+--------+--------------+
[| sh_id  | s_date     | s_status | agent_id | emp_id | est_delivery |
+--------+------------+----------+----------+--------+--------------+
| S3003  | 2024-02-25 | SD       | A002     | E5005  | 2024-02-27   |
| S3004  | 2024-02-25 | SR       | A001     | E5005  | 2024-02-27   |
| S3005  | 2024-02-25 | SP       | A003     | E5006  | 2024-02-27   |
| S3006  | 2024-02-25 | SD       | A002     | E5007  | 2024-02-27   |
| S30067 | 2024-02-25 | SD       | A001     | E5004  | 2024-02-27   |
| S3009  | 2024-02-25 | SD       | A002     | E5008  | 2024-02-27   |
| S3010  | 2024-02-27 | SD       | A003     | E5009  | 2024-02-28   |
| S3011  | 2024-02-27 | SP       | A004     | E5008  | 2024-02-28   |
| S3012  | 2024-02-27 | SR       | A005     | E5006  | 2024-02-28   |
| S3014  | 2024-02-27 | SR       | A002     | E5007  | 2024-02-28   |
| S3016  | 2024-02-27 | SD       | A001     | E5005  | 2024-02-28   |
[+--------+------------+----------+----------+--------+--------------+
11 rows in set (0.00 sec)
```

14. shipref



```
mysql> select * from shipref;
+-------+----------+
| sh_id | order_id |
+-------+----------+
| S3003 | O2004    |
| S3004 | O2005    |
| S3005 | O2007    |
| S3006 | O2009    |
| S3009 | O2011    |
| S3010 | O2012    |
| S3010 | O2015    |
+-------+----------+
7 rows in set (0.00 sec)
```

15. transactions



```
[mysql> select * from transactions;
+----------------+---------------------+----------+
| transaction_id | start_time          | end_time |
+----------------+---------------------+----------+
| T123456        | 2024-04-25 01:08:09 | NULL     |
+----------------+---------------------+----------+
```

CHAPTER 3

Complex queries based on the concepts of constraints, sets, joins, views, Triggers and Cursors

1.Constraints: Query to add a foreign key constraint.

```
ALTER TABLE salesinfo
ADD CONSTRAINT fk_productcode
FOREIGN KEY (productcode) REFERENCES products(productcode);
```

2.Sets: Query to find products that are common between current stock and sales info.

```
SELECT productcode FROM currentstock
INTERSECT
SELECT productcode FROM salesinfo;
```

3.Joins: Query to get a list of sales with customer details.

```
SELECT s.salesid, s.date, p.productname, c.fullname
FROM salesinfo s
INNER JOIN products p ON s.productcode = p.productcode
INNER JOIN customers c ON s.customercode = c.customercode;
```

4.Views: Create a view to display product details along with current stock.

```
CREATE VIEW product_stock AS
SELECT p.productname, p.brand, c.quantity
FROM products p
INNER JOIN currentstock c ON p.productcode = c.productcode;
```

5.Triggers: Trigger to update current stock after a sale.

```
CREATE TRIGGER update_stock AFTER INSERT ON salesinfo FOR
EACH ROW
BEGIN
UPDATE currentstock
```

```
SET quantity = quantity - NEW.quantity WHERE productcode =
NEW.productcode;
END;
```

6.Cursors: Cursor to calculate total revenue for each product.

```
DELIMITER //
CREATE PROCEDURE calculate_revenue() BEGIN
DECLARE done INT DEFAULT FALSE;
DECLARE prod_code VARCHAR(45);
DECLARE total_revenue DOUBLE;
DECLARE cur CURSOR FOR SELECT DISTINCT productcode FROM salesinfo;
DECLARE CONTINUE
HANDLER FOR NOT FOUND SET done = TRUE;
OPEN cur; revenue_loop: LOOP
FETCH cur INTO prod_code; IF done THEN
LEAVE revenue_loop;
END IF;
SELECT SUM(revenue) INTO total_revenue FROM salesinfo WHERE productcode =
prod_code;
INSERT INTO revenue_summary (productcode, total_revenue) VALUES (prod_code,
total_revenue); END LOOP; CLOSE cur;
END //
DELIMITER ;
CALL calculate_revenue();
```

CHAPTER 4

Analyzing the pitfalls, identifying the dependencies, and applying normalizations

I.Pitfalls Identified

1.Data Integrity Constraints: While you've defined primary keys and some foreign keys, there are missing foreign key constraints between tables. For example, the cust_id in the orders table shouldreference the custid in the cust table. Adding these constraints ensures data integrity and prevents orphan records.

2.Data Consistency: Some columns like email_id in the cust table and p_id in the orderref table should be defined as VARCHAR(100) and VARCHAR(10) respectively, to accommodate longer email addresses or product IDs.

3.Normalization: The schema could benefit from further normalization to reduce redundancy and improve data integrity. For example, instead of storing agent names directly in the orders table, a separate agent table could be created and linked to the orders table via foreign key. This would prevent inconsistencies if an agent's name were to change.

4.Data Completeness: There are some fields marked as DEFAULT NULL which could potentially lead to incomplete data if not handled properly. For example, the password field in the cust table should not be allowed to be NULL to ensure that all customer records have a password associated with them.

5.Data Types: Ensure that appropriate data types are used for each column. For example, using CHAR(20) for cat_id and dept_id may not be efficient if most values are shorter. Using VARCHAR might be more appropriate unless there's a specific reason for fixed-length strings.

6.Indexing: Consider adding indexes to columns that are frequently used in search queries or join conditions to improve query performance. For example, adding an index on cust_id in the orders table could speed up searches based on customer IDs.

7.Error Handling: Ensure proper error handling mechanisms are in place, especially for user inputs such as email addresses and passwords to prevent SQL injection attacks and other security vulnerabilities.

8. **Consistent Naming Conventions:** It's a good practice to use consistent naming conventions for tables and columns. For example, you have orders and orderref, itmight be clearer if they were named consistently like orders and order_references.

9. **Data Validation:** Implement data validation mechanisms to ensure that only valid data is inserted into the database. This can help prevent data corruption and maintain data quality over time.

10. **Backup and Recovery:** Implement regular backup and recovery procedures to protect against data loss due to hardware failure, human error, or other unforeseen circumstances.

II. Functional Dependencies [FDs]

custid → c_name, phoneno, c_address, email_id, password

The customer ID uniquely determines the customer's name, phone number, address, email, andpassword.

dept_id → departname

Each department ID uniquely determines the department name

emp_id → e_name, dept_id, qualification, dob, e_address, e_phno, comm, salary

The employee ID uniquely determines the employee's name, department ID, qualification, date of birth, address, phone number, commission, and salary.

order_id → o_date, prod_count, cust_id, o_status

Each order ID uniquely determines the order date, product count, customer ID, and order status.

p_id → p_name, quantity, catid, discount, price

Each product ID uniquely determines the product name, quantity, category ID, discount, and price.

sh_id → s_date, s_status, agent_id, emp_id, est_delivery

Each shipment ID uniquely determines the shipment date, shipment status, agent ID, employee ID,and estimated delivery date.
agent_id → agent_name

Each agent ID uniquely determines the agent name.

cat_id → category

Each category ID uniquely determines the category name.

sh_id, order_id → (composite dependency)

Each combination of shipment ID and order ID uniquely determines the shipment associated with the order.

III. Normalisation

First Normal Form (1NF)

A table is in 1NF if:

- It contains only atomic values.

- There are no repeating groups or arrays.

All the provided tables seem to be in 1NF as they do not contain repeating groups, and each cell holds a single value.

Second Normal Form (2NF)

A table is in 2NF if:

- It is in 1NF.

- All non-key attributes are fully functional dependent on the primary key.

Let's check each table:

Table: agentd

- agent_id is the primary key, and agent_name is fully functionally dependent on it.
- Already in 2NF.

Table: categ

- cat_id is the primary key, and category is fully functionally dependent on it.
- Already in 2NF.

Table: cust

- custid is the primary key.
- There is a partial dependency on the primary key for the password column (if password depends only on custid). We need to remove this partial dependency.
- To achieve 2NF, we need to remove the partial dependency of the password column on the custid.

Table: dept

- dept_id is the primary key, and departname is fully functionally dependent on it.
- Already in 2NF.

Table: emp

- emp_id is the primary key.
- dept_id is a foreign key.
- The rest of the attributes are fully functionally dependent on the primary key.
- Already in 2NF.

Table: orders

- order_id is the primary key.
- The o_status column is functionally dependent only on order_id. However, it's a non-key attribute. We need to check if it has a dependency on any other attribute.
- cust_id is a foreign key.

Table: products

- p_id is the primary key.
- catid is a foreign key.
- All non-key attributes are fully functionally dependent on the primary key.
- Already in 2NF.

Table: shipments

- sh_id is the primary key.
- agent_id and emp_id are foreign keys.
- All non-key attributes are fully functionally dependent on the primary key.
- Already in 2NF.

Table: shipref

- sh_id is a foreign key.
- order_id is a foreign key.
- Already in 2NF.

Addressing 2NF Issues

- **cust:**To remove the partial dependency of the password column on the custid, we'll create a separate table for customer authentication.
- **orders:**The o_status column depends only on order_id, no further normalization required.

Third Normal Form (3NF)

A table is in 3NF if:

- It is in 2NF.
- All non-key attributes are non-transitively dependent on the primary key.

We need to create a separate table for customer authentication to ensure 3NF.

Here are the normalized tables:

Table: cust_auth

Field	Type	Null	Key
Custid	char(10)	NO	PRI
email_id	char(100)	NO	
password	char(20)	NO	

Table: cust

Field	Type	Null	Key
custid	char(10)	NO	PRI
c_name	char(15)	YES	
phoneno	varchar(10)	NO	UNI
c_address	char(20)	YES	

Now, the database is in 3NF.

CHAPTER 5

Implementation of concurrency control and recovery mechanisms

I.Concurrency Control

Concurrency control is a critical aspect of database management systems, ensuring that multiple transactions can execute concurrently without interfering with each other. There are several methods to implement concurrency control, with the two main approaches being:

1.Lock-based Concurrency Control

2.Timestamp-based Concurrency Control

Let's dive into each of these methods:

1. Lock-based Concurrency Control

In lock-based concurrency control, transactions acquire locks on data items to prevent other transactions from accessing or modifying them until the lock is released. There are different types of locks:

- **Read Lock (Shared Lock):** Allows multiple transactions to read the data item simultaneously but prevents any transaction from modifying it.

- **Write Lock (Exclusive Lock):** Allows only one transaction to modify the data item and prevents all other transactions from reading or writing it.

Implementation Steps

1.Lock Table: Create a table to store information about locks.

```
CREATE TABLE locks (
  lock_id INT AUTO_INCREMENT PRIMARY KEY,
  resource_id VARCHAR(50) NOT NULL,
  locked_by VARCHAR(50) NOT NULL,
  lock_type ENUM('READ', 'WRITE') NOT NULL,
  FOREIGN KEY (locked_by) REFERENCES transactions(transaction_id)
) ENGINE=InnoDB;
```

2.Transactions Table: Create a table to store information about active transactions.

```
CREATE TABLE transactions (
 transaction_id VARCHAR(50) PRIMARY KEY,
start_time TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
end_time TIMESTAMP DEFAULT NULL
) ENGINE=InnoDB;
```

3.Example Transaction (Read):

```
START TRANSACTION;
SELECT * FROM your_table WHERE some_condition;
-- Read data and process it
COMMIT;
```

4. Example Transaction (Write)

```
START TRANSACTION;

-- Lock the resource for writing

INSERT INTO locks (resource_id, locked_by, lock_type) VALUES ('your_table', 'transaction_id', 'WRITE');

-- Perform the write operation

UPDATE your_table SET column1 = value1 WHERE some_condition;

-- Release the lock

DELETE FROM locks WHERE resource_id = 'your_table' AND locked_by = 'transaction_id';

COMMIT;
```

2. Timestamp-based Concurrency Control

In timestamp-based concurrency control, each transaction is assigned a unique timestamp, and the database system uses these timestamps to determine the order in which transactions should be executed.

Implementation Steps

1.Transactions Table with Timestamps

```
CREATE TABLE transactions (

  transaction_id VARCHAR(50) PRIMARY KEY,

  start_time TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

  end_time TIMESTAMP DEFAULT NULL

) ENGINE=InnoDB;
```

2.Example Transaction

```
START TRANSACTION;

-- Read or write operations

COMMIT;
```

3.Concurrency Control at Query Execution: For each query execution, the DBMS compares the timestamps of the requesting transaction and the last transaction that modified the data item to ensure that the requesting transaction sees a consistent view of the database.

Deadlock Detection and Resolution

In addition to basic concurrency control mechanisms, deadlock detection and resolution mechanisms should be implemented to handle situations where transactions are waiting for each other, resulting in a deadlock.

Deadlock Detection

- Periodically scan the locks table to detect cycles in the wait-for graph.
- Once a deadlock is detected, one or more transactions involved in the deadlock are rolled back.

Deadlock Resolution

- Rollback one or more transactions to break the deadlock.
- The choice of which transaction to roll back can be based on factors such as transaction priority, transaction age, etc.

Example

```
-- Rollback a transaction involved in a deadlock

ROLLBACK transaction_id;
```

Concurrency control ensures that transactions execute correctly and efficiently in a multi-user environment, maintaining the integrity and consistency of the database. The choice between lock-based and timestamp-based concurrency control depends on factors such as the nature of the application, the database system being used, and performance considerations.

II. RECOVERY MECHANISMS

1. Regular Backups

- Perform regular backups of your database using MySQL's mysqldump utility or any other backup tool.
- Schedule backups to run automatically at regular intervals.

2. Binary Logging

- MySQL supports binary logging which records all changes to the database.
- Binary logs can be used for point-in-time recovery.
- You can enable binary logging by setting the log_bin variable in your MySQL configuration file.

3. Transaction Logs

- MySQL uses transaction logs (also known as redo logs) to recover from crashes.
- Transaction logs store information about all committed transactions.
- InnoDB, the default storage engine for MySQL, uses transaction logs for crash recovery.

4. RAID (Redundant Array of Independent Disks)

- Use RAID configurations to duplicate data across multiple disks.
- RAID configurations can improve fault tolerance and help in recovering from disk failures.

5. Replication

- Set up master-slave replication to create redundant copies of your data.
- In case of a failure on the master server, you can promote one of the slave servers to become the new master.

6. Point-in-time Recovery (PITR)

- Use a combination of binary logs and full backups to restore your database to a specific point in time.

- Restore the latest full backup and apply binary logs up to the desired recovery point.

7. Monitoring and Alerting

- Implement monitoring systems to alert you of any potential issues.

- Monitor disk space, server performance, and database health.

8. Testing Recovery Procedures

- Regularly test your recovery procedures to ensure they work as expected.

- Simulate different failure scenarios and verify that you can recover your database without data loss.

These mechanisms can be implemented individually or in combination to provide comprehensive protection for your MySQL database against failures and data loss.

CHAPTER 6

Code for the project

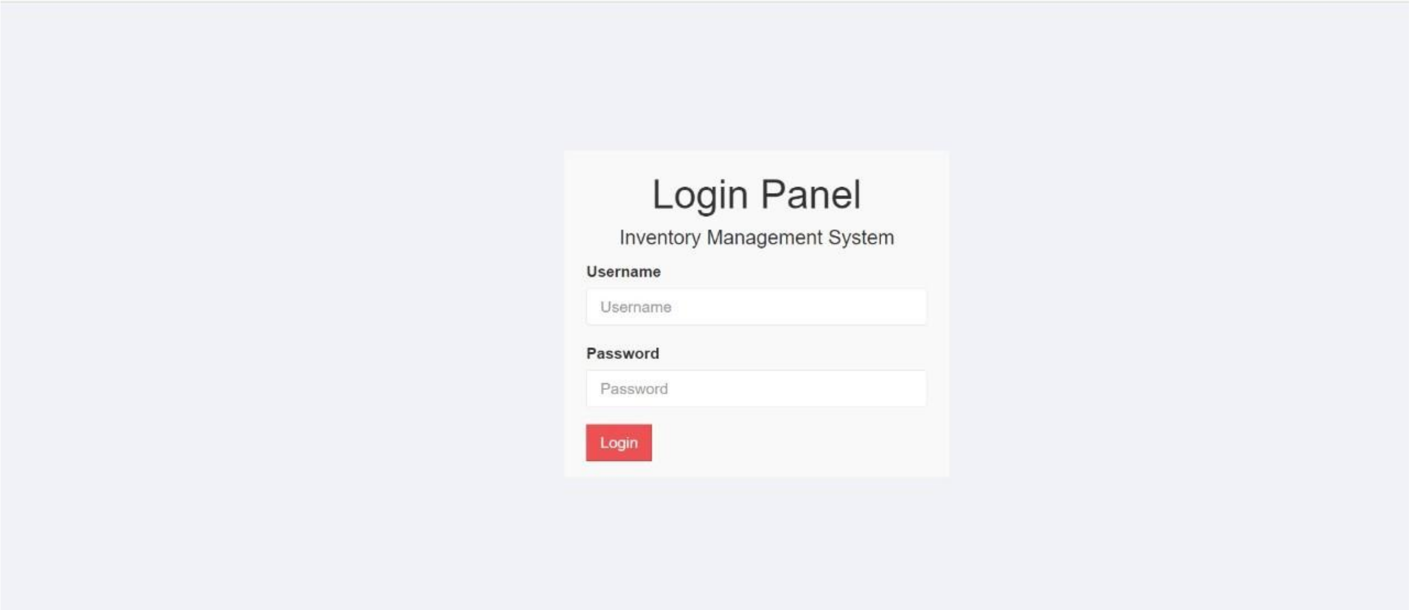
```
CREATE TABLE agentd (
agent_id varchar(5) NOT NULL,
agent_name varchar(10) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

INSERT INTO agentd (agent_id, agent_name) VALUES
('A001', 'transco'),
('A002', 'fedx'),
('A003', 'upl'),
('A004', 'bluedart'),
('A005', ‘dhl');

CREATE TABLE categ (
cat_id char(20) NOT NULL,
category char(20) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

INSERT INTO categ (cat_id, category) VALUES
('CAT001', 'essentials'),
('CAT002', 'Furniture'),
('CAT003', 'bedding'),
('CAT004', 'fashion'),
('CAT005', 'electronics'),
('CAT006', ‘perfumes');

CREATE TABLE cust (
custid char(10) NOT NULL,
c_name char(15) DEFAULT NULL,
phoneno varchar(10) NOT NULL,
c_address char(20) DEFAULT NULL,
email_id char(100) NOT NULL,
password char(20) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

INSERT INTO cust (custid, c_name, phoneno, c_address, email_id, password) VALUES
('C200012', 'JOHN', '422234565', '16 BARREN STREET', 'abc@mail.com',
'john@123'),
('C20002', 'ROCK', '42223457', '15 HOLMES STREET', 'rock1@mail.com',
'rock@123'),
('C20003', 'ADAM', '42223458', '11 BAKERS STREET', 'adam1@mail.com', "),
('C20004', 'EVA', '42223459', '23 LADA STREET', 'eva1@mail.com', "),
('C20005', 'SHYAM', '42223460', '18 LADA STREET', 'shyam1@mail.com', "),
('C20006', 'KAREN', '42223461', '15 BAKERS STREET', 'karen1@mail.com',
'karan@123'),
('C20007', 'MIKE', '42223462', '14 LOTUS STREET', 'mike1@mail.com', "),
('C20008', 'ROSE', '42223463', '12 ALOK STREET', 'rose1@mail.com', "),
('C20009', 'DIVI', '42223464', '17 AFFLE AVENCE', 'divi1@mail.com', "),
('C200114', 'ABHI boss', '422234654', '10 HOLME STREET', 'abhi1@gmail.com',
‘abhi@123');

CREATE TABLE dept (
dept_id varchar(5) NOT NULL,
departname varchar(20) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

INSERT INTO dept (dept_id, departname) VALUES
('D001', 'CEO'),
('D002', 'Manager'),('D003', 'Accounts'),
('D004', 'sales'),
('D005', ‘Technical');

CREATE TABLE emp (
emp_id varchar(10) NOT NULL,
e_name varchar(15) DEFAULT NULL,
dept_id varchar(5) DEFAULT NULL,
qualification varchar(20) DEFAULT NULL,
dob date DEFAULT NULL,
e_address varchar(20) DEFAULT NULL,
e_phno int(11) DEFAULT NULL,
comm int(3) DEFAULT NULL,
salary int(6) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

INSERT INTO emp (emp_id, e_name, dept_id, qualification, dob, e_address, e_phno,
comm, salary)
VALUES
('E5001', 'RAGHAV', 'D001', 'MBA', '1987-04-12', '16 VENTURE STREET', 42222001,
15, 7000),
('E5002', 'PRANSHU', 'D002', 'MBA', '1989-08-25', '22 PARK AVENUE', 42222114, 8,
3000),
('E5003', 'PHANI', 'D002', 'MBA', '2000-03-29', '10 JACKSON STREET', 42222154, 8,
3000),
('E5004', 'PRUTHVI', 'D003', 'MBA', '1986-11-17', '12 VENTURE STREET',
42222178, 0, 3000),
('E5005', 'SUKU', 'D004', 'B.COM', '1989-01-12', '18 PARK AVENUE', 42222185, 4,
1500),
('E5006', 'ARJUN', 'D005', 'M.COM', '1977-07-07', '17 PARK AVENUE', 42222789, 5,
1400),
('E5007', 'SANDDY', 'D002', 'BTECH', '1985-04-03', '16 KINGSTON AVENUE',
42222455, 3, 1400),
('E5008', 'ABDUL SHAIK', 'D001', 'PHD', '1982-02-15', '15 MIKE AVENUE', 422266,
12, 5000),
('E5009', 'NIKIL', 'D005', 'BTECH', '1980-01-23', '11 PARK AVENUE', 42222963, 5,
2500),
('E5010', 'VARUN', 'D005', 'MTECH', '1976-10-31', '09 JACKSON STREET',
42222741, 0, 1250);

CREATE TABLE orderref (
order_id varchar(10) DEFAULT NULL,
p_id varchar(10) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

INSERT INTO orderref (order_id, p_id) VALUES
('O2004', 'P4001'),
('O2004', 'P4007'),
('O2005', 'P4002'),
('O2007', 'P4003'),
('O2009', 'P4004'),
('O2011', 'P4005'),
('O2012', 'P4006'),
('O2015', 'P4010'),
('O2001', 'P4009'),
('O2012', 'P4008'),
('O20088', 'P4001'),
('O20089', ‘P40010');

CREATE TABLE orders (
order_id varchar(10) NOT NULL,
o_date date DEFAULT NULL,
prod_count int(10) DEFAULT NULL,
cust_id varchar(10) DEFAULT NULL,
o_status varchar(10) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

INSERT INTO orders (order_id, o_date, prod_count, cust_id, o_status) VALUES
('O2002', '2024-02-10', 8, '123446', 'OD'),
('O20021', '2024-02-11', 8, 'C20004', 'COD'),
('O2003', '2024-02-12', 3, 'C20002', 'OD'),
('O2004', '2024-02-26', 7, 'C20001', 'COD'),
('O2005', '2024-02-25', 8, 'C20002', 'OD'),
('O2006', '2024-02-24', 7, 'C20002', 'OD'),
('O20088', '2024-02-27', 12, 'C20001', 'COD'),
('O20089', '2024-02-22', 12, 'C20001', 'COD'),
('O2009', '2024-02-21', 4, 'C20003', 'OD'),
('O2011', '2024-02-20', 4, 'C20006', 'OD'),
('O2012', '2024-02-18', 10, 'C20008', 'OD'),
('O2013', '2024-02-19', 2, 'C20010', 'OD'),
('O2015', '2024-02-17', 2, 'C20010', ‘OD');

CREATE TABLE products (
p_id char(10) NOT NULL,
p_name char(10) DEFAULT NULL,
quantity int(5) DEFAULT NULL,
catid char(20) DEFAULT NULL,
discount int(3) DEFAULT NULL,
price int(5) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

INSERT INTO products (p_id, p_name, quantity, catid, discount, price) VALUES
('P40010', 'BLANKET', 12, 'CAT003', 40, 450),
('P4002', 'BLANKETS', 20, 'CAT003', 24, 149),
('P400234', 'shampoo', 100, 'CAT003', 1, 7),('P4003', 'PILLOWS', 15, 'CAT003', 8, 209),
('P4004', 'UNIFORM_a', 22, 'CAT004', 0, 199),
('P4005', 'COMPUTERS', 5, 'CAT005', 10, 100),
('P4006', 'PERFUMES', 10, 'CAT006', 5, 169),
('P4007', 'CHAIRS', 5, 'CAT002', 5, 100),
('P4009', 'MOBILE', 12, 'CAT001', 10, 5000),
('P5010', 'TABLES', 5, 'CAT002', 2, 169),
('P5011', 'HANGERS', 20, 'CAT002', 10, 22),
('P5015', 'HANDLES', 25, 'CAT002', 5, 75);

CREATE TABLE shipments (
sh_id varchar(10) NOT NULL,
s_date date DEFAULT NULL,
s_status varchar(10) DEFAULT NULL,
agent_id varchar(5) DEFAULT NULL,
emp_id varchar(10) DEFAULT NULL,
est_delivery date DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

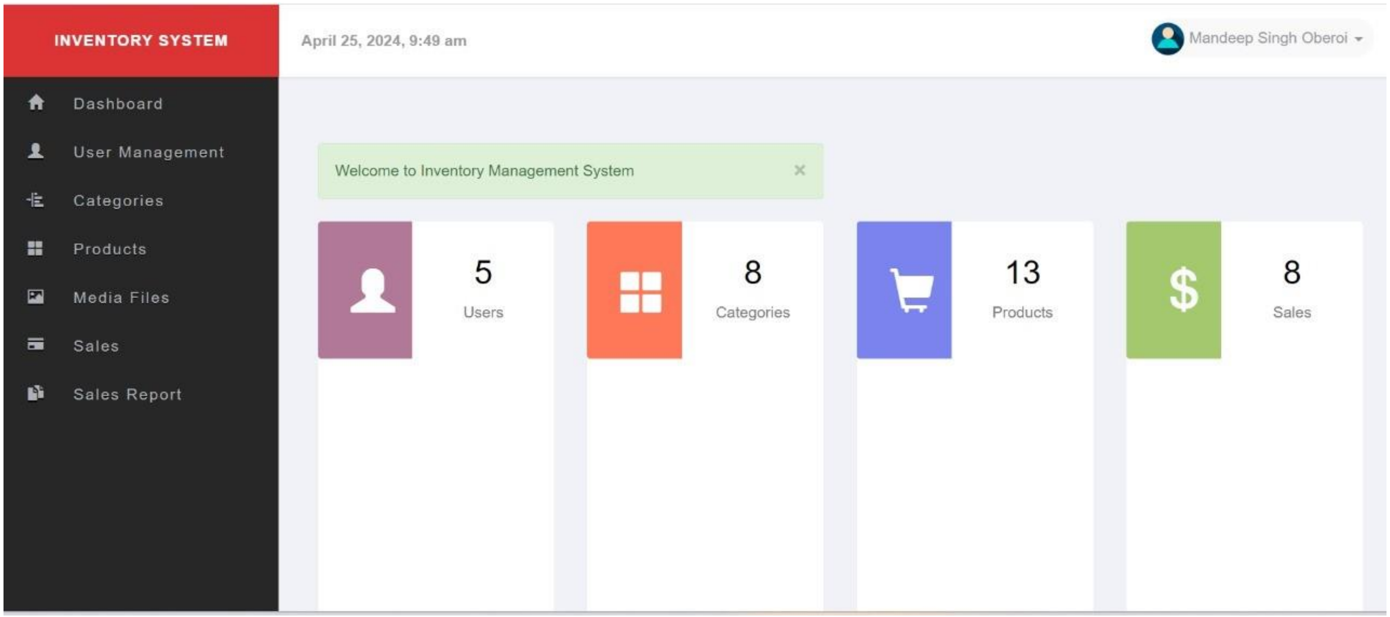
```
INSERT INTO shipments (sh_id, s_date, s_status, agent_id, emp_id, est_delivery)
VALUES
('S3003', '2024-02-25', 'SD', 'A002', 'E5005', '2024-02-27'),
('S3004', '2024-02-25', 'SR', 'A001', 'E5005', ‘2024-02-27'),
('S3005', '2024-02-25', 'SP', 'A003', 'E5006', '2024-02-27'),
('S3006', '2024-02-25', 'SD', 'A002', 'E5007', '2024-02-27'),
('S30067', '2024-02-25', 'SD', 'A001', 'E5004', '2024-02-27'),
('S3009', '2024-02-25', 'SD', 'A002', 'E5008', '2024-02-27'),
('S3010', '2024-02-27', 'SD', 'A003', 'E5009', '2024-02-28'),
('S3011', '2024-02-27', 'SP', 'A004', 'E5008', '2024-02-28'),
('S3012', '2024-02-27', 'SR', 'A005', 'E5006', '2024-02-28'),('S3014', '2024-02-27', 'SR',
'A002', 'E5007', '2024-02-28'),
('S3016', '2024-02-27', 'SD', 'A001', 'E5005', ‘2024-02-28');

CREATE TABLE shipref (
sh_id varchar(10) DEFAULT NULL,
order_id varchar(10) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

INSERT INTO shipref (sh_id, order_id) VALUES
('S3003', 'O2004'),
('S3004', 'O2005'),
('S3005', 'O2007'),
('S3006', 'O2009'),
('S3009', 'O2011'),
('S3010', 'O2012'),
('S3010', ‘O2015');

ALTER TABLE agentd
ADD PRIMARY KEY (agent_id);

ALTER TABLE categ
ADD PRIMARY KEY (cat_id);

ALTER TABLE cust
ADD PRIMARY KEY (custid),
ADD UNIQUE KEY phoneno (phoneno);

ALTER TABLE dept
ADD PRIMARY KEY (dept_id);

ALTER TABLE emp
ADD PRIMARY KEY (emp_id);
```

```
ALTER TABLE orders
ADD PRIMARY KEY (order_id);

ALTER TABLE products
ADD PRIMARY KEY (p_id);

ALTER TABLE shipments
ADD PRIMARY KEY (sh_id);

COMMIT;
```

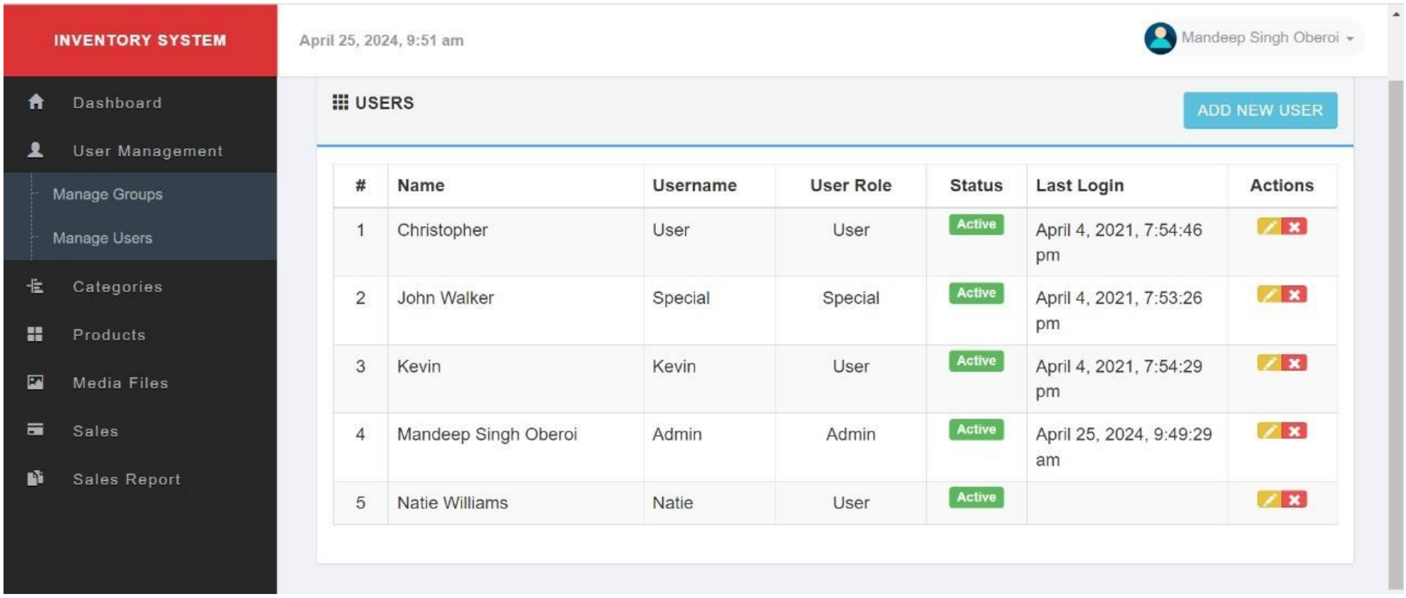
CHAPTER 7

Results and Discussion

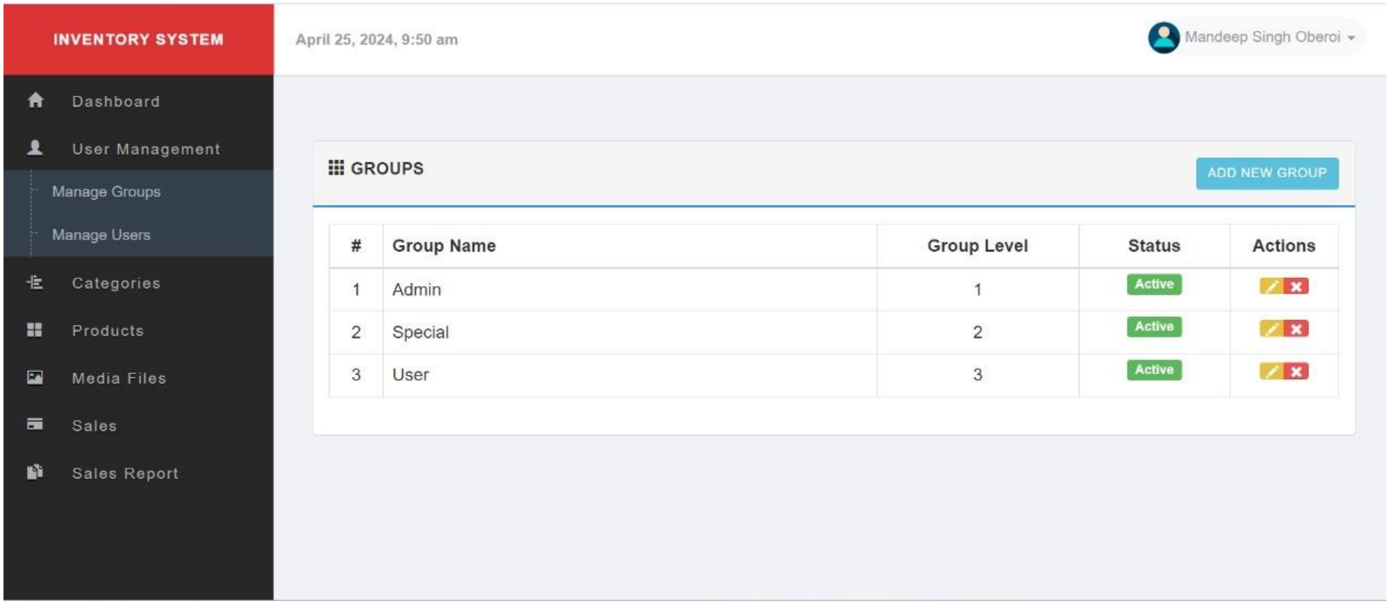




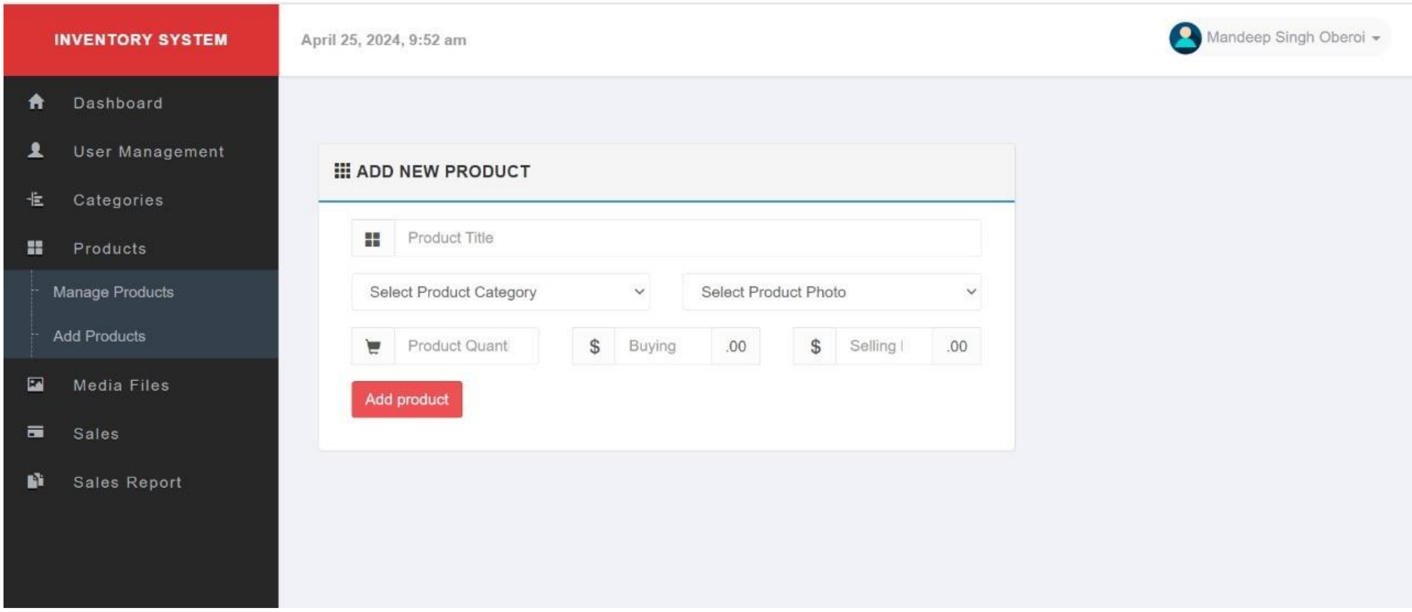
Welcome to Inventory Management System



#	Name	Username	User Role	Status	Last Login	Actions
1	Christopher	User	User	Active	April 4, 2021, 7:54:46 pm	✎ ✖
2	John Walker	Special	Special	Active	April 4, 2021, 7:53:26 pm	✎ ✖
3	Kevin	Kevin	User	Active	April 4, 2021, 7:54:29 pm	✎ ✖
4	Mandeep Singh Oberoi	Admin	Admin	Active	April 25, 2024, 9:49:29 am	✎ ✖
5	Natie Williams	Natie	User	Active		✎ ✖



#	Group Name	Group Level	Status	Actions
1	Admin	1	Active	✎ ✖
2	Special	2	Active	✎ ✖
3	User	3	Active	✎ ✖



ADD NEW PRODUCT

Product Title

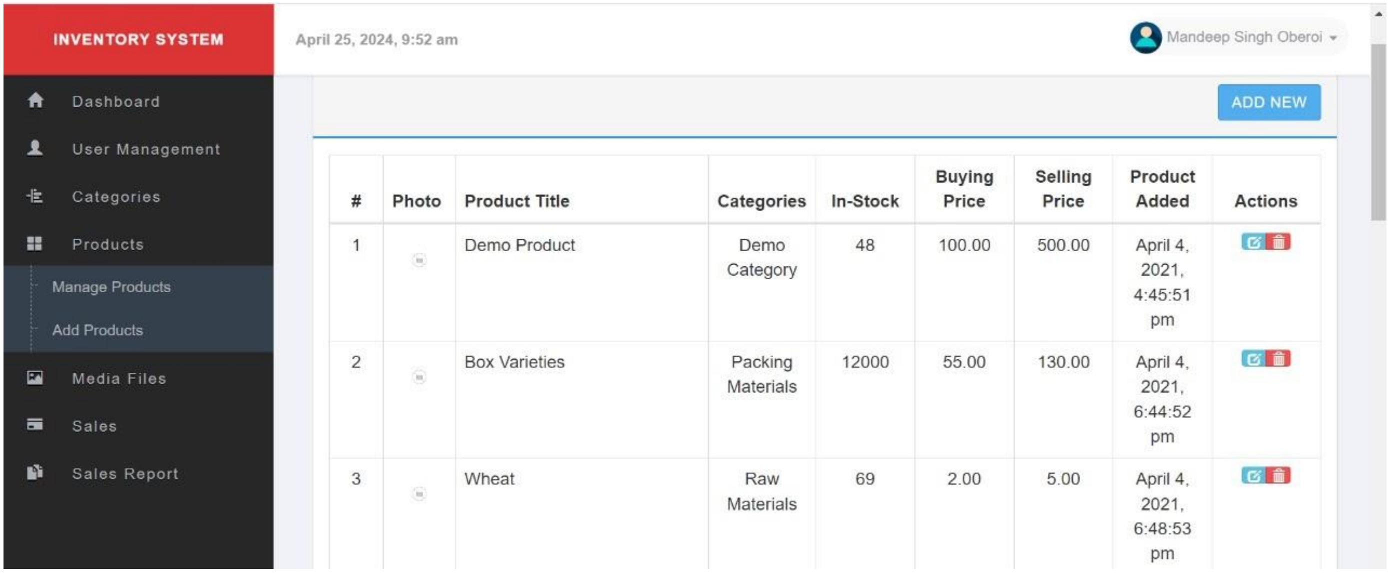
Select Product Category

Select Product Photo

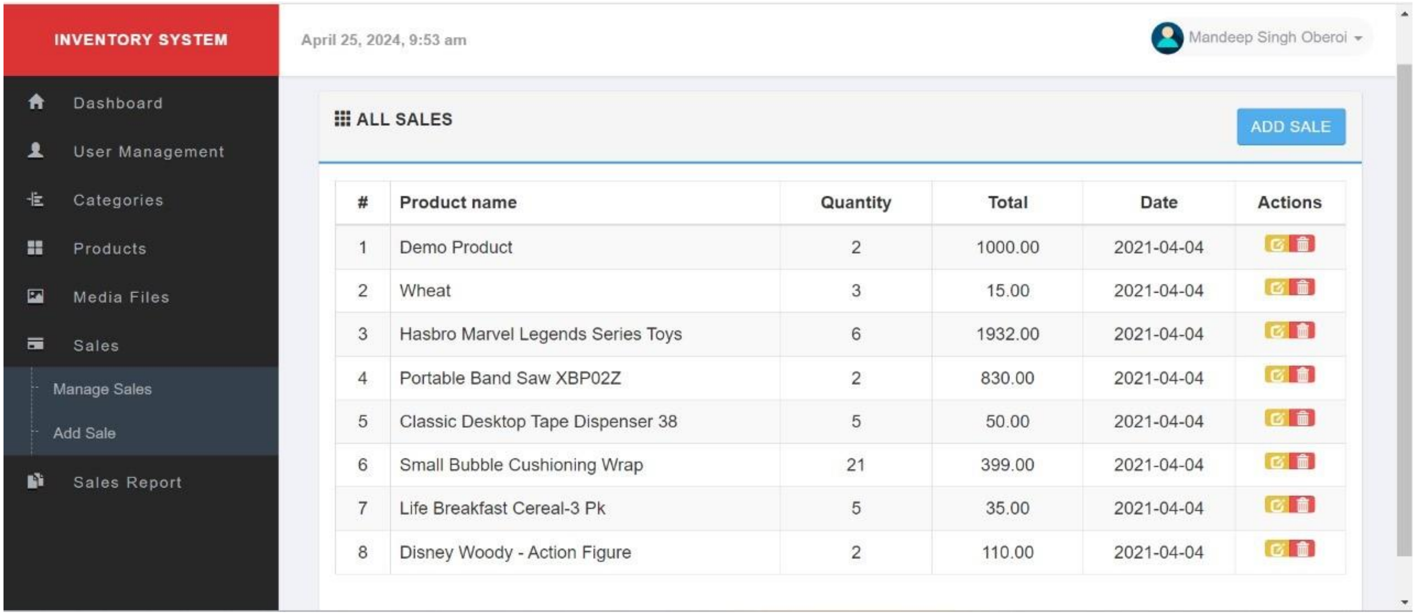
Product Quant

$ Buying .00

$ Selling | .00



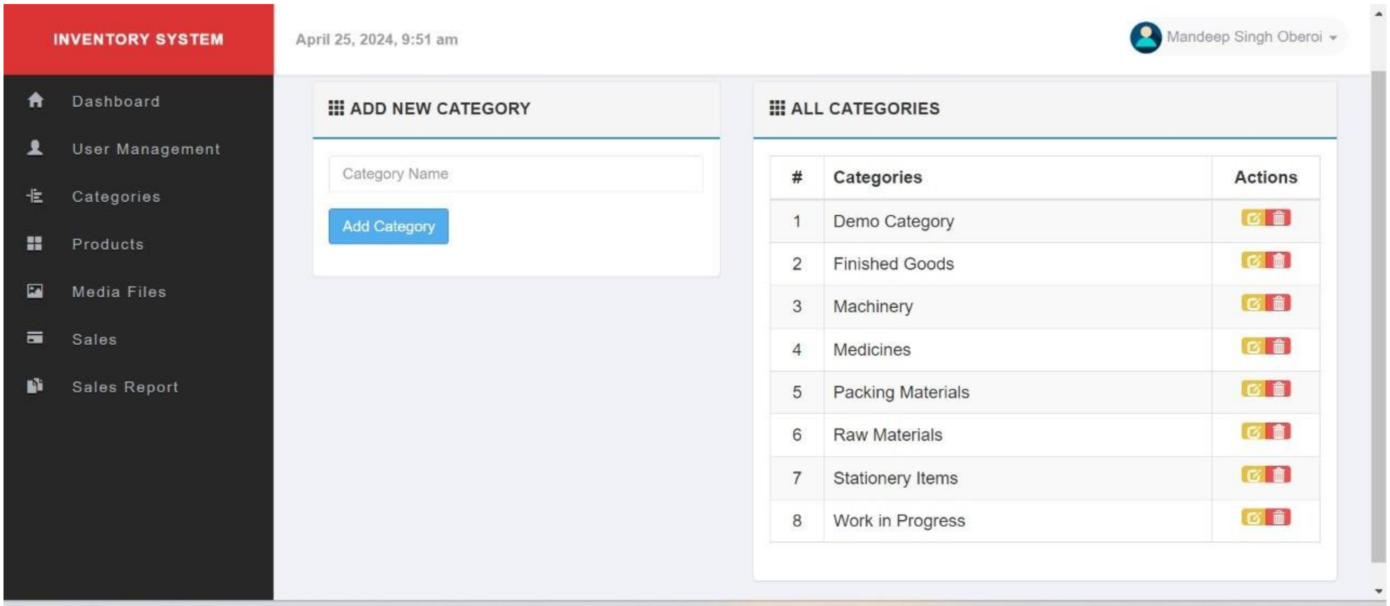
#	Photo	Product Title	Categories	In-Stock	Buying Price	Selling Price	Product Added	Actions
1		Demo Product	Demo Category	48	100.00	500.00	April 4, 2021, 4:45:51 pm	
2		Box Varieties	Packing Materials	12000	55.00	130.00	April 4, 2021, 6:44:52 pm	
3		Wheat	Raw Materials	69	2.00	5.00	April 4, 2021, 6:48:53 pm	



April 25, 2024, 9:53 am

ALL SALES

#	Product name	Quantity	Total	Date	Actions
1	Demo Product	2	1000.00	2021-04-04	✎ 🗑
2	Wheat	3	15.00	2021-04-04	✎ 🗑
3	Hasbro Marvel Legends Series Toys	6	1932.00	2021-04-04	✎ 🗑
4	Portable Band Saw XBP02Z	2	830.00	2021-04-04	✎ 🗑
5	Classic Desktop Tape Dispenser 38	5	50.00	2021-04-04	✎ 🗑
6	Small Bubble Cushioning Wrap	21	399.00	2021-04-04	✎ 🗑
7	Life Breakfast Cereal-3 Pk	5	35.00	2021-04-04	✎ 🗑
8	Disney Woody - Action Figure	2	110.00	2021-04-04	✎ 🗑

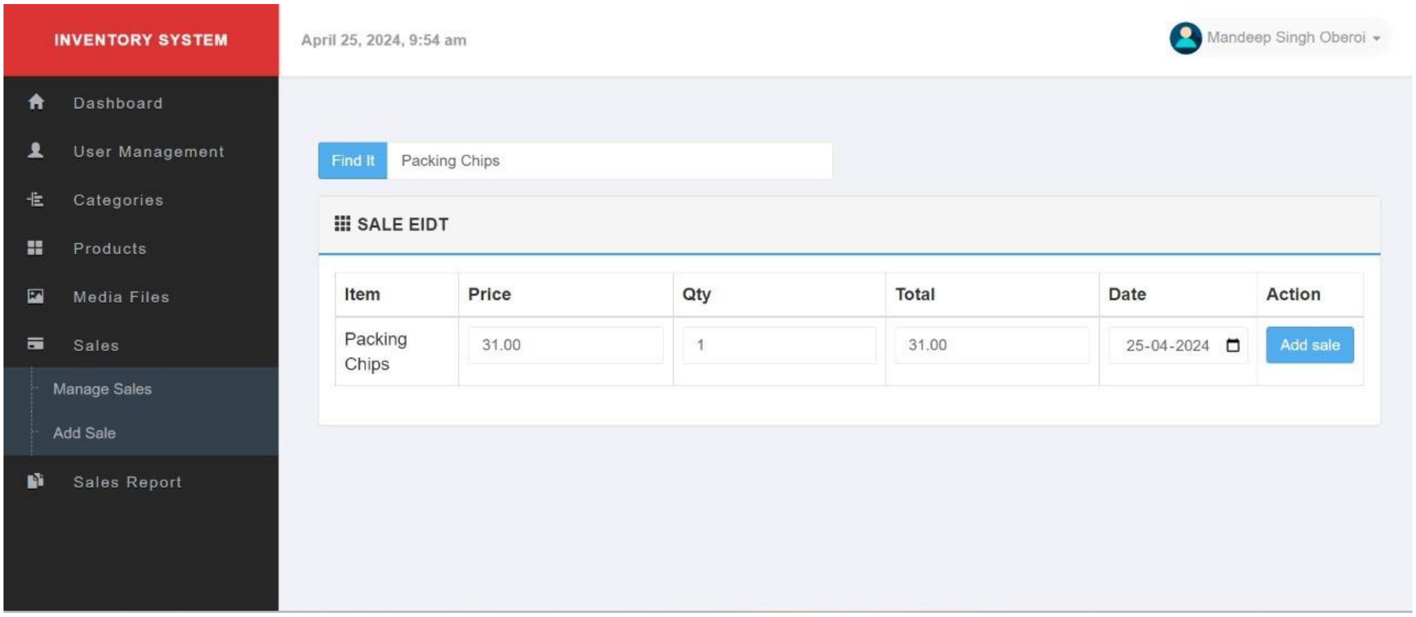


April 25, 2024, 9:51 am

ADD NEW CATEGORY

ALL CATEGORIES

#	Categories	Actions
1	Demo Category	✎ 🗑
2	Finished Goods	✎ 🗑
3	Machinery	✎ 🗑
4	Medicines	✎ 🗑
5	Packing Materials	✎ 🗑
6	Raw Materials	✎ 🗑
7	Stationery Items	✎ 🗑
8	Work in Progress	✎ 🗑

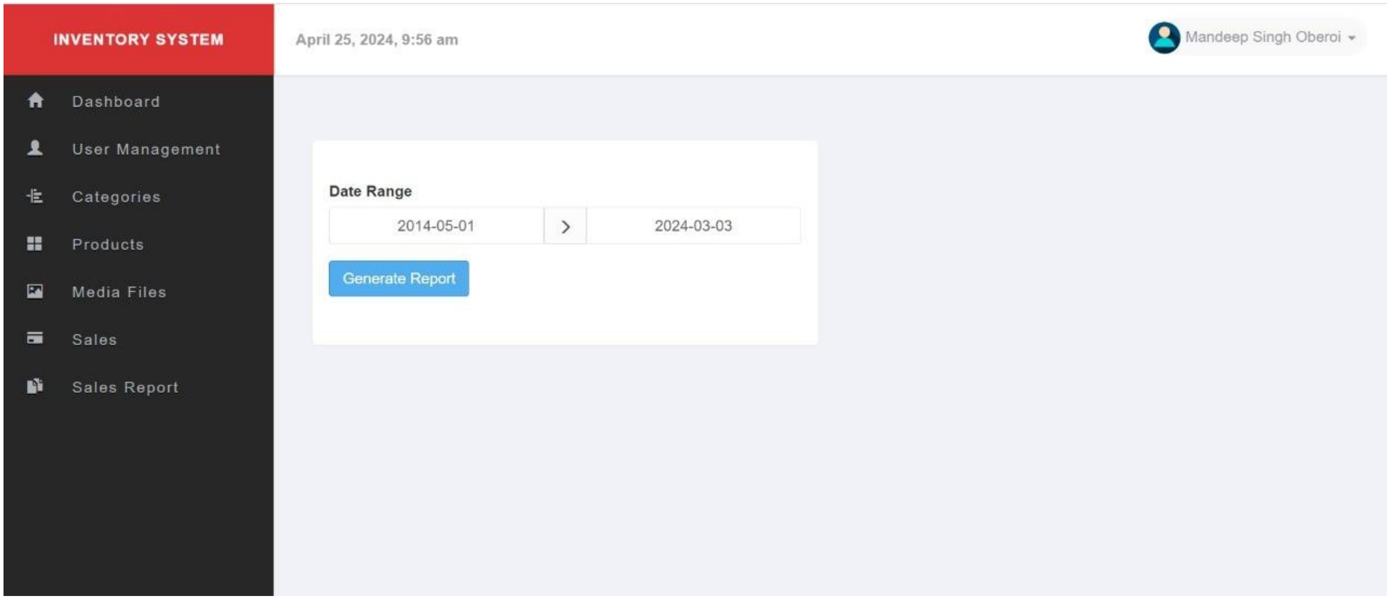


April 25, 2024, 9:54 am

Find It: Packing Chips

SALE EIDT

Item	Price	Qty	Total	Date	Action
Packing Chips	31.00	1	31.00	25-04-2024	Add sale

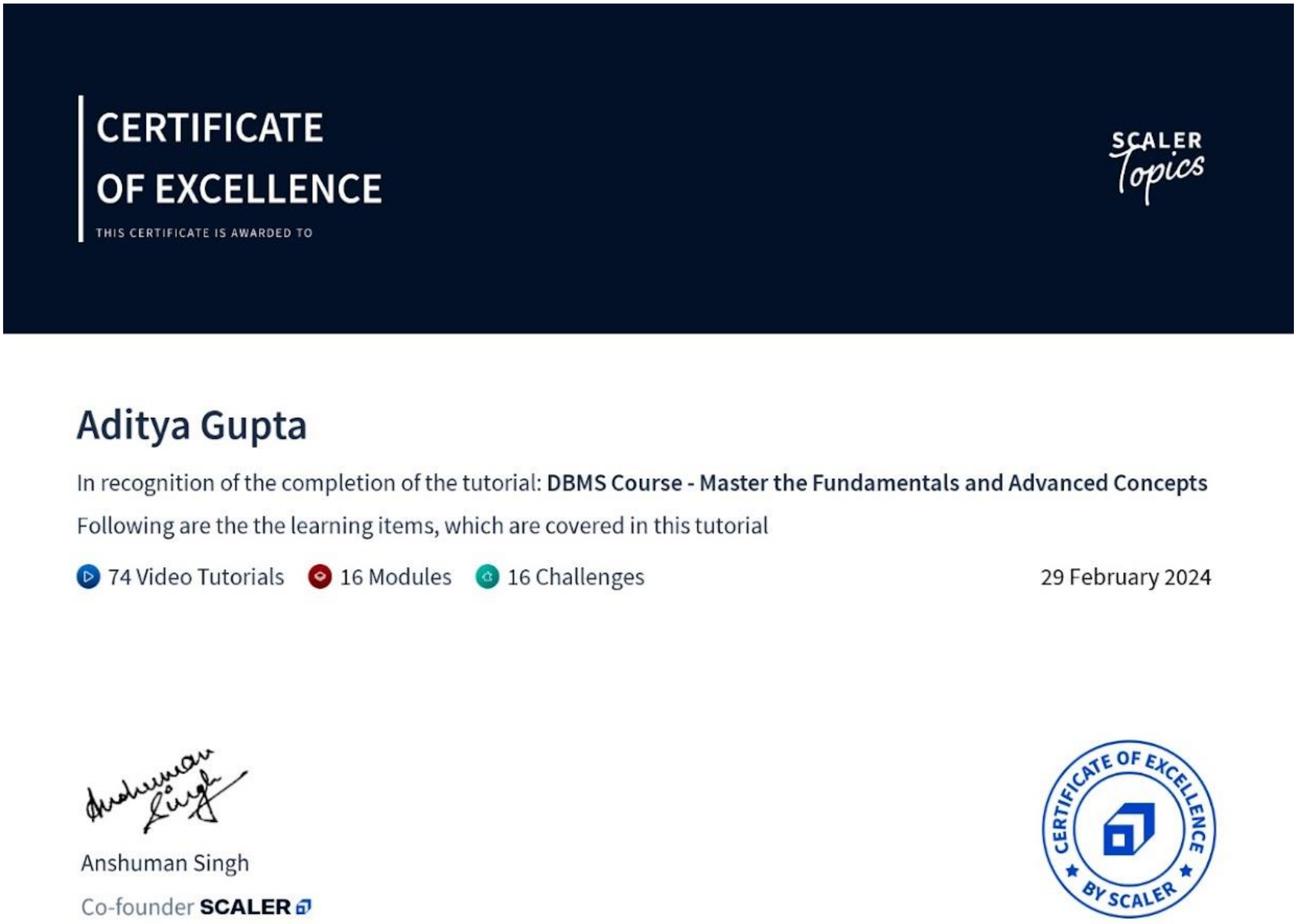


April 25, 2024, 9:56 am

Date Range: 2014-05-01 > 2024-03-03

CHAPTER 8

Online course certificate



CERTIFICATE
OF EXCELLENCE

THIS CERTIFICATE IS AWARDED TO

SCALER Topics

Aditya Gupta

In recognition of the completion of the tutorial: **DBMS Course - Master the Fundamentals and Advanced Concepts**

Following are the the learning items, which are covered in this tutorial

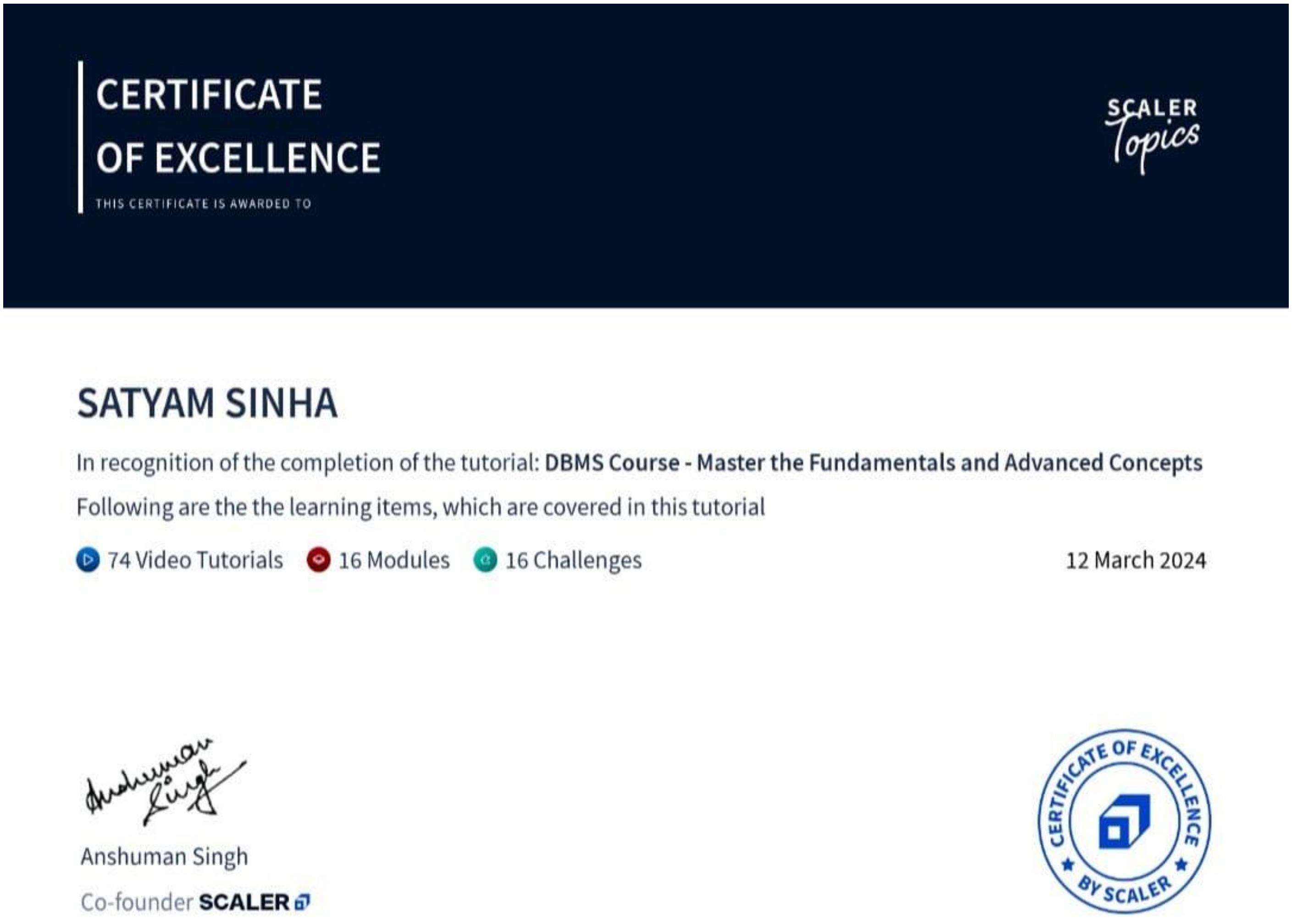
74 Video Tutorials   16 Modules   16 Challenges   29 February 2024

Anshuman Singh

Anshuman Singh

Co-founder **SCALER**

CERTIFICATE OF EXCELLENCE ★ BY SCALER ★



CERTIFICATE OF EXCELLENCE

THIS CERTIFICATE IS AWARDED TO

SCALER Topics

SATYAM SINHA

In recognition of the completion of the tutorial: **DBMS Course - Master the Fundamentals and Advanced Concepts**

Following are the the learning items, which are covered in this tutorial

74 Video Tutorials 16 Modules 16 Challenges

12 March 2024

Anshuman Singh

Anshuman Singh

Co-founder **SCALER**

CERTIFICATE OF EXCELLENCE ★ BY SCALER ★

CHAPTER 9

Conclusion

In conclusion, the implementation of an efficient inventory management system in a database management system (DBMS) not only streamlines operations but also enhances productivity, reduces costs, and improves decision-making processes. By centralizing data, automating tasks, and providing real-time insights, businesses can better manage their inventory levels, forecast demand accurately, minimize stockouts, and optimize their supply chain. As technology continues to evolve, investing in a robust DBMS for inventory management remains a cornerstone for staying competitive in today's dynamic marketplace.