**INVENTORY MANAGEMENT SYSTEM**

A PROJECT REPORT

*Submitted by*

**SANYOG DANI [RA2211031010087]**

**ARUSH SIROTIYA [RA2211031010092]**

**NIKHIL KUMAR [RA2211031010097]**

*Under the guidance of*

**Dayana. D. S**

Assistant Professor, Networking and Communications

*In partial fulfilment of the requirements of the degree of*

**BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE AND ENGINEERING W/S IN INFORMATION TECHNOLOGY**



**DEPARTMENT OF NETWORKING AND COMMUNICATIONS COLLEGE OF ENGINEERING AND TECHNOLOGY SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**

**KATTANKULATHUR - 603203**

**MAY 2024**



**SRM INSTITUTE OF SCIENCE AND TECHNOLOGY KATTANKULATHUR–603 203**

**BONAFIDE CERTIFICATE**

**Register no. RA2211031010087, RA2211031010092, RA2211031010097** Certified to be the bonafide work done by **Sanyog Dani, Arush Sirotiya, Nikhil Kumar** 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 in Charge:**  **Dayana. D. S,**  Assistant Professor, Department of Networking and Communications,  SRMIST-KTR. | **Head of the Department:**  **Dr.AnnapuraniPanaiyappanK,** Professor and Head Department of Networking and Communications,  SRMIST-KTR. |

# ABSTRACT

The Inventory Management System (IMS) is a pivotal tool for organizations seeking efficient management of their inventory operations. IMS offers a comprehensive solution designed to streamline processes, maximize productivity, and enhance overall operational efficiency. At its core, IMS provides a centralized platform encompassing inventory control, procurement, and monitoring functionalities.

One of the key features of IMS is its robust inventory tracking capabilities. By offering real-time visibility into stock levels, locations, and movement, IMS enables organizations to maintain accurate and up-to-date inventory records. This visibility not only facilitates efficient inventory management but also helps in minimizing stockouts and overstock situations, thereby optimizing inventory levels.

IMS allows users to swiftly record incoming and outgoing inventory transactions. This feature not only minimizes errors but also enhances accuracy in inventory management processes. By automating transaction recording, IMS reduces manual effort, mitigates the risk of human error, and ensures data integrity throughout the inventory lifecycle.

In addition to transaction recording, IMS provides advanced reporting and analytics functionalities. Stakeholders can leverage these features to gain valuable insights into inventory performance, identify trends, and make data-driven decisions. From analyzing inventory turnover rates to evaluating supplier performance metrics, IMS empowers organizations to optimize inventory strategies and reduce carrying costs.

# TABLE OF CONTENTS

|  |  |  |
| --- | --- | --- |
| **Chapter No** | **Chapter Name** | **Page No** |
|  | Problem understanding, Identification of Entity and Relationships, Construction of DB using ER Model for the project | **1** |
|  | Design of Relational Schemas, Creation of Database Tables for the project. | **7** |
|  | Complex queries based on the concepts of constraints, sets, joins, views, Triggers and Cursors. | **10** |
|  | Analyzing the pitfalls, identifying the dependencies, and applying normalizations | **22** |
|  | Implementation of concurrency control and recovery mechanisms | **27** |
|  | Code for the project | **31** |
|  | Result and Discussion (Screen shots of the implementation with front end. | **38** |
|  | Attach the Real Time project certificate / Online course certificate | **45** |

**CHAPTER 1 INTRODUCTION**

Inventory management is a critical aspect of operations for businesses across diverse industries. An Inventory Management System (IMS) serves as a pivotal tool in facilitating the efficient handling, monitoring, and optimization of inventory levels within an organization.

The primary objective of an IMS is to ensure that businesses maintain optimal stock levels while minimizing costs associated with excess inventory or stockouts. By providing real-time visibility into inventory data, such as stock quantities, locations, and movement, an IMS enables businesses to make informed decisions regarding procurement, storage, and distribution.

Traditionally, inventory management involved manual processes, such as spreadsheet-based tracking or physical counts. However, the complexity and scale of modern supply chains necessitate more sophisticated solutions.

Key features of an Inventory Management System typically include:

Inventory Tracking: Real-time monitoring of stock levels, allowing businesses to accurately assess inventory positions and respond to fluctuations in demand.

Warehouse Management: Efficient management of warehouse operations, including inventory receipt, storage, picking, packing, and shipping, to streamline logistics processes and minimize errors.

Reporting and Analytics: Comprehensive reporting and analytics capabilities to track key performance indicators analyze inventory trends, and identify opportunities for improvement.

In today's competitive business landscape, effective inventory management is essential for meeting customer demand, optimizing cash flow, and maintaining a competitive edge. An Inventory Management System plays a pivotal role in helping businesses achieve these objectives by providing visibility, control, and efficiency throughout the inventory lifecycle.

Overall, the adoption of an Inventory Management System empowers businesses to enhance operational efficiency, minimize costs, and improve customer satisfaction, ultimately driving sustainable growth and success.

# PROBLEM STATEMENT

In today's dynamic business environment, effective inventory management is crucial for businesses to meet customer demand, minimize costs, and maintain a competitive edge. However, many organizations face challenges in efficiently tracking, managing, and optimizing their inventory operations. Manual processes, outdated systems, and lack of real-time visibility often result in inventory inaccuracies, stockouts, excess inventory, and inefficiencies in procurement and logistics.

The problem statement revolves around the need for a comprehensive Inventory Management System (IMS) that addresses the following challenges:

* Inventory Tracking and Visibility
* Inventory Optimization
* Procurement Efficiency
* Warehouse Management
* Reporting and Analytics

LITERATURE SURVEY

**1. Introduction**

- Define the significance of inventory management in business operations.

- Introduce the purpose of the literature survey, focusing on exploring existing research and methodologies related to implementing inventory management systems using MySQL and PHP.

**2. Historical Overview**

- Trace the historical development of inventory management systems and the evolution of database technologies like MySQL.

- Highlight key milestones and advancements in PHP development relevant to inventory management.

**3. Theoretical Framework**

- Discuss theoretical concepts in inventory management that can be implemented using MySQL and PHP, such as database normalization for efficient data storage and retrieval.

- Explore how PHP scripting language can be used to develop dynamic inventory management interfaces and automate inventory-related tasks.

**4. Types of Inventory Management Systems**

- Review different types of inventory management systems and how they can be implemented using MySQL databases and PHP scripts.

- Compare various inventory management methodologies, such as FIFO, LIFO, and JIT, in the context of MySQL and PHP implementation.

**5. Technological Advances**

- Explore technological advancements in MySQL and PHP that enhance inventory management capabilities, such as transaction handling, stored procedures, and data encryption for security.

- Discuss the integration of web-based interfaces and mobile applications using PHP frameworks like Laravel or CodeIgniter for real-time inventory tracking and management.

**6. Supply Chain Integration**

- Investigate methods for integrating inventory management systems with other components of the supply chain using MySQL and PHP.

- Discuss the use of APIs and web services to facilitate data exchange between inventory systems, suppliers, and distributors.

**7. Challenges and Solutions**

- Identify common challenges in implementing inventory management systems using MySQL and PHP, such as scalability, data consistency, and security.

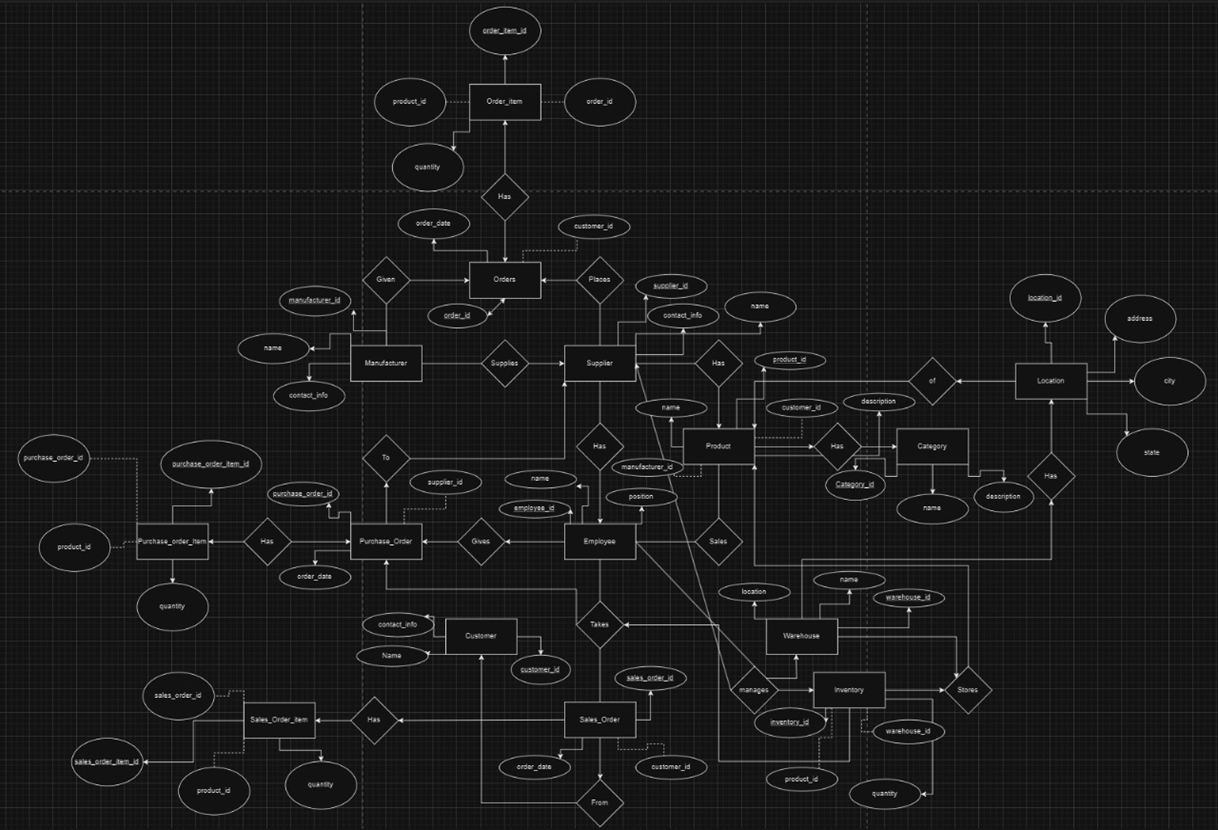
- Discuss best practices and solutions to overcome these challenges, including database optimization techniques and PHP security measures.

**8. Case Studies and Empirical Research**

- Review case studies and empirical research on the implementation of inventory management systems using MySQL and PHP.

- Analyze the effectiveness of different approaches and methodologies in real-world scenarios.

# ER DIAGRAM



# CHAPTER 2

# RELATIONAL SCHEMA

# 1. ims\_brand

# id (Primary Key) ,categoryid (ForeignKey referencing ims\_ category.categoryid) , bname , status

# 2. ims\_category

# categoryid (Primary Key), name , status

# 3. ims\_customer

# id (Primary Key) , name , address , mobile , balance

# 4. ims\_order

# order\_id (Primary Key) , product\_id (Foreign Key referencing ims\_product.pid) , total\_shipped , customer\_id (Foreign Key referencing ims\_customer.id) , order\_date

# 5. ims\_product

# pid (Primary Key) category , id (Foreign Key referencing ims\_category.category) , id brand , id (Foreign Key referencing ims\_brand.id), pname model , base\_price , tax , minimum\_order , supplier , status , date

# 6. ims\_purchase

# purchase\_id (Primary Key) , supplier\_id (Foreign Key referencing ims\_supplier.supplier\_id) , product\_id (Foreign Key referencing ims\_product.pid), quantity purchase\_date

# 7. ims\_supplier,

# supplier\_id (Primary Key), supplier\_name, mobile, address, status

# 8. ims\_user

# userid (Primary Key), email, password, name, type, status

# Foreign keys in tables `ims\_brand`, `ims\_order`, `ims\_product`, `ims\_purchase` reference primary keys in other tables, establishing relationships between them.

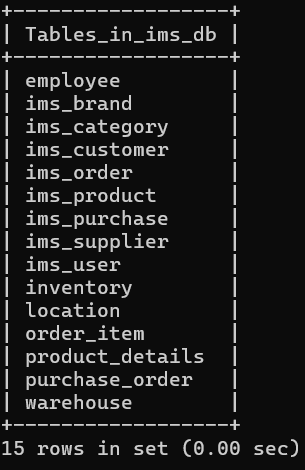
# Each table represents a specific entity and its attributes. Relationships between entities are established through foreign key constraints, ensuring data integrity and consistency.

# 

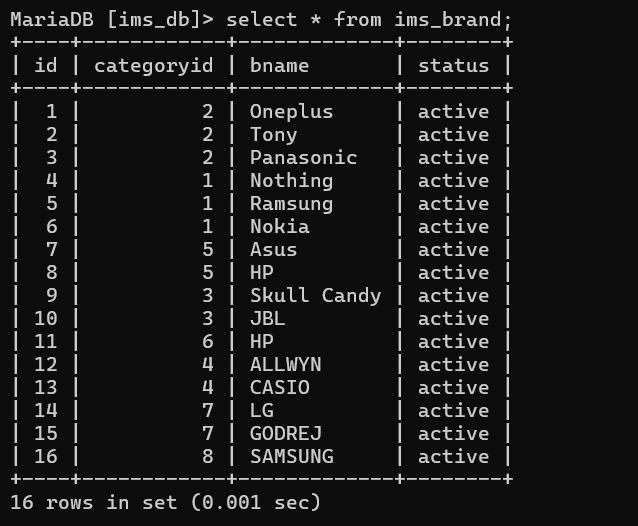
# CHAPTER 3

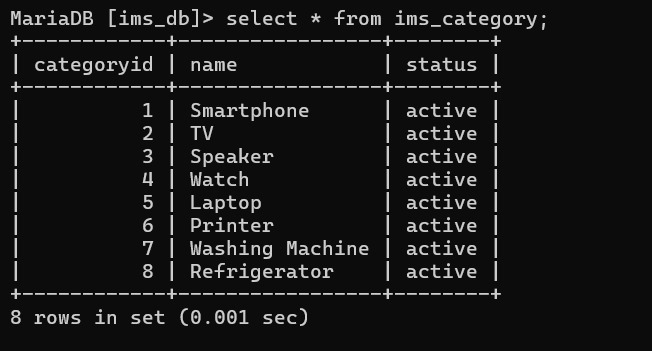
# TABLES

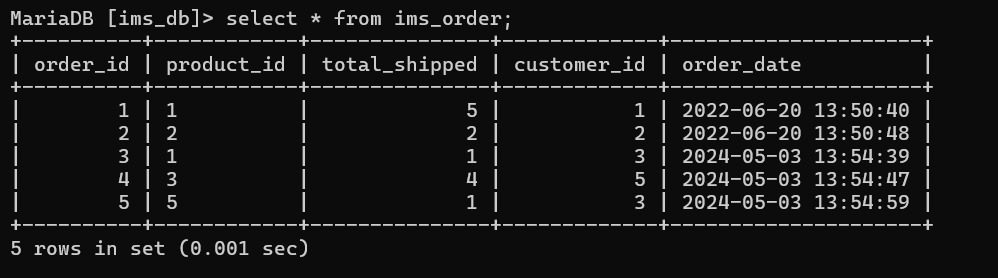
* Database

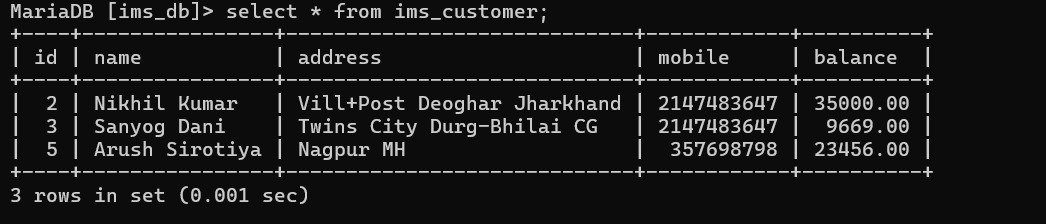


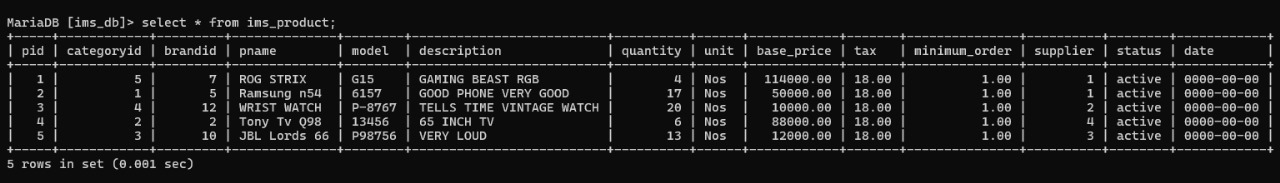
* Tables

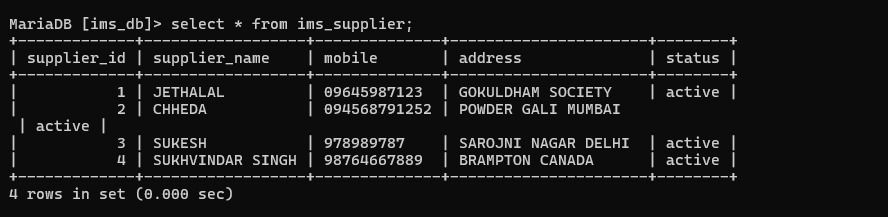


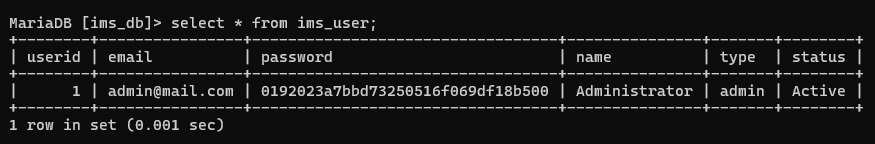
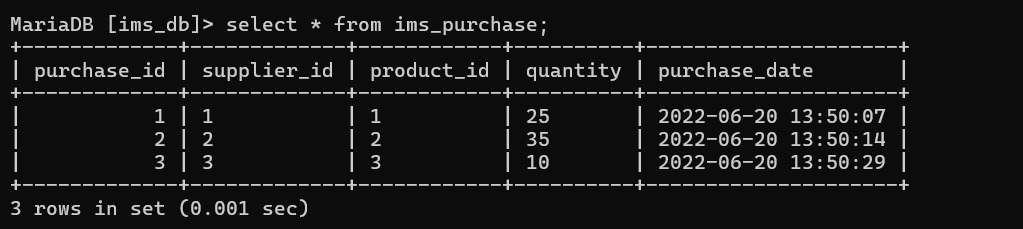


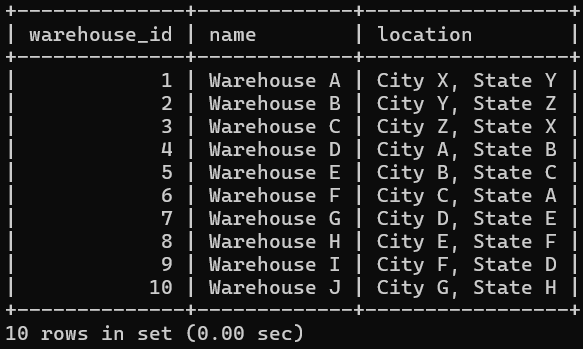


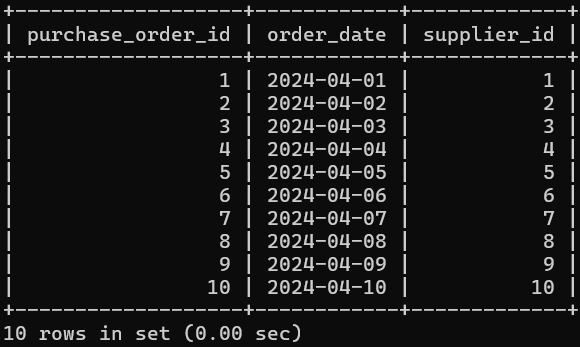


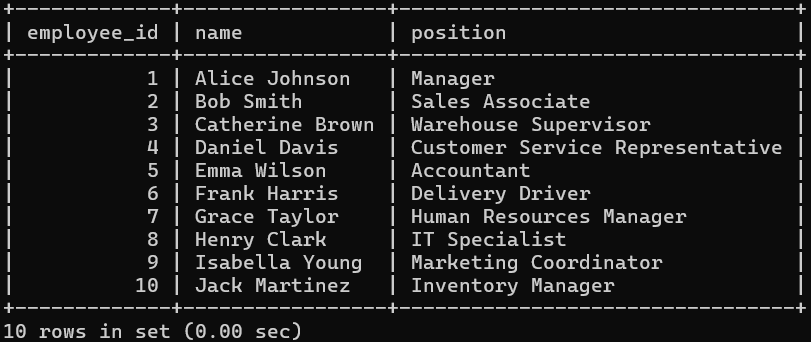




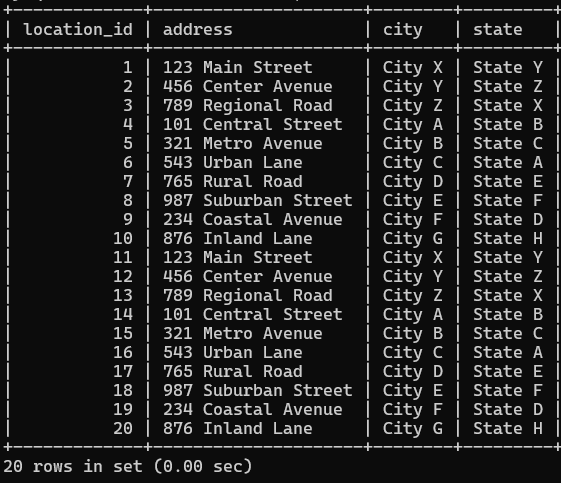


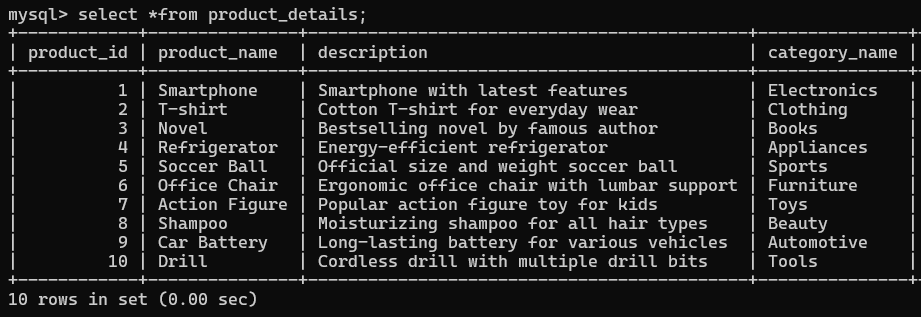


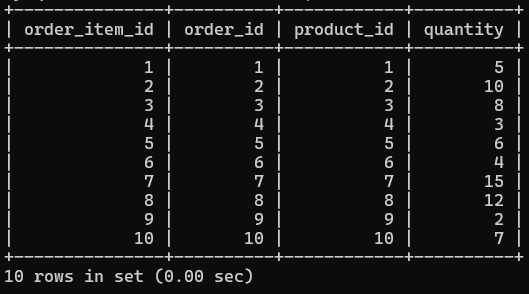










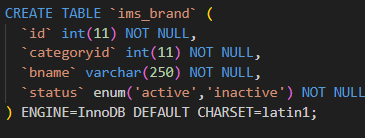


# CONSTRAINTS

Constraints are the rules that we can apply on the type of data in a table. That is, we can specify the limit on the type of data that can be stored in a particular column in a table using constraints.

The constraints used in the IMS project is:

* **NOT NULL**: This constraint tells that we cannot store a null value in a column. That is, if a column is specified as NOT NULL then we will not be able to store null in this particular column any more.
* **PRIMARY KEY**: A primary key is a field which can uniquely identify each row in a table. And this constraint is used to specify a field in a table as primary key.
* **FOREIGN KEY**: A Foreign key is a field which can uniquely identify each row in another table. And this constraint is used to specify a field as foreign key.
* **DEFAULT**: This constraint specifies a default value for the column when no value is specified by the user.



# QUERIES

SQL query is used to query or retrieve information from the databases. These queries can be used to create a new database and insert data into the database, to retrieve (or fetch) data from the database, to modify or update the existing data in the database, to delete or drop the data or table from the database, to set permissions for the tables, views and procedures.

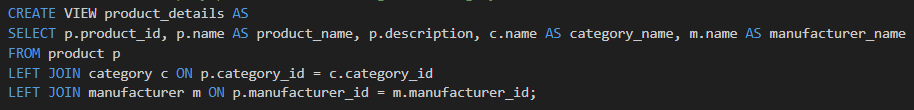
# SUB QUERIES

Subqueries (also known as inner queries or nested queries) are a tool for performing operations in multiple steps. An SQL Subquery, is a SELECT query within another query. It is also known as Inner query or Nested query and the query containing it is the outer query.

The outer query can contain the SELECT, INSERT, UPDATE, and DELETE statements. We can use the subquery as a column expression, as a condition in SQL clauses, and with operators like =, >, <, >=, <=, IN, BETWEEN, etc.

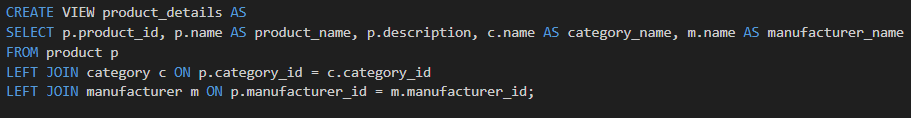
# JOINS

A **join** clause in the Structured Query Language ([SQL](https://en.wikipedia.org/wiki/SQL)) combines [columns](https://en.wikipedia.org/wiki/Column_(database)) from one or more [tables](https://en.wikipedia.org/wiki/Table_(database)) into a new table. The operation corresponds to a [join operation in relational algebra](https://en.wikipedia.org/wiki/Join_(relational_algebra)). Informally, a join stitches two tables and puts on the same row records with matching fields. There are 5 types of joins, namely:



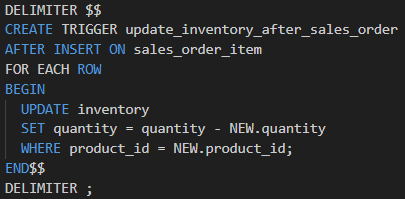
# VIEWS

Views in SQL are a kind of virtual table. A view also has rows and columns like tables, but a view doesn’t store data on the disk like a table. View defines a customized query that retrieves data from one or more tables, and represents the data as if it was coming from a single source. We can create a view by selecting fields from one or more tables present in the database. A View can either have all the rows of a table or specific rows based on certain conditions.



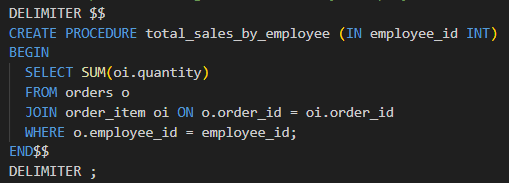
# TRIGGERS

A trigger is a stored procedure in a database that automatically invokes whenever a special event in the database occurs. In simple words, a trigger is a collection of [SQL](https://www.geeksforgeeks.org/sql-tutorial/) statements with particular names that are stored in system memory. It belongs to a specific class of stored procedures that are automatically invoked in response to database server events. Every trigger has a table attached to it.



# CURSORS

A cursor in SQL Server is a database object that allows us to retrieve each row at a time and manipulate its data**.** A cursor is nothing more than a pointer to a row. It's always used in conjunction with a SELECT statement. It is usually a collection of [SQL](https://www.javatpoint.com/sql-tutorial) logic that loops through a predetermined number of rows one by one.



CHAPTER 4

# ANALYZING PITFALLS

**PITFALLS:**

* Security Vulnerabilities: Inadequate access controls, encryption, or other security measures can lead to unauthorized access, data breaches, or malicious activities. Example: Weak password policies, lack of encryption for sensitive data like passwords or personal information, and insufficient protection against SQL injection attacks.
* Scalability Issues: The database may struggle to handle increased load, data volume, or user concurrency as the application grows. Example: Poorly optimized queries, lack of indexing, and inefficient database schema design can result in performance degradation and scalability limitations.
* Insufficient Maintenance and Updates: Neglecting regular maintenance tasks such as database backups, software updates, and performance optimizations can lead to system instability, data loss, or security vulnerabilities. Example: Failure to regularly update database software and security patches, infrequent backups, and inadequate monitoring of database performance and health.
* Lack of Analytics and Insights: Without proper analytics capabilities, the organization may miss opportunities for data-driven decision- making, performance optimization, and identification of trends or patterns. Example: Absence of tools or processes for analyzing database usage, query performance, user behavior, or business metrics, leading to missed insights and potential inefficiencies.

# 

# FUNCTIONAL DEPENDENCIES

# In the "Brand" table, attributes like "brand\_name" and "status" are functionally dependent on the "brand\_id". This means each brand ID uniquely determines its corresponding name and status, ensuring consistency and integrity in brand records.

# Similarly, in the "Category" table, "category\_name" and "status" depend functionally on the "category\_id". Each category ID uniquely determines its name and status, maintaining coherence in category information.

# Moving to the "Customer" table, attributes such as "customer\_name", "address", "mobile", and "balance" depend on the "customer\_id". This dependency ensures each customer ID uniquely corresponds to its name, address, mobile number, and balance, maintaining accurate customer records.

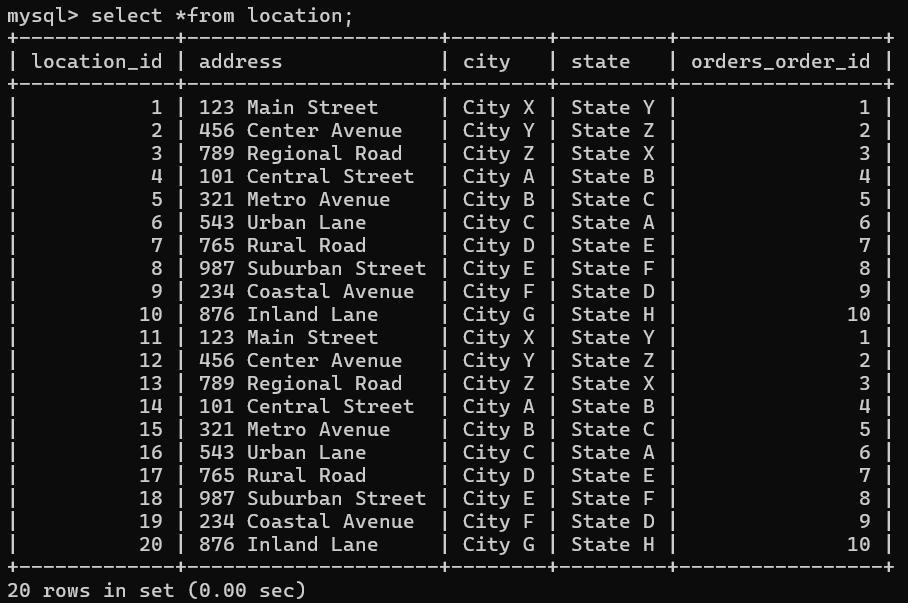
# In the "Order" table, "customer\_id" and "order\_date" attributes are functionally dependent on the "order\_id". Each order ID uniquely determines the customer ID and order date, facilitating efficient order tracking and management.

# In the "Product" table, attributes like "product\_name", "quantity", "unit", "base\_price", and "supplier\_id" depend on the "product\_id". Each product ID uniquely determines these attributes, ensuring accurate representation and identification of products within the system.

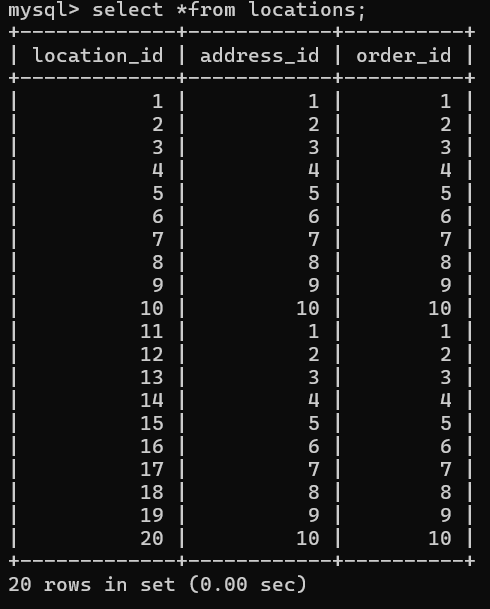
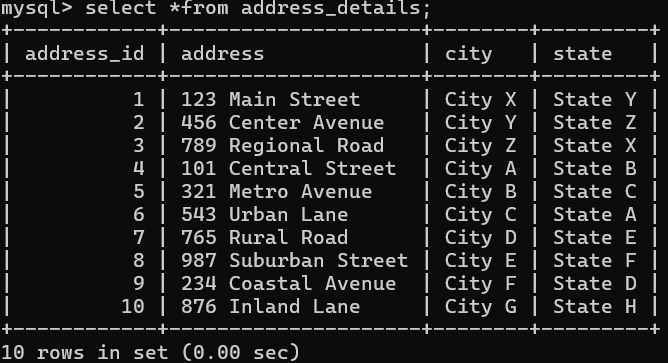
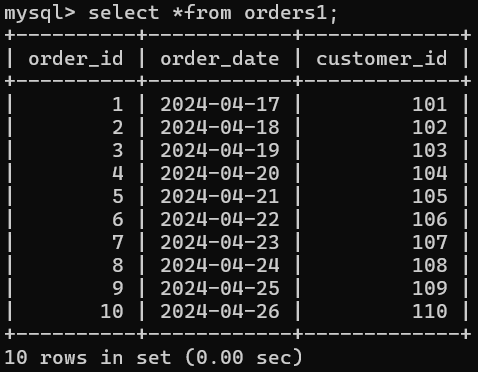
# Overall, functional dependencies ensure data integrity and consistency in the inventory management system by establishing clear relationships between primary keys and other attributes, enhancing system reliability and effectiveness.

NORMALIZATION

Victim table before normalization:



After applying normalization:

CHAPTER 5

Implementation of concurrency control and recovery mechanisms

MODULES USED:

**1. Database Setup:**

The Inventory Management System (IMS) is implemented using PHP and MySQL. To begin with, a MySQL database named "ims\_db" is created using phpMyAdmin, which is part of the XAMPP stack. XAMPP is a free and open-source cross-platform web server solution stack package developed by Apache Friends. It consists mainly of the Apache HTTP Server, MariaDB database, and interpreters for scripts written in PHP .

**2. XAMPP Control Panel:**

XAMPP provides a user-friendly control panel that allows easy management of the Apache server, MySQL database. It can be accessed by starting XAMPP and navigating to http://localhost/phpmyadmin/ to access phpMyAdmin for database management.

**3. Table Structure:** The database structure consists of several tables to store various aspects of inventory management:

* Home Table: This table stores the home page content of the system, including banners, featured products, and promotions.
* Customer Table: Information about customers is stored in this table, including customer ID, name, contact information, and address.
* Category Table: Categories for products are stored in this table, including category ID and name.
* Brand Table: Brands for products are stored in this table, including brand ID and name.
* Product Table: This table stores information about the products in the inventory, including product ID, name, description, quantity, price, category ID, brand ID, and supplier ID.
* Supplier Table: Information about suppliers is stored in this table, including supplier ID, name, contact information, and address.
* Purchase Table: All purchase transactions are recorded in this table, including purchase ID, product ID, quantity, purchase date, and supplier ID.
* Order Table: All sales orders are recorded in this table, including order ID, customer ID, product ID, quantity, and order date.

**4. PHP Implementation:**

The Inventory Management System is implemented using PHP for server-side scripting. PHP scripts interact with the MySQL database to perform various functions such as adding, updating, and deleting products, managing suppliers, recording transactions, managing orders, and user authentication.

**6. Command Prompt Usage:**

* Open XAMPP Control Panel.
* Start the Apache and MySQL services.
* Navigate to http://localhost/phpmyadmin/ to access phpMyAdmin for database management.
* Open a web browser and navigate to the directory where the PHP files are located.
* Access the Inventory Management System by typing http://localhost/ in the browser.

**5. MySQL Database Connectivity:**

PHP scripts establish a connection to the MySQL database using the MySQL or PDO extension. The connection details such as server name, username, password are specified in the PHP scripts.

**7. Concurrency Module:**

To handle concurrency issues such as simultaneous access to the database by multiple users, the IMS incorporates a concurrency control module. This module ensures data consistency and prevents anomalies such as lost updates, uncommitted data, and inconsistent reads.

The Inventory Management System implemented using PHP and MySQL, with the support of XAMPP, provides a solution for efficient inventory management. By leveraging PHP for server-side scripting, MySQL for database management, and XAMPP for local server environment setup, the system ensures reliable performance, scalability, and security. With XAMPP's user-friendly control panel, managing the server environment and database becomes seamless, making it an ideal solution for local development and testing. The concurrency module ensures data consistency and prevents anomalies, allowing multiple users to access the system simultaneously without risking data integrity. Together, these technologies enable organizations to streamline inventory processes, maximize productivity, and enhance overall operational efficiency.

CHAPTER 6

CODE OF THE PROJECT

SET SQL\_MODE = "NO\_AUTO\_VALUE\_ON\_ZERO";

START TRANSACTION;

SET time\_zone = "+00:00";

--

-- Database: `ims\_db`

--

-- --------------------------------------------------------

--

-- Table structure for table `ims\_brand`

--

CREATE TABLE `ims\_brand` (

`id` int(11) NOT NULL,

`categoryid` int(11) NOT NULL,

`bname` varchar(250) NOT NULL,

`status` enum('active','inactive') NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--

-- Dumping data for table `ims\_brand`

--

INSERT INTO `ims\_brand` (`id`, `categoryid`, `bname`, `status`) VALUES

(1, 2, 'Brand 1', 'active'),

(2, 2, 'Brand 2', 'active'),

(3, 2, 'Brand 3', 'active'),

(4, 1, 'Brand 201', 'active'),

(5, 1, 'Brand 202', 'active'),

(6, 1, 'Brand 203', 'active'),

(7, 3, 'Brand 301', 'active'),

(8, 3, 'Brand 302', 'active'),

(9, 3, 'Brand 303', 'active');

-- --------------------------------------------------------

--

-- Table structure for table `ims\_category`

--

CREATE TABLE `ims\_category` (

`categoryid` int(11) NOT NULL,

`name` varchar(250) NOT NULL,

`status` enum('active','inactive') NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--

-- Dumping data for table `ims\_category`

--

INSERT INTO `ims\_category` (`categoryid`, `name`, `status`) VALUES

(1, 'Smartphone', 'active'),

(2, 'Random Item', 'active'),

(3, 'Speaker', 'active');

-- --------------------------------------------------------

--

-- Table structure for table `ims\_customer`

--

CREATE TABLE `ims\_customer` (

`id` int(11) NOT NULL,

`name` varchar(200) NOT NULL,

`address` text NOT NULL,

`mobile` int(50) NOT NULL,

`balance` double(10,2) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--

-- Dumping data for table `ims\_customer`

--

INSERT INTO `ims\_customer` (`id`, `name`, `address`, `mobile`, `balance`) VALUES

(1, 'Mark Cooper', 'Sample Address', 2147483647, 25000.00),

(2, 'George Wilson', '2306 St, Here There', 2147483647, 35000.00);

-- --------------------------------------------------------

--

-- Table structure for table `ims\_order`

--

CREATE TABLE `ims\_order` (

`order\_id` int(11) NOT NULL,

`product\_id` varchar(255) NOT NULL,

`total\_shipped` int(11) NOT NULL,

`customer\_id` int(11) NOT NULL,

`order\_date` timestamp NOT NULL DEFAULT current\_timestamp()

) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--

-- Dumping data for table `ims\_order`

--

INSERT INTO `ims\_order` (`order\_id`, `product\_id`, `total\_shipped`, `customer\_id`, `order\_date`) VALUES

(1, '1', 5, 1, '2022-06-20 08:20:40'),

(2, '2', 3, 2, '2022-06-20 08:20:48');

-- --------------------------------------------------------

--

-- Table structure for table `ims\_product`

--

CREATE TABLE `ims\_product` (

`pid` int(11) NOT NULL,

`categoryid` int(11) NOT NULL,

`brandid` int(11) NOT NULL,

`pname` varchar(300) NOT NULL,

`model` varchar(255) NOT NULL,

`description` text NOT NULL,

`quantity` int(11) NOT NULL,

`unit` varchar(150) NOT NULL,

`base\_price` double(10,2) NOT NULL,

`tax` decimal(4,2) NOT NULL,

`minimum\_order` double(10,2) NOT NULL,

`supplier` int(11) NOT NULL,

`status` enum('active','inactive') NOT NULL,

`date` date NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--

-- Dumping data for table `ims\_product`

--

INSERT INTO `ims\_product` (`pid`, `categoryid`, `brandid`, `pname`, `model`, `description`, `quantity`, `unit`, `base\_price`, `tax`, `minimum\_order`, `supplier`, `status`, `date`) VALUES

(1, 2, 1, 'Product 101', 'P-1001', 'usce auctor faucibus efficitur.', 10, 'Bottles', 500.00, '12.00', 1.00, 1, 'active', '0000-00-00'),

(2, 1, 4, 'Product 102', 'P-1002', 'Proin vehicula mi pulvinar ipsum ornare tincidunt.', 15, 'Box', 7500.00, '12.00', 1.00, 2, 'active', '0000-00-00'),

(3, 3, 7, 'Product 103', 'P-1003', 'Integer interdum, odio eget mattis venenatis', 20, 'Bags', 350.00, '12.00', 1.00, 3, 'active', '0000-00-00');

-- --------------------------------------------------------

--

-- Table structure for table `ims\_purchase`

--

CREATE TABLE `ims\_purchase` (

`purchase\_id` int(11) NOT NULL,

`supplier\_id` varchar(255) NOT NULL,

`product\_id` varchar(255) NOT NULL,

`quantity` varchar(255) NOT NULL,

`purchase\_date` timestamp NOT NULL DEFAULT current\_timestamp()

) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--

-- Dumping data for table `ims\_purchase`

--

INSERT INTO `ims\_purchase` (`purchase\_id`, `supplier\_id`, `product\_id`, `quantity`, `purchase\_date`) VALUES

(1, '1', '1', '25', '2022-06-20 08:20:07'),

(2, '2', '2', '35', '2022-06-20 08:20:14'),

(3, '3', '3', '10', '2022-06-20 08:20:29');

-- --------------------------------------------------------

--

-- Table structure for table `ims\_supplier`

--

CREATE TABLE `ims\_supplier` (

`supplier\_id` int(11) NOT NULL,

`supplier\_name` varchar(200) NOT NULL,

`mobile` varchar(50) NOT NULL,

`address` text NOT NULL,

`status` enum('active','inactive') NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--

-- Dumping data for table `ims\_supplier`

--

INSERT INTO `ims\_supplier` (`supplier\_id`, `supplier\_name`, `mobile`, `address`, `status`) VALUES

(1, 'Supplier 101', '09645987123', 'Over Here', 'active'),

(2, 'Supplier 102', '094568791252', 'Over There', 'active'),

(3, 'Supplier 103', '09789897879', 'Anywhere There', 'active');

-- --------------------------------------------------------

--

-- Table structure for table `ims\_user`

--

CREATE TABLE `ims\_user` (

`userid` int(11) NOT NULL,

`email` varchar(200) NOT NULL,

`password` varchar(200) NOT NULL,

`name` varchar(200) NOT NULL,

`type` enum('admin','member') NOT NULL,

`status` enum('Active','Inactive') NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=latin1;

--

-- Dumping data for table `ims\_user`

--

INSERT INTO `ims\_user` (`userid`, `email`, `password`, `name`, `type`, `status`) VALUES

(1, '[admin@mail.com](mailto:admin@mail.com)', '0192023a7bbd73250516f069df18b500', 'Administrator', 'admin', 'Active');

--

-- Indexes for dumped tables

--

--

-- Indexes for table `ims\_brand`

--

ALTER TABLE `ims\_brand`

ADD PRIMARY KEY (`id`);

--

-- Indexes for table `ims\_category`

--

ALTER TABLE `ims\_category`

ADD PRIMARY KEY (`categoryid`);

--

-- Indexes for table `ims\_customer`

--

ALTER TABLE `ims\_customer`

ADD PRIMARY KEY (`id`);

--

-- Indexes for table `ims\_order`

--

ALTER TABLE `ims\_order`

ADD PRIMARY KEY (`order\_id`);

--

-- Indexes for table `ims\_product`

--

ALTER TABLE `ims\_product`

ADD PRIMARY KEY (`pid`);

--

-- Indexes for table `ims\_purchase`

--

ALTER TABLE `ims\_purchase`

ADD PRIMARY KEY (`purchase\_id`);

--

-- Indexes for table `ims\_supplier`

--

ALTER TABLE `ims\_supplier`

ADD PRIMARY KEY (`supplier\_id`);

--

-- Indexes for table `ims\_user`

--

ALTER TABLE `ims\_user`

ADD PRIMARY KEY (`userid`);

--

-- AUTO\_INCREMENT for dumped tables

--

--

-- AUTO\_INCREMENT for table `ims\_brand`

--

ALTER TABLE `ims\_brand`

MODIFY `id` int(11) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=10;

--

-- AUTO\_INCREMENT for table `ims\_category`

--

ALTER TABLE `ims\_category`

MODIFY `categoryid` int(11) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=4;

-- AUTO\_INCREMENT for table `ims\_customer`

--

ALTER TABLE `ims\_customer`

MODIFY `id` int(11) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=3;

--

-- AUTO\_INCREMENT for table `ims\_order`

--

ALTER TABLE `ims\_order`

MODIFY `order\_id` int(11) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=3;

--

-- AUTO\_INCREMENT for table `ims\_product`

--

ALTER TABLE `ims\_product`

MODIFY `pid` int(11) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=4;

--

-- AUTO\_INCREMENT for table `ims\_purchase`

--

ALTER TABLE `ims\_purchase`

MODIFY `purchase\_id` int(11) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=4;

--

-- AUTO\_INCREMENT for table `ims\_supplier`

--

ALTER TABLE `ims\_supplier`

MODIFY `supplier\_id` int(11) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=4;

--

-- AUTO\_INCREMENT for table `ims\_user`

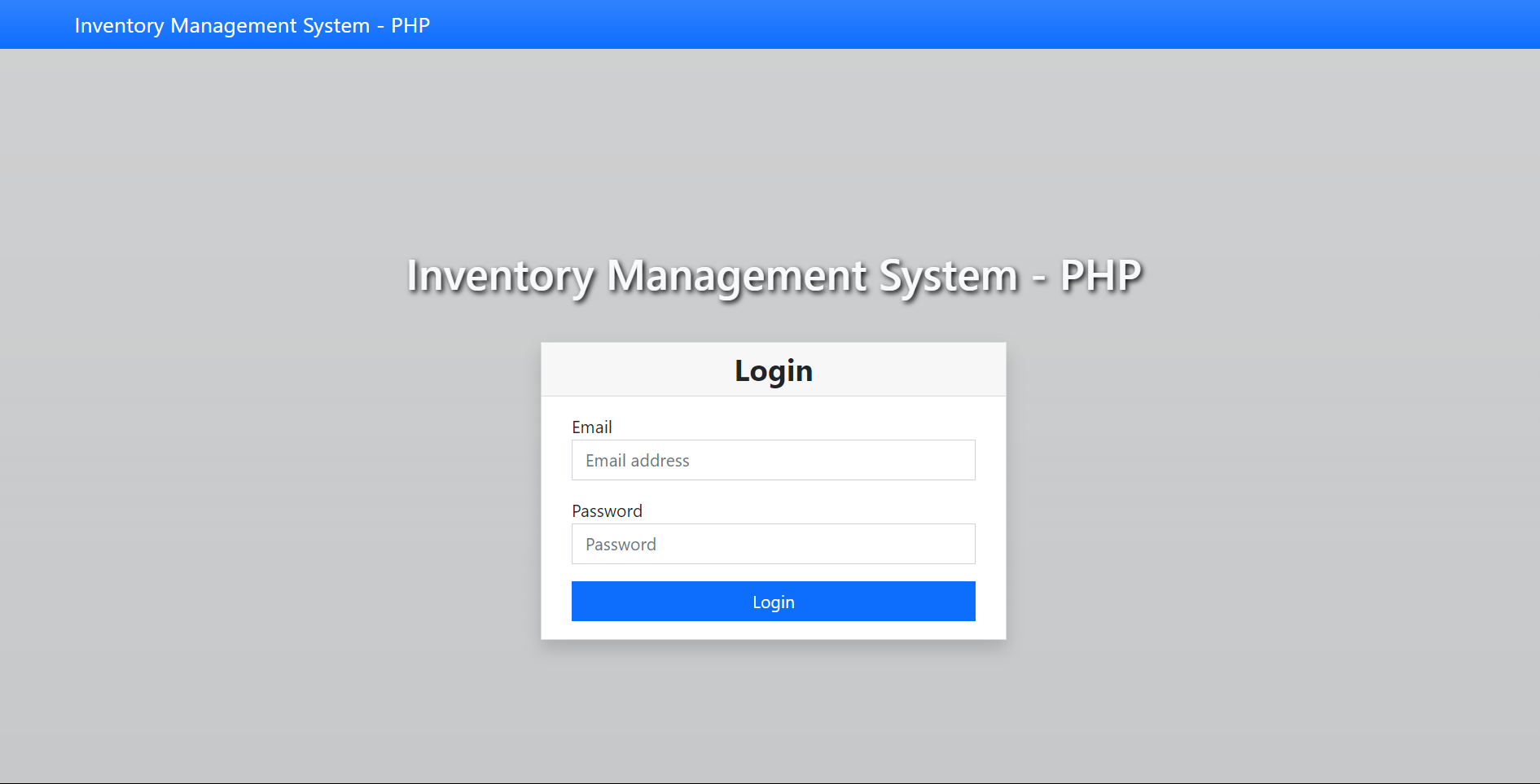
--

ALTER TABLE `ims\_user`

MODIFY `userid` int(11) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=3;

COMMIT;

CHAPTER 7 RESULTS & SCREENSHOTS



# 

# Inserting image...

# Inserting image...

# Inserting image...

# Inserting image...

# Inserting image...

# Inserting image...

# Inserting image...

# CONCLUSION

In conclusion, the Inventory Management System (IMS) stands as a pivotal solution for organizations seeking to overcome the challenges associated with inventory management in today's dynamic business environment. IMS offers a comprehensive platform designed to streamline processes, maximize productivity, and enhance overall operational efficiency.

With its robust inventory tracking capabilities, IMS provides real-time visibility into stock levels, locations, and movement, enabling organizations to maintain accurate and up-to-date inventory records. This visibility not only facilitates efficient inventory management but also helps minimize stockouts and overstock situations, thereby optimizing inventory levels.

IMS allows users to swiftly record incoming and outgoing inventory transactions, minimizing errors and enhancing accuracy in inventory management processes. By automating transaction recording, IMS reduces manual effort, mitigates the risk of human error, and ensures data integrity throughout the inventory lifecycle.

Furthermore, IMS provides advanced reporting, empowering stakeholders to gain valuable insights into inventory performance, identify trends, and make data-driven decisions. From analyzing inventory turnover rates to evaluating supplier performance metrics, IMS enables organizations to optimize all inventory strategies.

In today's competitive business landscape, effective inventory management is essential for meeting customer demand, optimizing cash flow, and maintaining a competitive edge. An Inventory Management System plays a crucial role in helping businesses achieve these objectives by providing visibility, control, and efficiency throughout the inventory lifecycle.

By adopting an Inventory Management System, businesses can enhance operational efficiency, minimize costs, and improve customer satisfaction, ultimately driving sustainable growth and success. IMS addresses the challenges of inventory tracking and visibility, inventory optimization, procurement efficiency, warehouse management, reporting, and analytics, ensuring that organizations can effectively manage their inventory operations and stay ahead in the marketplace.

CHAPTER 8

ONLINE COURSE CERTIFICATION



