UNIVERSITY INSTITUTE of COMPUTING
Asia's Fastest Growing University

CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

NAAC GRADE A+
ACCREDITED UNIVERSITY

PROJECT REPORT
on
Inventory Management System

Program Name : BCA(DS)
Subject Name : Database Management System Lab
Code : 24CAP-204

Submitted By -
Sanskriti Kapoor
24BCD10039
24BCD1-A

Submitted To -
Suraj Parkash
Assistant Professor

Teacher’s Signature

Project Report : Inventory Management System

Aim of the Project

The aim of the **Inventory Management System** project is to **design and implement a database** that efficiently manages information related to products, suppliers, stock levels, and purchase records of an organization.

It seeks to automate the process of **inventory tracking, reduce manual errors, maintain data accuracy,** and **provide easy access** to information for effective decision-making.

Objectives of the Project

1. To design and develop a **relational database** that effectively manages all inventory-related information, including products, suppliers, stock levels, and purchase transactions.
2. To **streamline the inventory management process** by replacing manual record-keeping with an automated, reliable database system.
3. To maintain **data integrity** and **consistency** through the use of proper database constraints such as primary keys, foreign keys, and normalization up to 3NF.
4. To enable quick and accurate retrieval of information using SQL queries involving **SELECT, WHERE, JOIN, GROUP BY,** and **HAVING** clauses.
5. To provide **real-time visibility** into stock availability, reorder levels, and supplier performance.
6. To **generate useful reports** such as stock summaries, purchase histories, and profit or loss analysis for better decision-making.
7. To **enhance data security** and **minimize redundancy** through relational database principles and constraints.

Tools and Technologies Used

- **Database:** MySQL — for creating and managing the database.
- **Language:** SQL — for performing data definition and manipulation operations.
- **Design Tool:** Draw.io / MySQL Workbench — for creating the E–R diagram.
- **Development Environment:** VS Code / MySQL Workbench — for writing and executing queries.
- **Operating System:** Windows — used for developing and testing the project.

System Requirements

Hardware Requirements

- **Processor:** Intel i3 or above
- **RAM:** Minimum 4 GB
- **Hard Disk:** At least 500 MB free space
- **Monitor:** 15" or higher resolution display

Software Requirements

- **Operating System:** Windows / Linux
- **Database:** MySQL / Oracle / PostgreSQL
- **IDE / Editor:** MySQL Workbench / VS Code / Oracle SQL Developer
- **Design Tool:** Draw.io / ERDPlus for E–R Diagram
- **Language:** SQL

Scope of the Project

The **Inventory Management System** provides an efficient and user-friendly platform for managing and analyzing inventory data.

It is **suitable for small and medium-sized retail businesses, wholesalers, and departmental stores** that require digital record-keeping of stock and suppliers.

Scope Includes:

- Adding, updating, and managing product and supplier details.
- Tracking stock availability, purchase cost, and sales data.
- Maintaining supplier and purchase records.
- Generating inventory reports for analysis and decision-making.
- Ensuring data consistency and accuracy through relational database design and normalization.
- Supporting quick retrieval of data through SQL queries and views.

Limitations of the Project

- The system is **limited to basic inventory management** and does not include advanced features like billing, real-time analytics, or barcode integration.
- **User authentication and security features** are minimal or not implemented.
- The project is designed for **single-user access**, not for concurrent multi-user environments.
- It requires manual data entry for updates and transactions.
- The system's scalability is **limited to small and medium-sized businesses.**

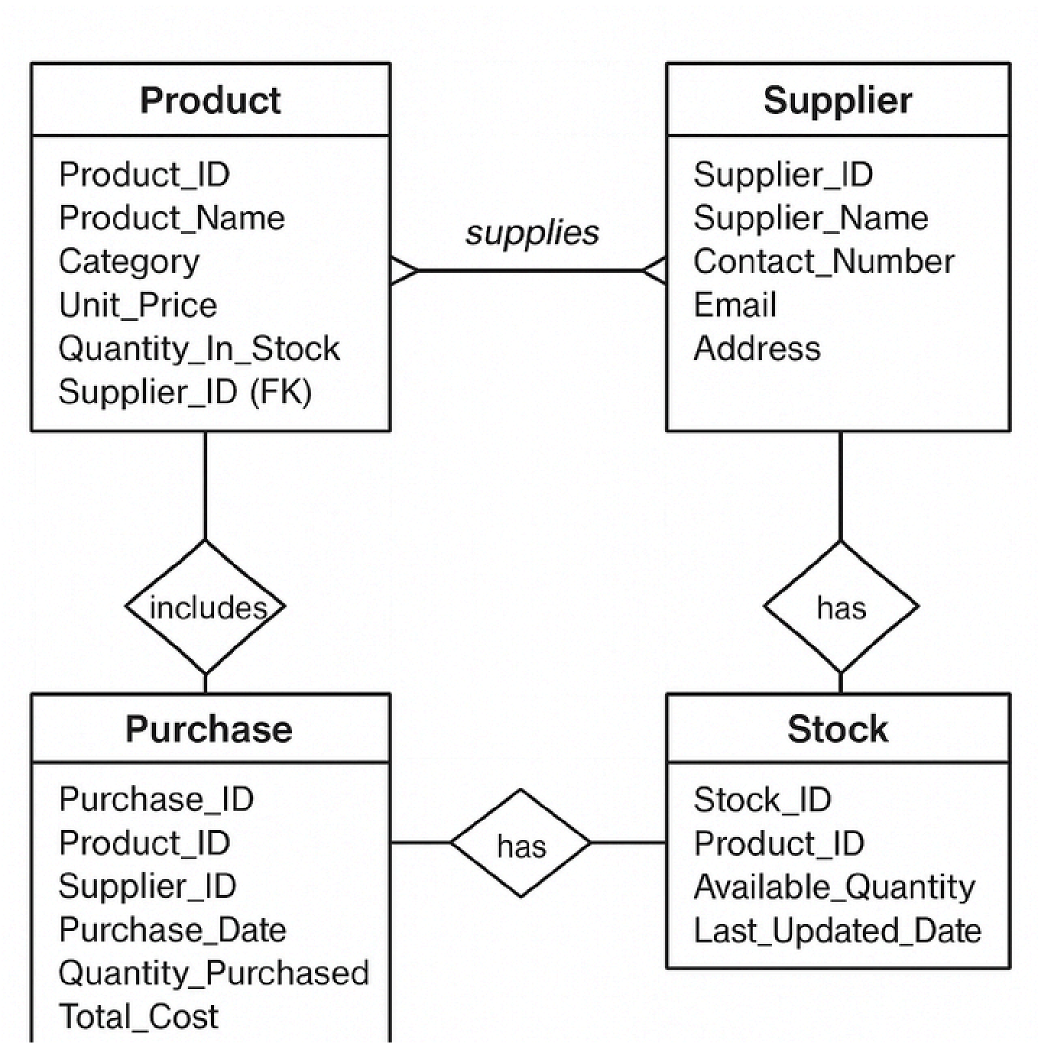
Algorithm:

1. Start
2. Connect to the Database using MySQL or the chosen DBMS.
3. Create Tables for the main entities:
    - Products
    - Suppliers
    - Stock
    - Purchases
4. Insert Data into each table (product details, supplier information, stock levels, etc.).
5. Establish Relationships using primary keys and foreign keys.
6. Perform Operations:
    - Add a new product or supplier.
    - Update product stock when a purchase is made.
    - Delete or modify existing records.
7. Execute Queries:
    - Retrieve product and stock details using SELECT and WHERE.
    - Generate reports using GROUP BY, HAVING, and JOIN.
    - Display supplier-wise product and purchase details.
8. View Results in tabular format.
9. End

Logic:

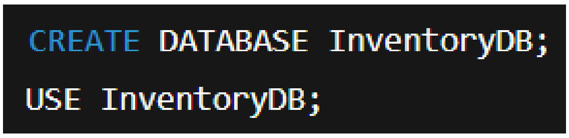
- Each product has a unique Product ID and is linked to a Supplier ID.
- When new stock is purchased, the quantity in the stock table is updated accordingly.
- SQL constraints like PRIMARY KEY, FOREIGN KEY, NOT NULL, and CHECK ensure data accuracy and referential integrity.
- Joins are used to combine data from multiple tables (e.g., products and suppliers).
- Views are created to simplify complex queries and display summarized information such as current stock status or supplier-wise inventory.

ER Diagram
(Entity -Relationship Diagram)



Code Explanation

1. Database Creation



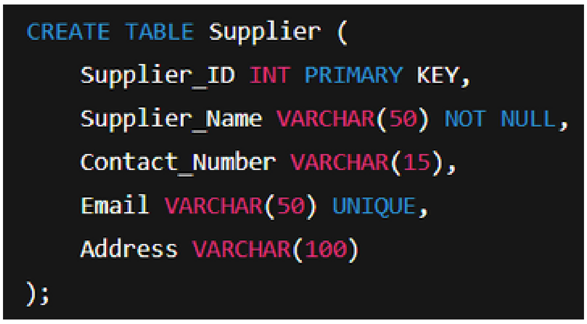
```
CREATE DATABASE InventoryDB;
USE InventoryDB;
```

This part creates a new database named **InventoryDB** and makes it the active database where all tables and data will be stored.

2. Creating Tables (DDL – Data Definition Language)

Each table represents a real-world entity from the inventory system.

Supplier Table



```
CREATE TABLE Supplier (
    Supplier_ID INT PRIMARY KEY,
    Supplier_Name VARCHAR(50) NOT NULL,
    Contact_Number VARCHAR(15),
    Email VARCHAR(50) UNIQUE,
    Address VARCHAR(100)
);
```

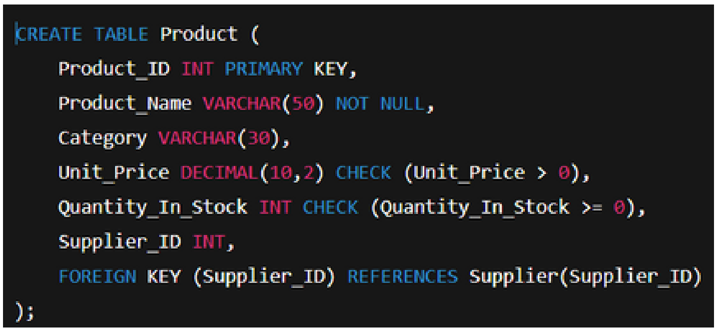
This table stores details about suppliers.

- **Supplier_ID** uniquely identifies each supplier.
- **Email** is marked **UNIQUE** so no two suppliers have the same email.
- **NOT NULL** ensures important fields (like names) aren’t left empty.

Product Table

This table stores product details.

- Each product has a unique **Product_ID.**
- **Supplier_ID** connects each product to its supplier (using Foreign Key).
- **CHECK** constraints make sure price and quantity values are valid (non-negative).

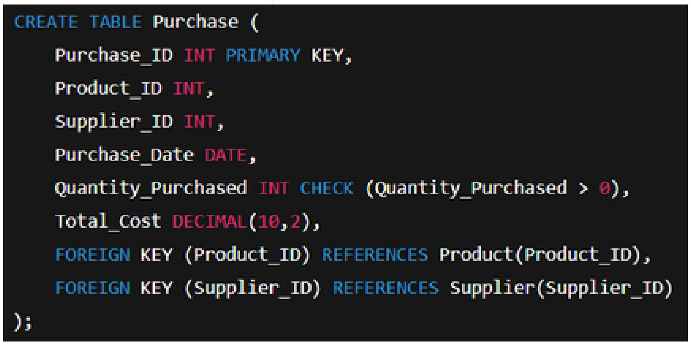


```
CREATE TABLE Product (
    Product_ID INT PRIMARY KEY,
    Product_Name VARCHAR(50) NOT NULL,
    Category VARCHAR(30),
    Unit_Price DECIMAL(10,2) CHECK (Unit_Price > 0),
    Quantity_In_Stock INT CHECK (Quantity_In_Stock >= 0),
    Supplier_ID INT,
    FOREIGN KEY (Supplier_ID) REFERENCES Supplier(Supplier_ID)
);
```

Purchase Table

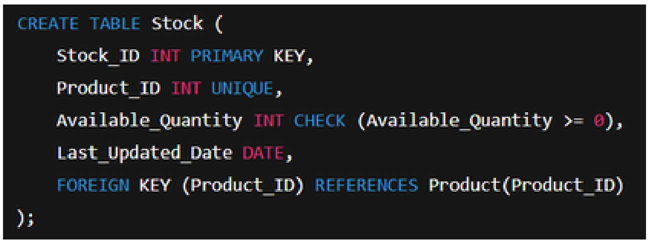
This table keeps **records of purchase transactions.**

- Links products and suppliers together.
- Stores how many items were purchased and the total cost.
- The foreign keys maintain referential integrity, ensuring only valid product and supplier IDs are used.



```
CREATE TABLE Purchase (
    Purchase_ID INT PRIMARY KEY,
    Product_ID INT,
    Supplier_ID INT,
    Purchase_Date DATE,
    Quantity_Purchased INT CHECK (Quantity_Purchased > 0),
    Total_Cost DECIMAL(10,2),
    FOREIGN KEY (Product_ID) REFERENCES Product(Product_ID),
    FOREIGN KEY (Supplier_ID) REFERENCES Supplier(Supplier_ID)
);
```

Stock Table

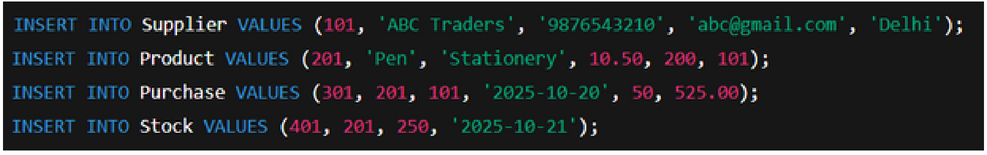


```
CREATE TABLE Stock (
    Stock_ID INT PRIMARY KEY,
    Product_ID INT UNIQUE,
    Available_Quantity INT CHECK (Available_Quantity >= 0),
    Last_Updated_Date DATE,
    FOREIGN KEY (Product_ID) REFERENCES Product(Product_ID)
);
```

This table keeps track of current **stock levels.**

- Each product has one corresponding stock entry **(Product_ID UNIQUE).**
- **Available_Quantity** updates whenever purchases are made.

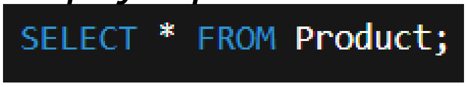
3. Inserting Data (DML Operations)



```
INSERT INTO Supplier VALUES (101, 'ABC Traders', '9876543210', 'abc@gmail.com', 'Delhi');
INSERT INTO Product VALUES (201, 'Pen', 'Stationery', 10.50, 200, 101);
INSERT INTO Purchase VALUES (301, 201, 101, '2025-10-20', 50, 525.00);
INSERT INTO Stock VALUES (401, 201, 250, '2025-10-21');
```

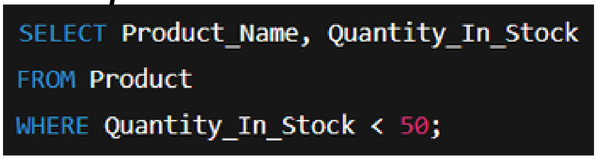
4. Sample Queries

a. Display all product details



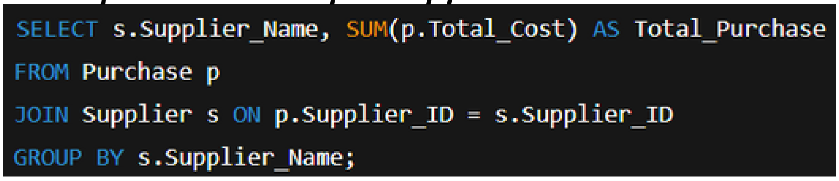
```
SELECT * FROM Product;
```

b. Show products with low stock



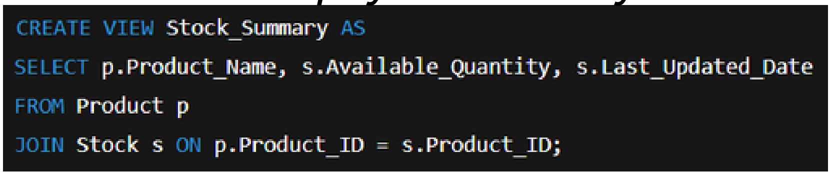
```
SELECT Product_Name, Quantity_In_Stock
FROM Product
WHERE Quantity_In_Stock < 50;
```

c. Total purchase cost per supplier



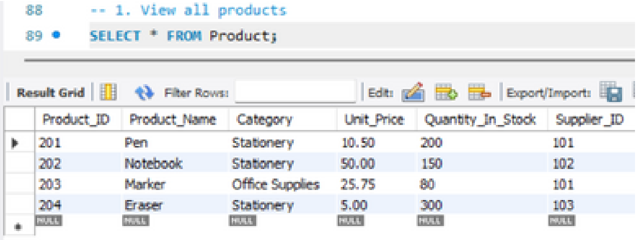
```
SELECT s.Supplier_Name, SUM(p.Total_Cost) AS Total_Purchase
FROM Purchase p
JOIN Supplier s ON p.Supplier_ID = s.Supplier_ID
GROUP BY s.Supplier_Name;
```

d. Create a view to display stock summary



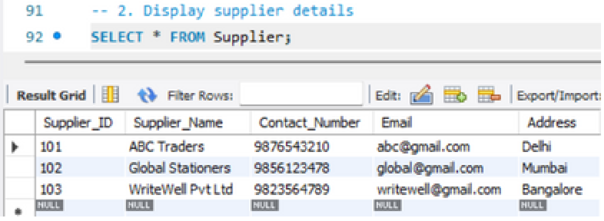
```
CREATE VIEW Stock_Summary AS
SELECT p.Product_Name, s.Available_Quantity, s.Last_Updated_Date
FROM Product p
JOIN Stock s ON p.Product_ID = s.Product_ID;
```

Screenshots/Output



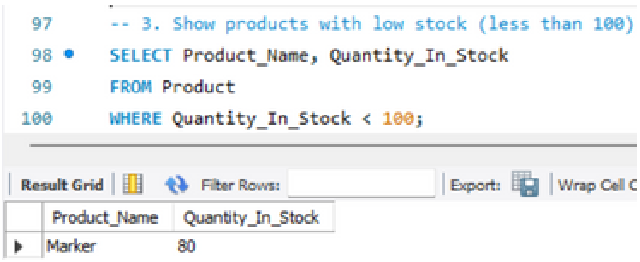
```
-- 1. View all products
SELECT * FROM Product;
```

Product_ID	Product_Name	Category	Unit_Price	Quantity_In_Stock	Supplier_ID
201	Pen	Stationery	10.50	200	101
202	Notebook	Stationery	50.00	150	102
203	Marker	Office Supplies	25.75	80	101
204	Eraser	Stationery	5.00	300	103
NULL	NULL	NULL	NULL	NULL	NULL



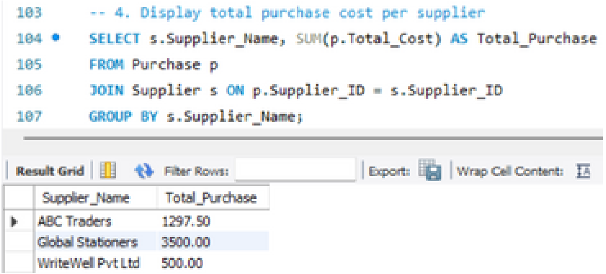
```
-- 2. Display supplier details
SELECT * FROM Supplier;
```

Supplier_ID	Supplier_Name	Contact_Number	Email	Address
101	ABC Traders	9876543210	abc@gmail.com	Delhi
102	Global Stationers	9856123478	global@gmail.com	Mumbai
103	WriteWell Pvt Ltd	9823564789	writewell@gmail.com	Bangalore
NULL	NULL	NULL	NULL	NULL



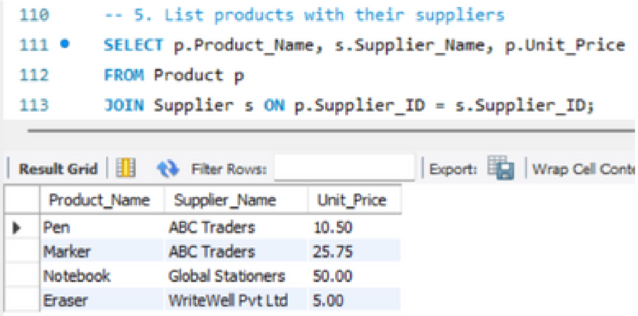
```
-- 3. Show products with low stock (less than 100)
SELECT Product_Name, Quantity_In_Stock
FROM Product
WHERE Quantity_In_Stock < 100;
```

Product_Name	Quantity_In_Stock
Marker	80



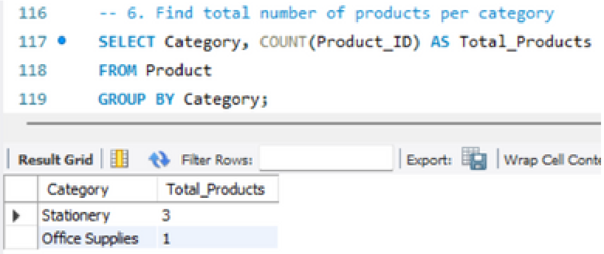
```
-- 4. Display total purchase cost per supplier
SELECT s.Supplier_Name, SUM(p.Total_Cost) AS Total_Purchase
FROM Purchase p
JOIN Supplier s ON p.Supplier_ID = s.Supplier_ID
GROUP BY s.Supplier_Name;
```

Supplier_Name	Total_Purchase
ABC Traders	1297.50
Global Stationers	3500.00
WriteWell Pvt Ltd	500.00



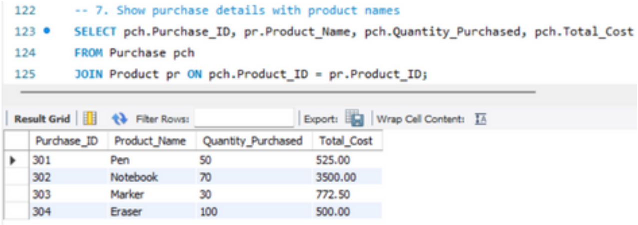
```
-- 5. List products with their suppliers
SELECT p.Product_Name, s.Supplier_Name, p.Unit_Price
FROM Product p
JOIN Supplier s ON p.Supplier_ID = s.Supplier_ID;
```

Product_Name	Supplier_Name	Unit_Price
Pen	ABC Traders	10.50
Marker	ABC Traders	25.75
Notebook	Global Stationers	50.00
Eraser	WriteWell Pvt Ltd	5.00



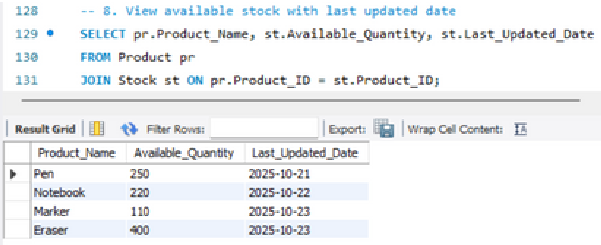
```
-- 6. Find total number of products per category
SELECT Category, COUNT(Product_ID) AS Total_Products
FROM Product
GROUP BY Category;
```

Category	Total_Products
Stationery	3
Office Supplies	1



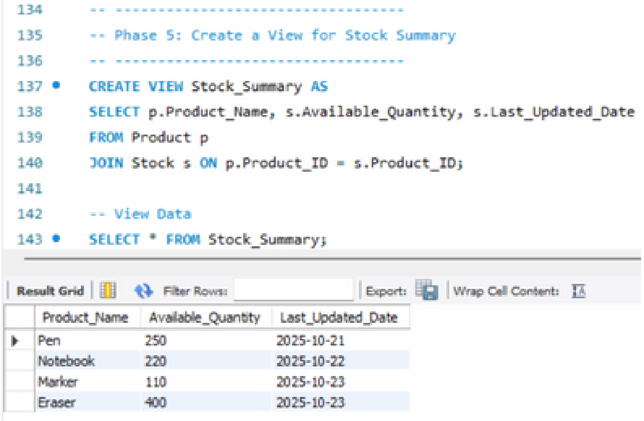
```
-- 7. Show purchase details with product names
SELECT pch.Purchase_ID, pr.Product_Name, pch.Quantity_Purchased, pch.Total_Cost
FROM Purchase pch
JOIN Product pr ON pch.Product_ID = pr.Product_ID;
```

Purchase_ID	Product_Name	Quantity_Purchased	Total_Cost
301	Pen	50	525.00
302	Notebook	70	3500.00
303	Marker	30	772.50
304	Eraser	100	500.00



```
-- 8. View available stock with last updated date
SELECT pr.Product_Name, st.Available_Quantity, st.Last_Updated_Date
FROM Product pr
JOIN Stock st ON pr.Product_ID = st.Product_ID;
```

Product_Name	Available_Quantity	Last_Updated_Date
Pen	250	2025-10-21
Notebook	220	2025-10-22
Marker	110	2025-10-23
Eraser	400	2025-10-23



```
-- ---------------------------------
-- Phase 5: Create a View for Stock Summary
-- ---------------------------------
CREATE VIEW Stock_Summary AS
SELECT p.Product_Name, s.Available_Quantity, s.Last_Updated_Date
FROM Product p
JOIN Stock s ON p.Product_ID = s.Product_ID;

-- View Data
SELECT * FROM Stock_Summary;
```

Product_Name	Available_Quantity	Last_Updated_Date
Pen	250	2025-10-21
Notebook	220	2025-10-22
Marker	110	2025-10-23
Eraser	400	2025-10-23

Conclusion

The Inventory Management System successfully manages products, suppliers, and stock details using database concepts. It provides an efficient and accurate way to handle inventory operations while maintaining data consistency and reducing redundancy. The project demonstrates how SQL and database design principles can be used to solve real-world business problems.

Learning Outcomes

- Understood the process of designing a database using E–R modeling.
- Learned normalization up to 3NF to avoid redundancy.
- Practiced SQL commands (DDL, DML, and queries).
- Implemented data constraints for integrity and consistency.
- Developed problem-solving and project presentation skills.

References

- Database System Concepts – Silberschatz, Korth, Sudarshan
- W3Schools SQL Tutorial
- GeeksforGeeks DBMS Notes
- MySQL Workbench / Oracle Database

Remarks