**COMPUTERSCIENCEDEPARTMENT**

## TotalMarks:

## ObtainedMarks:

DataBase

**Smart Inventory Management System**

**Submitted to: Ms. Kashia Riaz**

**Submitted by: Manahil Ejaz Raja**

**Aliza Sadaqat**

**Hifza Shafqat**

**Aleena Akhtar**

**Registration no:2212394, 22123415 , 2212390 , 2212378**

**Section: BSCS-4B**

**INDEX**

* [Introduction 03](#_bookmark1)

[Background 3](#_bookmark0)

[Problem Statement 3](#_bookmark1)

* [Objectives 04](#_TOC_250002)
* [Problem solving 05](#_bookmark2)
* [Functionalities 06](#_bookmark3)
* ER Diagram…………………………………………………………………………………..07
* Relational Model Mapping…………………………………………………………..08
* Schema………………………………………………………………………………………..09
* Relation Model…………………………………………………………………………….10
* Sql Queries with output………………..……………………………………………..11
* [Summary 16](#_bookmark3)
* [Conclusion 17](#_TOC_250001)
* [References 18](#_TOC_250000)

# Introduction

A Smart Inventory Management System is a sophisticated approach to overseeing stock levels, orders, and deliveries using advanced technologies like automation, data analytics, and real-time monitoring. It significantly improves the efficiency of tracking goods within businesses, reducing human errors, optimizing stock levels, and streamlining supply chain processes. In the context of a relational database, such systems rely on structured data management, ensuring the efficient storage, retrieval, and manipulation of inventory-related information. This project focuses on developing a robust, efficient inventory management solution using relational database models.

* **Background**

Inventory management has been a critical aspect of businesses for decades, ensuring that stock is available to meet demand without overstocking, which ties up resources. Traditional methods of managing inventory relied heavily on manual processes and simple tools like spreadsheets. However, with the advent of modern technology and data management systems, there has been a shift towards automated systems. Relational databases, which organize data into structured tables, have become a cornerstone of inventory management due to their ability to handle large datasets, support complex queries, and maintain data integrity. The need for more efficient, data-driven inventory systems has increased with the growth of businesses, globalization, and the advent of e-commerce, making smart inventory management systems indispensable.

* **Problem Statement**

In traditional inventory management systems, businesses face issues like overstocking, understocking, and manual errors, which lead to inefficiencies and losses. These systems often lack real-time tracking and data analysis capabilities, leading to delays in decision-making and missed opportunities for optimization. Additionally, with growing customer demands and the complexities of modern supply chains, it becomes challenging to maintain accurate and timely inventory records. The need for a smart, automated inventory management system that integrates real-time data processing, advanced analytics, and effective database management is crucial to solving these problems. This project aims to address these challenges by developing a relational database-driven solution for efficient and automated inventory management.

# 

# Objectives

Here are the objectives for a Smart Inventory Management System (SIMS) using a relational database, presented in point form:

1. Minimize discrepancies in stock counts and reduce errors in data entry.
2. Allow for continuous monitoring of inventory levels and stock movements.
3. Implement triggers for automatic reordering when stock reaches predetermined levels.
4. Provide comprehensive reports on inventory performance, trends, and forecasts.
5. Ensure compatibility with existing sales, finance, and supply chain management systems.
6. Establish role-based access to protect sensitive inventory data.
7. Track and analyze inventory costs to improve budgeting and reduce waste.
8. Maintain effective records of supplier performance and facilitate better procurement decisions.
9. Support various valuation methods for financial reporting and compliance.
10. Design the system to accommodate business growth and increased inventory complexity.
11. Improve order fulfillment accuracy and speed, leading to better customer experiences.
12. Provide data-driven insights to aid strategic planning and operational improvements.
13. Utilize historical data and trends to predict future inventory needs accurately.
14. Streamline inventory processes to minimize delays in receiving and fulfilling orders.
15. Provide stakeholders with clear insights into inventory status across multiple locations or channels.

# 

# Problem Solving

1. **Instant Stock Updates:**

Set up automatic updates when items are sold or received, so you always know what's in stock and avoid running out.

1. **No Duplicate Data:**

Organize data properly to avoid having the same information stored multiple times, making it easier to manage and save space.

1. **Predicting Future Needs:**

Use past sales data to help predict what items will be needed in the future, which helps maintain the right amount of stock.

1. **Supplier Tracking:**

Connect inventory items with their suppliers to monitor how well they perform, making it easier to choose the best ones.

1. **User Permissions:**

Set up permissions so only authorized staff can change inventory information, improving security and accountability.

1. **Easy Item Search:**

Create a simple search feature that allows users to quickly find items in the inventory, saving time and reducing frustration.

1. **Low Stock Alerts:**

Set up notifications to alert staff when items are running low, so they can reorder before running out.

1. **Sales Tracking:**

Keep a record of sales data to help identify popular products and trends, making it easier to plan future inventory purchases.

# Functionalities

Here’s a list of specific functionalities you might implement in a Smart Inventory Management System project, along with descriptions that focus on what you are doing:

* **Create the ER Diagram:**

Design an Entity-Relationship (ER) diagram to visually represent the database structure. Identify entities (e.g., Product, Customer, Orders), their attributes, and relationships to ensure clarity in data organization.

* **Create the Relational Schema:**

Develop a relational schema based on the ER diagram. Define tables, primary keys, and foreign keys reflecting the entities and relationships outlined in the ER diagram. This serves as the blueprint for the database.

* **Make Relational Tables:**

Create the actual relational tables in the database using SQL. Define each table structure, including primary keys, foreign keys, data types, and constraints (e.g., NOT NULL, UNIQUE).

* **Make a Full Database Using Different SQL Queries:**

Implement the full database using SQL queries to:

• Create tables (CREATE TABLE)

• Insert data into tables (INSERT INTO)

• Modify table structure if needed (ALTER TABLE)

• Retrieve data (SELECT)

• Use subqueries (WITH clauses)

• Update existing data (UPDATE)

• Delete records (DELETE)

• Ensure integrity with constraints and enforce relationships (FOREIGN KEY).

* **SQL Queries with Subqueries and Joins:**

Include example SQL queries that demonstrate practical database usage, incorporating joins to combine data from multiple tables and subqueries (WITH clauses or embedded queries) for advanced operations.

**Advanced Features Included**

1. Joins: Combine data from multiple tables for comprehensive reports.

2. Subqueries: Use nested queries for advanced filtering or computations.

3. Retrieval Queries: Use retrieval queries for the retrieval of data.

4. Conditional Queies: Incorporate conditional queries using like and etc .

# ER Diagram

****

# 

# Relational Model Mapping

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Table name** | **Attribute** | **Data Type** | **Key Type** | **Reference** |
| **Supplier** | **SupplierID** | **INT** | **PK** |  |
|  | **SupplierName** | **VARCHAR(100)** |  |  |
|  | **ContactInformation** | **VARCHAR(255)** |  |  |
|  | **Address** | **VARCHAR(255)** |  |  |
|  |  |  |  |  |
| **Product** | **ProductID** | **INT** | **PK** |  |
|  | **ProductName** | **VARCHAR(100)** |  |  |
|  | **Description** | **VARCHAR(255)** |  |  |
|  | **Category** | **VARCHAR(55)** |  |  |
|  | **Price** | **INT** |  |  |
|  | **QuantityInStock** | **INT** |  |  |
|  | **RecordLevel** | **INT** |  |  |
|  | **SupplierID** | **INT** |  |  |
|  |  |  |  |  |
| **Order** | **OrderId** | **INT** | **PK , AUTO\_INCREMENT** |  |
|  | **OrderDate** | **DATE** |  |  |
|  | **OrderStatus** | **VARCHAR(55)** |  |  |
|  |  |  |  |  |
| **OrderItem** | **OrderID** | **INT** | **PK,FK** | **OrderId(OrderId)** |
|  | **ProductId** | **INT** | **PK,FK** | **Product(ProductID)** |
|  | **QuantityOrdered** | **INT** |  |  |
|  |  |  |  |  |
| **InventryLog** | **LogId** | **INT** | **PK** |  |
|  | **ProductId** | **INT** | **FK** | **Product(ProductID)** |
|  | **Timestamp** | **TIMESTAMP** |  |  |
|  | **QuantityChange** | **INT** |  |  |
|  | **Reason** | **VARCHAR(255)** |  |  |
|  |  |  |  |  |
| **SensorData** | **SensorId** | **INT** | **PK** |  |
|  | **Timestamp** | **TIMESTAMP** |  |  |
|  | **Temperature** | **DECIMAL(5,2)** |  |  |
|  | **Humidity** | **DECIMAL(5,2)** |  |  |
|  | **ProductId** | **INT** | **FK** | **Product(ProductID)** |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

# Schema

* Categories (CategoryID, CategoryName)
* Suppliers (SupplierID, SupplierName, ContactEmail)
* Products (ProductID, ProductName, CategoryID (FK), SupplierID (FK), Price, Description)
* Inventory (InventoryID, ProductID (FK), Quantity, StockValue)
* Sales (SaleID, ProductID (FK), QuantitySold, SaleDate, SalePrice)
* Customers (CustomerID, CustomerName, Email)
* SalesOrders (OrderID, CustomerID (FK), OrderDate, TotalAmount, Status)
* OrderItems (OrderItemID, OrderID (FK), ProductID (FK), Quantity, Price)

# Relational Model

# 

# Sql Queries with output

CREATE DATABASE SmartInventoryManagementSystem;

USE SmartInventoryManagementSystem;

CREATE TABLE Categories (

CategoryID INT PRIMARY KEY AUTO\_INCREMENT,

CategoryName VARCHAR(100) NOT NULL

);

CREATE TABLE Suppliers (

SupplierID INT PRIMARY KEY AUTO\_INCREMENT,

SupplierName VARCHAR(100) NOT NULL,

ContactEmail VARCHAR(100)

);

CREATE TABLE Products (

ProductID INT PRIMARY KEY AUTO\_INCREMENT,

ProductName VARCHAR(100) NOT NULL,

CategoryID INT,

SupplierID INT,

Price DECIMAL(10, 2),

Description TEXT,

FOREIGN KEY (CategoryID) REFERENCES Categories(CategoryID),

FOREIGN KEY (SupplierID) REFERENCES Suppliers(SupplierID)

);

CREATE TABLE Inventory (

InventoryID INT PRIMARY KEY AUTO\_INCREMENT,

ProductID INT,

Quantity INT,

StockValue DECIMAL(10, 2),

FOREIGN KEY (ProductID) REFERENCES Products(ProductID)

);

CREATE TABLE Sales (

SaleID INT PRIMARY KEY AUTO\_INCREMENT,

ProductID INT,

QuantitySold INT,

SaleDate DATE,

SalePrice DECIMAL(10, 2),

FOREIGN KEY (ProductID) REFERENCES Products(ProductID)

);

CREATE TABLE Customers (

CustomerID INT PRIMARY KEY AUTO\_INCREMENT,

CustomerName VARCHAR(100),

Email VARCHAR(100)

);

CREATE TABLE SalesOrders (

OrderID INT PRIMARY KEY AUTO\_INCREMENT,

CustomerID INT,

OrderDate DATE,

TotalAmount DECIMAL(10, 2),

Status VARCHAR(50), -- e.g., 'Pending', 'Shipped', 'Delivered'

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

CREATE TABLE OrderItems (

OrderItemID INT PRIMARY KEY AUTO\_INCREMENT,

OrderID INT,

ProductID INT,

Quantity INT,

Price DECIMAL(10, 2),

FOREIGN KEY (OrderID) REFERENCES SalesOrders(OrderID),

FOREIGN KEY (ProductID) REFERENCES Products(ProductID)

);

INSERT INTO Categories (CategoryName) VALUES

('Electronics'),

('Furniture'),

('Clothing'),

('Books'),

('Food');

INSERT INTO Suppliers (SupplierName, ContactEmail) VALUES

('Pakistani Electronics', 'contact@pakelectronics.com'),

('Lahore Furniture', 'info@lahorefurniture.com'),

('Karachi Fashion Hub', 'sales@karachifashionhub.com'),

('Islamabad Book World', 'support@islamabadbookworld.com'),

('Fresh Foods Pakistan', 'orders@freshfoodspakistan.com');

INSERT INTO Products (ProductName, CategoryID, SupplierID, Price, Description) VALUES

('Laptop', 1, 1, 1200.00, 'High performance laptop for business and gaming'),

('Sofa', 2, 2, 500.00, 'Comfortable leather sofa'),

('T-Shirt', 3, 3, 20.00, 'Cotton T-shirt with trendy designs'),

('Novel', 4, 4, 15.00, 'Best-selling fiction novel'),

('Apple', 5, 5, 2.00, 'Fresh and organic apples');

INSERT INTO Inventory (ProductID, Quantity, StockValue) VALUES

(1, 50, 60000.00),

(2, 30, 15000.00),

(3, 100, 2000.00),

(4, 60, 900.00),

(5, 200, 400.00);

INSERT INTO Sales (ProductID, QuantitySold, SaleDate, SalePrice) VALUES

(1, 5, '2024-12-01', 1200.00),

(2, 3, '2024-12-02', 500.00),

(3, 10, '2024-12-03', 20.00),

(4, 2, '2024-12-04', 15.00),

(5, 50, '2024-12-05', 2.00);

INSERT INTO Customers (CustomerName, Email) VALUES

('Ahmed Ali', 'ahmed.ali@example.com'),

('Fatima Khan', 'fatima.khan@example.com');

INSERT INTO SalesOrders (CustomerID, OrderDate, TotalAmount, Status) VALUES

(1, '2024-12-10', 1500.00, 'Pending'),

(2, '2024-12-12', 200.00, 'Shipped');

INSERT INTO OrderItems (OrderID, ProductID, Quantity, Price) VALUES

(1, 1, 1, 1200.00),

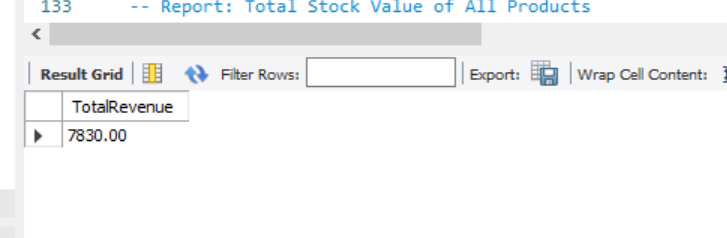
(1, 5, 10, 2.00),

(2, 3, 5, 20.00);

-- Total Revenue

SELECT SUM(Sales.QuantitySold \* Sales.SalePrice) AS TotalRevenue

FROM Sales;

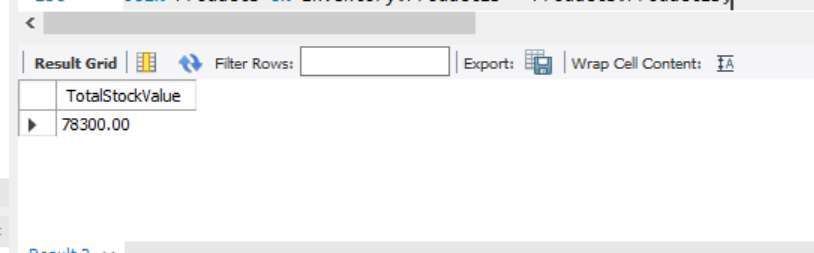


-- Total Stock Value

SELECT SUM(Inventory.Quantity \* Products.Price) AS TotalStockValue

FROM Inventory

JOIN Products ON Inventory.ProductID = Products.ProductID;



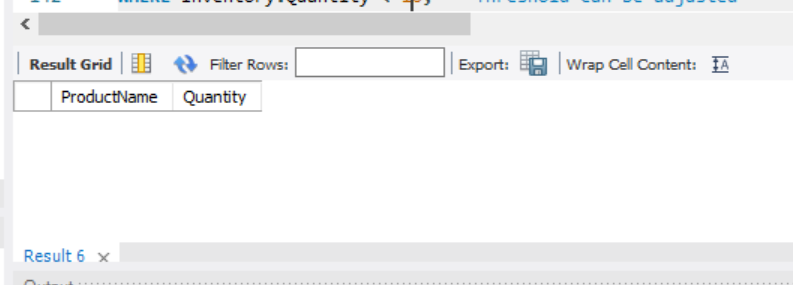
-- Products with Low Inventory

SELECT Products.ProductName, Inventory.Quantity

FROM Inventory

JOIN Products ON Inventory.ProductID = Products.ProductID

WHERE Inventory.Quantity < 10; -- Threshold can be adjusted



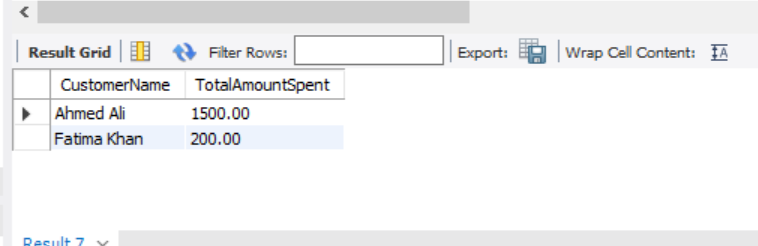
-- Total Amount of Orders by Customer

SELECT Customers.CustomerName, SUM(SalesOrders.TotalAmount) AS TotalAmountSpent

FROM SalesOrders

JOIN Customers ON SalesOrders.CustomerID = Customers.CustomerID

GROUP BY Customers.CustomerName;

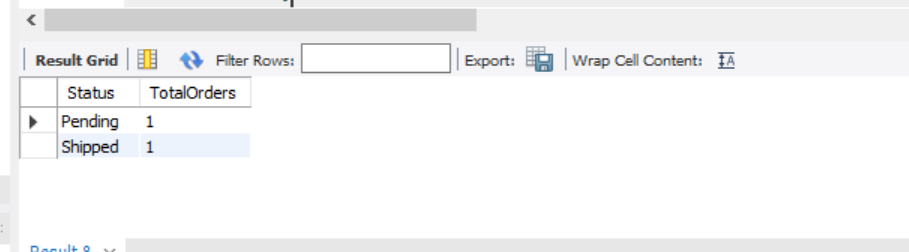


-- Orders by Status

SELECT Status, COUNT(\*) AS TotalOrders

FROM SalesOrders

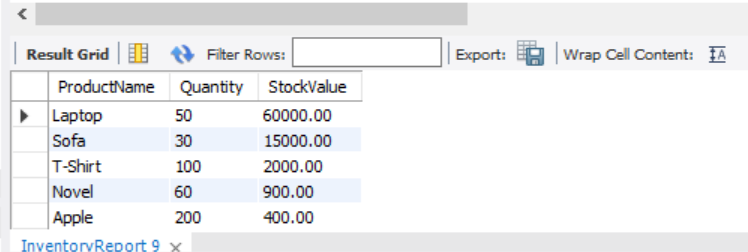
GROUP BY Status;



-- Adding a new product

INSERT INTO Products (ProductName, CategoryID, SupplierID, Price, Description)

VALUES ('Headphones', 1, 1, 100.00, 'Wireless headphones with noise-cancellation');



-- Query to view

SELECT \* FROM InventoryReport;

-- products with total sales greater than the average sales

SELECT ProductName

FROM Products

WHERE ProductID IN (

SELECT ProductID

FROM Sales

GROUP BY ProductID

HAVING SUM(QuantitySold) > (

SELECT AVG(TotalSales)

FROM (

SELECT SUM(QuantitySold) AS TotalSales

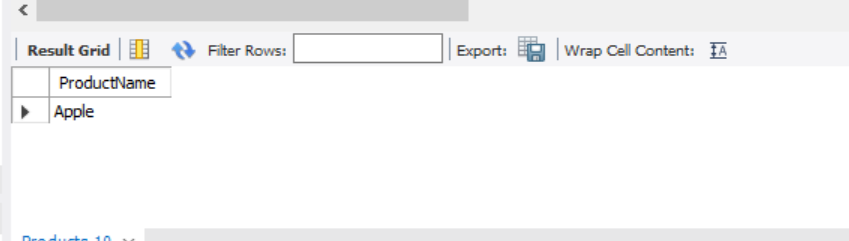
FROM Sales

GROUP BY ProductID

) AS AvgSales

)

);



-- category with the highest total sales

SELECT Categories.CategoryName,

SUM(Sales.QuantitySold \* Sales.SalePrice) AS TotalSales

FROM Sales

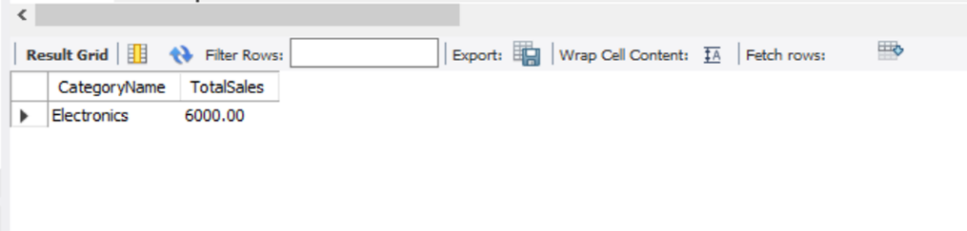
JOIN Products ON Sales.ProductID = Products.ProductID

JOIN Categories ON Products.CategoryID = Categories.CategoryID

GROUP BY Categories.CategoryName

ORDER BY TotalSales DESC

limit 1;



-- most profitable product

SELECT ProductName

FROM Products

WHERE ProductID = (

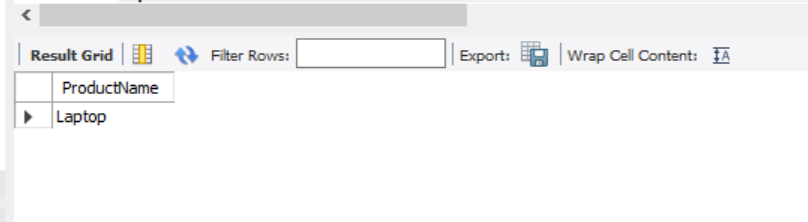
SELECT ProductID

FROM Sales

GROUP BY ProductID

ORDER BY SUM(QuantitySold \* SalePrice) DESC

LIMIT 1);



-- customer who bought the most expensive product

SELECT CustomerName

FROM Customers

WHERE CustomerID = (

SELECT CustomerID

FROM SalesOrders

WHERE OrderID IN (

SELECT OrderID

FROM OrderItems

WHERE ProductID = (

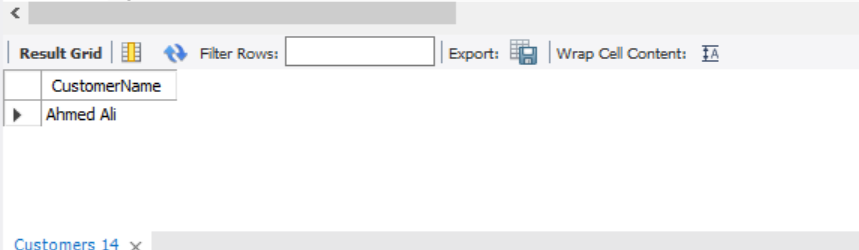
SELECT ProductID

FROM Products

ORDER BY Price DESC

LIMIT 1

)));

****

# Summary

A Smart Inventory Management System is a tool that helps businesses keep track of their products and supplies efficiently. It uses technology to monitor how much stock is available, when items need to be restocked, and where products are stored. A relational database is used in this system to store and organize all the information about the products, such as item names, quantities, suppliers, and prices.

In a relational database, data is stored in tables (like Excel spreadsheets) that are linked together. For example, one table may store information about products, another about suppliers, and a third about customers. These tables are connected so that the system can quickly access and update information, making inventory management faster and more accurate.

The goal of a smart system is to automate many of the manual tasks, like tracking inventory levels and sending alerts when stock is low, making the process more efficient and reducing human error.

# Conclusion

In conclusion, a smart inventory management system built on a relational database provides significant benefits to businesses by improving the efficiency of stock management. By automating key processes such as tracking inventory levels, organizing product data, and updating records in real-time, it reduces the chances of human error and enhances data accuracy. The system enables businesses to maintain an optimal stock balance, preventing both overstocking and stock shortages.

Additionally, with the ability to generate detailed reports, it supports better decision-making by providing insights into product demand, sales patterns, and inventory turnover. The relational database structure also allows for easy scalability, meaning the system can grow with the business as its needs expand. In summary, this smart inventory management solution not only streamlines daily operations but also boosts overall productivity, minimizes costs, and improves customer satisfaction by ensuring that products are always available when needed.

# 

# Refrences

1. <https://www.google.com/aclk?sa=l&ai=DChcSEwjSi-Gltp2JAxUmT0ECHZwwI6YYABAAGgJ3cw&co=1&gclid=CjwKCAjw1NK4BhAwEiwAVUHPUA7hxP6cgJ57P_kZY-U-GlNB4M2UTUkBT20v-NwGQuq1EHcZ72jsnhoCdC0QAvD_BwE&ei=1TUVZ-2-NcmgkdUPmLvs-AE&sig=AOD64_2kpuLCVAv0WMHLBxQNukwdZ223Rw&q&sqi=2&adurl&ved=2ahUKEwit_dqltp2JAxVJUKQEHZgdGx8Q0Qx6BAgJEAE>
2. <https://www.google.com/url?q=https://opus.govst.edu/cgi/viewcontent.cgi%3Farticle%3D1156%26context%3Dcapstones&sa=U&sqi=2&ved=2ahUKEwit_dqltp2JAxVJUKQEHZgdGx8QFnoECBUQAQ&usg=AOvVaw1ic0JDNr0sNLzDso1o7n-Z>
3. <https://www.google.com/url?q=https://www.slideshare.net/slideshow/inventory-management-system-132114765/132114765&sa=U&sqi=2&ved=2ahUKEwi6gIHGtp2JAxV2T6QEHXt6A_8QFnoECC0QAQ&usg=AOvVaw1DoFyl0WkrVuz6SVUj1AXL>
4. <https://www.google.com/url?q=https://www.geeksforgeeks.org/how-to-design-database-inventory-management-systems/&sa=U&sqi=2&ved=2ahUKEwi6gIHGtp2JAxV2T6QEHXt6A_8QFnoECCwQAQ&usg=AOvVaw1e6yn_fBkPE3LxqU-1hqyE>
5. <https://www.google.com/aclk?sa=l&ai=DChcSEwiP9pfotp2JAxUmPgYAHTuuHcQYABABGgJ3cw&co=1&gclid=CjwKCAjw1NK4BhAwEiwAVUHPUM7R3fyCvp91rWKZmcLRywxtqgv0GJz6Nk8C5dbwToTFhzp5HFTJERoCZZAQAvD_BwE&ei=YTYVZ7asC4GekdUP3o--iQ8&sig=AOD64_1nrioaZNGuNMaKQm8rtdjTOOG1-g&q&sqi=2&adurl&ved=2ahUKEwj24JHotp2JAxUBT6QEHd6HL_EQ0Qx6BAgHEAE>
6. <https://www.google.com/url?q=https://www.geeksforgeeks.org/how-to-design-database-inventory-management-systems/&sa=U&sqi=2&ved=2ahUKEwj24JHotp2JAxUBT6QEHd6HL_EQFnoECBUQAQ&usg=AOvVaw1njq9zsjj7L7TUF4WMNoqm>