
INVENTORY
MANAGEMENT

Title: Inventory Management System

Introduction: Inventory management is an important part of any business. It helps in tracking products, stock levels, sales, and purchases. Without proper inventory control, businesses may face product shortages or wastage. This project uses SQL to design an Inventory Management System that helps in managing inventory data efficiently.

Problem Statement: Retail businesses often struggle with maintaining proper stock levels. Overstocking leads to wastage and blocked capital, while understocking leads to loss of sales. There is a need for a database system that can track inventory, sales, and purchases accurately.

Objectives of the Project:

- To design a database for inventory management
- To store product, stock, sales, and supplier data
- To analyze inventory using SQL queries
- To support simple business decision making

Tables Used:

- Products: Stores basic details of all products sold by the business.
- Stock: Stores the current available quantity of each product.
- Sales: Stores records of products sold to customers.
- Purchases: Stores records of products bought from suppliers.
- Suppliers: Stores details of suppliers who provide products to the business.

Queries: SQL IMPLEMENTATION

 # Database Creation

Input:

```
create database Inventory_Management;
use Inventory_Management;
```

Output:

 # Create Tables and Insert Values:

Table: Products

Input:

```
-- Products Table
CREATE TABLE Products (
    product_id INT PRIMARY KEY,
    product_name VARCHAR(50),
    category VARCHAR(30),
    price DECIMAL(10,2),
    reorder_level INT);
```

Output:

	Field	Type	Null	Key	Default	Extra
▶	product_id	int	NO	PRI	NULL	
	product_name	varchar(50)	YES		NULL	
	category	varchar(30)	YES		NULL	
	price	decimal(10,2)	YES		NULL	
	reorder_level	int	YES		NULL	

Input:

```
INSERT INTO Products(product_id, product_name, category, price, reorder_level)
VALUES
(1,'Laptop','Electronics',55000,5),
(2,'Mouse','Electronics',500,20),
(3,'Keyboard','Electronics',1200,15),
(4,'Monitor','Electronics',15000,8),
(5,'Printer','Electronics',9000,6),
(6,'Chair','Furniture',2500,10),
(7,'Table','Furniture',6000,5),
(8,'Cupboard','Furniture',12000,4),
(9,'Notebook','Stationery',50,50),
(10,'Pen','Stationery',10,100),
(11,'Marker','Stationery',30,60),
(12,'Stapler','Stationery',150,20),
(13,'Headphones','Electronics',2000,12),
(14,'Webcam','Electronics',3500,7),
(15,'Router','Electronics',4000,6),
(16,'Water Bottle','Accessories',300,25),
(17,'Backpack','Accessories',1800,10),
(18,'Calculator','Stationery',700,15),
(19,'Extension Board','Electronics',800,10),
(20,'Paper Ream','Stationery',350,30);
```

Output:

product_id	product_name	category	price	reorder_level
1	Laptop	Electronics	55000.00	5
2	Mouse	Electronics	500.00	20
3	Keyboard	Electronics	1200.00	15
4	Monitor	Electronics	15000.00	8
5	Printer	Electronics	9000.00	6
6	Chair	Furniture	2500.00	10
7	Table	Furniture	6000.00	5
8	Cupboard	Furniture	12000.00	4
9	Notebook	Stationery	50.00	50
10	Pen	Stationery	10.00	100
11	Marker	Stationery	30.00	60
12	Stapler	Stationery	150.00	20
13	Headphones	Electronics	2000.00	12
14	Webcam	Electronics	3500.00	7
15	Router	Electronics	4000.00	6
16	Water Bottle	Accessories	300.00	25
17	Backpack	Accessories	1800.00	10
18	Calculator	Stationery	700.00	15
19	Extension Board	Electronics	800.00	10
20	Paper Ream	Stationery	350.00	30
NULL	NULL	NULL	NULL	NULL

Table: Suppliers

Input:

```
-- Suppliers Table
CREATE TABLE Suppliers (
    supplier_id INT PRIMARY KEY,
    supplier_name VARCHAR(50),
    contact_no VARCHAR(15));


-- Add City to Suppliers
ALTER TABLE Suppliers ADD city VARCHAR(50);
```

Output:

Field	Type	Null	Key	Default	Extra
supplier_id	int	NO	PRI	NULL	
supplier_name	varchar(50)	YES		NULL	
contact_no	varchar(15)	YES		NULL	
city	varchar(50)	YES		NULL	

Input:

```
INSERT INTO Suppliers VALUES
(1,'Tech Distributors','9876543210','Mumbai'),
(2,'Office Mart','9123456780','Pune'),
(3,'Electro World','9988776655','Delhi'),
(4,'Smart Supplies','9012345678','Bangalore'),
(5,'Furniture Hub','9090909090','Chennai'),
(6,'Stationery Zone','8888777766','Hyderabad'),
(7,'Digital Store','7777666655','Ahmedabad'),
(8,'Gadget Point','9999888877','Kolkata'),
(9,'Supply Chain Co','9666554433','Jaipur'),
(10,'Warehouse India','9555443322','Indore'),
(11,'Retail Source','9444332211','Bhopal'),
(12,'Prime Distributors','9333221100','Nagpur'),
(13,'Office Needs','9222110099','Noida'),
(14,'Techno Solutions','9111009988','Gurgaon'),
(15,'Metro Traders','9000998877','Thane'),
(16,'City Wholesalers','9888776654','Navi Mumbai'),
(17,'Bulk Buy Ltd','9777665544','Surat'),
(18,'Fast Supply','9666554432','Vadodara'),
(19,'Allied Traders','9555443321','Rajkot'),
(20,'Inventory Experts','9444332209','Udaipur');
```

Output:

supplier_id	supplier_name	contact_no	city
1	Tech Distributors	9876543210	Mumbai
2	Office Mart	9123456780	Pune
3	Electro World	9988776655	Delhi
4	Smart Supplies	9012345678	Bangalore
5	Furniture Hub	9090909090	Chennai
6	Stationery Zone	8888777766	Hyderabad
7	Digital Store	7777666655	Ahmedabad
8	Gadget Point	9999888877	Kolkata
9	Supply Chain Co	9666554433	Jaipur
10	Warehouse India	9555443322	Indore
11	Retail Source	9444332211	Bhopal
12	Prime Distributors	9333221100	Nagpur
13	Office Needs	9222110099	Noida
14	Techno Solutions	9111009988	Gurgaon
15	Metro Traders	9000998877	Thane
16	City Wholesalers	9888776654	Navi Mumbai
17	Bulk Buy Ltd	9777665544	Surat
18	Fast Supply	9666554432	Vadodara
19	Allied Traders	9555443321	Rajkot
20	Inventory Experts	9444332209	Udaipur
NULL	NULL	NULL	NULL

Table: Stock

Input:

```
-- Stock Table
CREATE TABLE Stock (
    product_id INT,
    quantity_available INT,
    last_updated DATE,
    FOREIGN KEY (product_id) REFERENCES Products(product_id)
);
```

Output:

Field	Type	Null	Key	Default	Extra
product_id	int	YES	MUL	NULL	
quantity_available	int	YES		NULL	
last_updated	date	YES		NULL	

Input:

```
INSERT INTO Stock (product_id, quantity_available, last_updated)
VALUES
(1,4,'2025-03-01'),
(2,25,'2025-03-01'),
(3,10,'2025-03-01'),
(4,6,'2025-03-01'),
(5,3,'2025-03-01'),
(6,8,'2025-03-01'),
(7,2,'2025-03-01'),
(8,5,'2025-03-01'),
(9,120,'2025-03-01'),
(10,300,'2025-03-01'),
(11,50,'2025-03-01'),
(12,18,'2025-03-01'),
(13,9,'2025-03-01'),
(14,6,'2025-03-01'),
(15,4,'2025-03-01'),
(16,30,'2025-03-01'),
(17,12,'2025-03-01'),
(18,10,'2025-03-01'),
(19,7,'2025-03-01'),
(20,40,'2025-03-01');
```

Output:

product_id	quantity_available	last_updated
1	4	2025-03-01
2	25	2025-03-01
3	10	2025-03-01
4	6	2025-03-01
5	3	2025-03-01
6	8	2025-03-01
7	2	2025-03-01
8	5	2025-03-01
9	120	2025-03-01
10	300	2025-03-01
11	50	2025-03-01
12	18	2025-03-01
13	9	2025-03-01
14	6	2025-03-01
15	4	2025-03-01
16	30	2025-03-01
17	12	2025-03-01
18	10	2025-03-01
19	7	2025-03-01
20	40	2025-03-01

Table: Sale

Input:

```
-- Sales Table
CREATE TABLE Sales (
    sale_id INT PRIMARY KEY,
    product_id INT,
    sale_date DATE,
    quantity_sold INT,
    FOREIGN KEY (product_id) REFERENCES Products(product_id)
);
```

Output:

Field	Type	Null	Key	Default	Extra
sale_id	int	NO	PRI	NULL	
product_id	int	YES	MUL	NULL	
sale_date	date	YES		NULL	
quantity_sold	int	YES		NULL	

Input:

```
INSERT INTO Sales VALUES
(201,1,'2025-03-01',6),
(202,2,'2025-03-01',30),
(203,3,'2025-03-01',20),
(204,4,'2025-03-01',9),
(205,5,'2025-03-01',7),
(206,6,'2025-03-02',12),
(207,7,'2025-03-02',8),
(208,8,'2025-03-02',3),
(209,9,'2025-03-02',80),
(210,10,'2025-03-02',200),
(211,11,'2025-03-03',45),
(212,12,'2025-03-03',22),
(213,13,'2025-03-03',16),
(214,14,'2025-03-03',14),
(215,15,'2025-03-03',12),
(216,16,'2025-03-04',25),
(217,17,'2025-03-04',18),
(218,18,'2025-03-04',20),
(219,19,'2025-03-04',28),
(220,20,'2025-03-04',35);
```

Output:

sale_id	product_id	sale_date	quantity_sold
201	1	2025-03-01	6
202	2	2025-03-01	30
203	3	2025-03-01	20
204	4	2025-03-01	9
205	5	2025-03-01	7
206	6	2025-03-02	12
207	7	2025-03-02	8
208	8	2025-03-02	3
209	9	2025-03-02	80
210	10	2025-03-02	200
211	11	2025-03-03	45
212	12	2025-03-03	22
213	13	2025-03-03	16
214	14	2025-03-03	14
215	15	2025-03-03	12
216	16	2025-03-04	25
217	17	2025-03-04	18
218	18	2025-03-04	20
219	19	2025-03-04	28
220	20	2025-03-04	35
NULL	NULL	NULL	NULL

Table: Purchases

Input:

```
-- Purchases Table
CREATE TABLE Purchases (
    purchase_id INT PRIMARY KEY,
    product_id INT,
    purchase_date DATE,
    quantity_purchased INT,
    FOREIGN KEY (product_id) REFERENCES Products(product_id)
);
```

Input:

```
-- Add supplier_id to Purchases
ALTER TABLE Purchases
ADD supplier_id INT,
ADD FOREIGN KEY (supplier_id) REFERENCES Suppliers(supplier_id);
```

Output:

Field	Type	Null	Key	Default	Extra
purchase_id	int	NO	PRI	NULL	
product_id	int	YES	MUL	NULL	
purchase_date	date	YES		NULL	
quantity_purchased	int	YES		NULL	
supplier_id	int	YES	MUL	NULL	

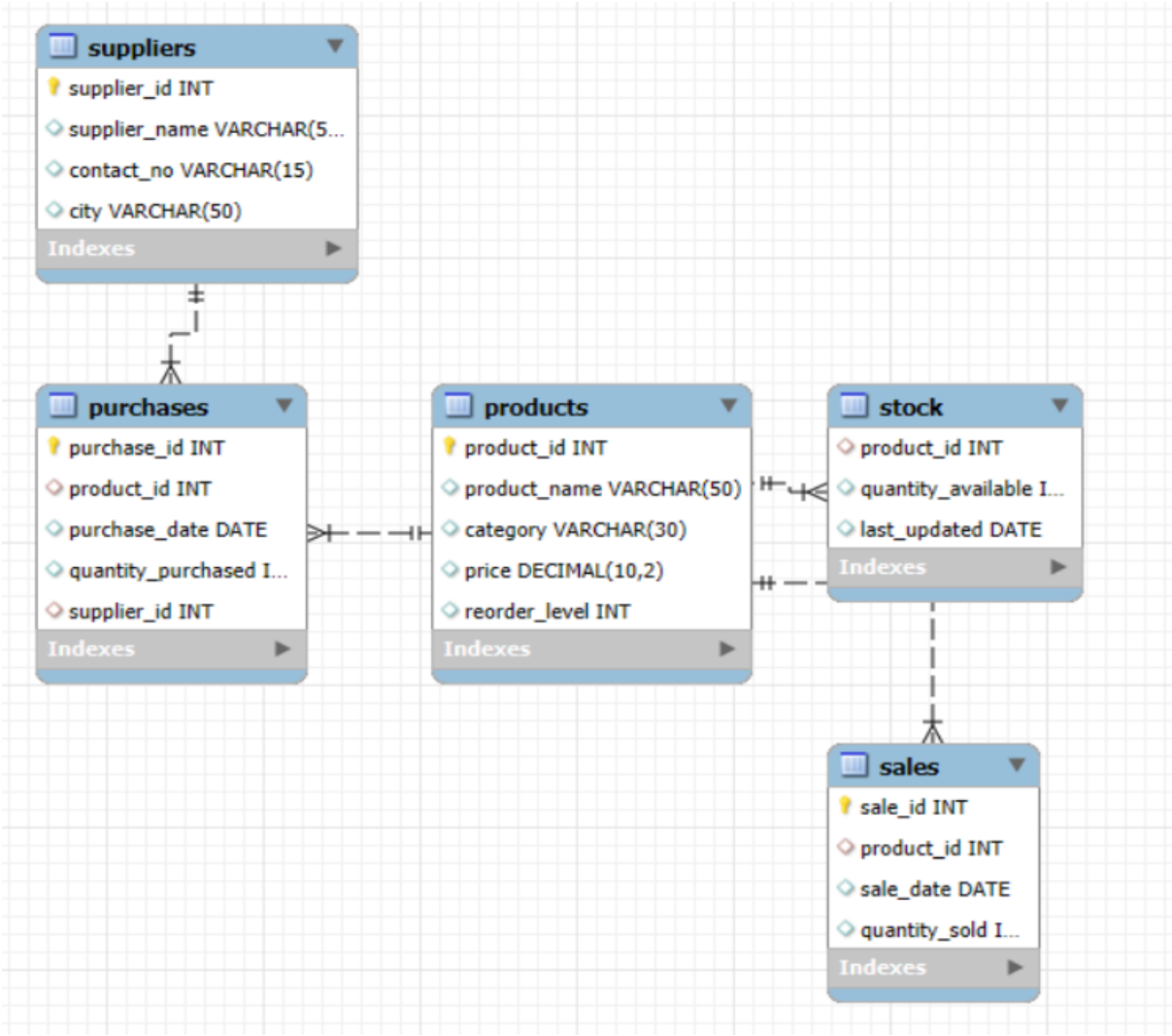
Input:

```
-- Insert into Purchases Table
INSERT INTO Purchases
(purchase_id, product_id, purchase_date, quantity_purchased, supplier_id)
VALUES
(101,1,'2025-02-01',10,1),
(102,2,'2025-02-02',50,1),
(103,3,'2025-02-03',30,1),
(104,4,'2025-02-04',15,1),
(105,5,'2025-02-05',10,1),
(106,6,'2025-02-06',20,2),
(107,7,'2025-02-07',10,2),
(108,8,'2025-02-08',8,2),
(109,9,'2025-02-09',200,2),
(110,10,'2025-02-10',500,2),
(111,11,'2025-02-11',100,3),
(112,12,'2025-02-12',50,3),
(113,13,'2025-02-13',25,3),
(114,14,'2025-02-14',20,3),
(115,15,'2025-02-15',18,3),
(116,16,'2025-02-16',60,4),
(117,17,'2025-02-17',30,4),
(118,18,'2025-02-18',40,4),
(119,19,'2025-02-19',35,4),
(120,20,'2025-02-20',80,4);
```

Output:

purchase_id	product_id	purchase_date	quantity_purchased	supplier_id
101	1	2025-02-01	10	1
102	2	2025-02-02	50	1
103	3	2025-02-03	30	1
104	4	2025-02-04	15	1
105	5	2025-02-05	10	1
106	6	2025-02-06	20	2
107	7	2025-02-07	10	2
108	8	2025-02-08	8	2
109	9	2025-02-09	200	2
110	10	2025-02-10	500	2
111	11	2025-02-11	100	3
112	12	2025-02-12	50	3
113	13	2025-02-13	25	3
114	14	2025-02-14	20	3
115	15	2025-02-15	18	3
116	16	2025-02-16	60	4
117	17	2025-02-17	30	4
118	18	2025-02-18	40	4
119	19	2025-02-19	35	4
120	20	2025-02-20	80	4
NULL	NULL	NULL	NULL	NULL

ENTITY RELATIONSHIP DIAGRAM:



1. Suppliers → Purchases (One-to-Many)

Reason:

- A supplier can supply many orders over time.
- Each purchase order comes from only one supplier.

2. Products → Purchases (One-to-Many)

Reason:

- The same product can be purchased multiple times.
- Each purchase record refers to one product.

3. Products → Stock (One-to-One)

Reason:

- You maintain one stock row per product.
- Stock table shows current quantity, not warehouse-wise stock.

4**. Products → Sales (One-to-Many)**

Reason:

- A product can be sold many times.
- Each sale record is linked to one product.

5. Stock → Sales (One-to-Many)

Reason:

- Stock quantity decreases every time a sale occurs.
- Sales depend on stock availability.

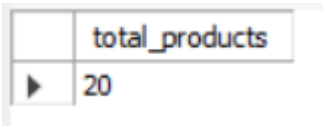
#Top 10 Analysis

1.Total Number of Products

Input:

```
Select COUNT(*) AS total_products from Products;
```

Output:



	total_products
▶	20

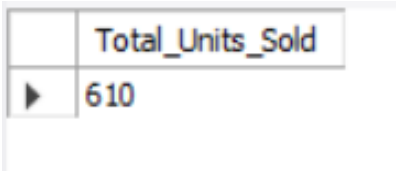
- **Function Used: COUNT ()**
- **Business Insight: Shows how many different products the business manages.**
- **Business Action: Helps understand inventory size.**

2.Total Units Sold

Input:

```
Select SUM(quantity_sold) as Total_Units_Sold from Sales;
```

Output:



	Total_Units_Sold
▶	610

- **Function Used: SUM()**
- **Business Insight: Shows overall sales volume.**
- **Business Action: Helps measure business performance.**

3.Current Inventory Value

Input:

```
Select ROUND(SUM(p.price * s.quantity_available),2) AS inventory_value
From Products p
Inner Join  Stock s
on p.product_id = s.product_id;
```

Output:

	inventory_value
▶	578900.00

- **Functions Used: SUM(), Arithmetic operation (*), ROUND(), Inner Join**
- **Business Insight: Shows how much money is tied up in inventory.**
- **Business Action: Helps control overstocking and cash flow.**

4.Category-Wise Sales

Input:

```
Select p.category,SUM(s.quantity_sold) AS total_units_sold
From Sales s
Inner Join Products p on s.product_id = p.product_id
Group by p.category;
```

Output:

	category	total_units_sold
	Electronics	142
	Furniture	23
	Stationery	402
	Accessories	43

- **Functions / Clauses Used: SUM(), INNER JOIN, GROUP BY, AS**
- **Business Insight:**
  Identifies which product category sells the most.
- **Business Action:**
  Focus marketing and stock on strong categories.

5.Low-Stock Products (Reorder Alert)

Input:

```
Select p.product_name, s.quantity_available, p.reorder_level
From Products p
Inner Join  Stock s on p.product_id = s.product_id
where s.quantity_available < p.reorder_level;
```

Output:

product_name	quantity_available	reorder_level
Laptop	4	5
Keyboard	10	15
Monitor	6	8
Printer	3	6
Chair	8	10
Table	2	5
Marker	50	60
Stapler	18	20
Headphones	9	12
Webcam	6	7
Router	4	6
Calculator	10	15
Extension Board	7	10

- **Functions / Clauses Used: INNER JOIN, WHERE, AS**
- **Business Insight:** Identifies products that may go out of stock.
- **Business Action:** Reorder products on time.

6.Top 5 Best-Selling Products

Input:

```
Select p.product_name,SUM(s.quantity_sold) AS total_sold
From Sales s
Join Products p
on s.product_id = p.product_id
group by p.product_name
order by total_sold DESC
limit 5;
```

Output:

product_name	total_sold
Pen	200
Notebook	80
Marker	45
Paper Ream	35
Mouse	30

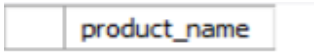
- **Functions / Clauses Used: SUM()**, **INNER JOIN**, **GROUP BY**, **ORDER BY, LIMIT** , **AS (Alias)**
- **Business Insight:** Shows most popular products.
- **Business Action:** Keep higher stock for these products.

7.Dead Stock (Products with No Sales)

Input:

```
Select p.product_name
from Products p
left join Sales sa on p.product_id = sa.product_id
where sa.product_id is null;
```

Output:



- **Functions / Clauses Used: LEFT JOIN, IS NULL, SELECT , ON**
- **Business Insight:** Identifies products with zero sales.
- **Business Action:** Discount or stop stocking these items.

8.Supplier-Wise Purchase Analysis

Input:

```
SELECT s.supplier_name, SUM(pu.quantity_purchased) as total_purchased
from Purchases pu
Inner Join Suppliers s
on pu.supplier_id = s.supplier_id
group by s.supplier_name
order by total_purchased DESC;
```

Output:

supplier_name	total_purchased
Office Mart	738
Smart Supplies	245
Electro World	213
Tech Distributors	115

- **Functions / Clauses Used: SUM(), INNER JOIN, GROUP BY, ORDER BY (DESC), AS (Alias)**
- **Business Insight:**
  Shows which supplier supplies the most products.
- **Business Action:**
  Build long-term partnerships with reliable suppliers.

9.Stock Risk Analysis (Near Reorder Level)

Input:

```
select p.product_name, s.quantity_available
FROM Products p
inner join Stock s
on p.product_id = s.product_id
where s.quantity_available <= p.reorder_level + 2;
```

Output:

product_name	quantity_available
Laptop	4
Keyboard	10
Monitor	6
Printer	3
Chair	8
Table	2
Cupboard	5
Marker	50
Stapler	18
Headphones	9
Webcam	6
Router	4
Backpack	12
Calculator	10
Extension Board	7

- **Functions / Clauses Used: INNER JOIN , WHERE ,Arithmetic Operation (+) , SELECT**
- **Business Insight:** Identifies products close to shortage.
- **Business Action:** Take preventive restocking action.

10.Inventory Turnover

Input:

```
select p.product_name, SUM(sa.quantity_sold) as total_sold
from Products p
left join Sales sa on p.product_id = sa.product_id
group by p.product_name;
```

Output:

product_name	total_sold
Mouse	30
Keyboard	20
Monitor	9
Printer	7
Chair	12
Table	8
Cupboard	3
Notebook	80
Pen	200
Marker	45
Stapler	22
Headphones	16
Webcam	14
Router	12
Water Bottle	25
Backpack	18
Calculator	20
Extension Board	28
Paper Ream	35

Functions / Clauses Used

- **SUM() , LEFT JOIN , GROUP BY , AS (Alias) ,SELECT**
  - **Business Insight:** Shows how fast each product is sold.
  - **Business Action:** Increase stock for fast-moving products.

Conclusion:

This project uses SQL to analyze inventory data and support basic business decisions.
It helps monitor stock levels, identify fast- and slow-moving products, and evaluate supplier performance.
Overall, the system improves inventory control and supports efficient business operations.