Project For SQL MODULE

Smart Inventory Management System for Perishable Goods

Institute Name:- Itvedant Education Pvt .Ltd

Name:- OM SANJAY SURYAWANSHI

Email Address:- suryawanshis107@gmail.com

Date of submission:-

Project Aims:

- ## Optimize Inventory Levels:

  Develop a system to maintain optimal inventory levels, reducing waste due to overstocking and preventing shortages that could lead to lost sales.

- ## Real-Time Monitoring:

  Implement real-time tracking of inventory data to enable timely decision-making regarding restocking or discounting near-expiry items.

- ## Shelf-Life Management:

  Create a system to monitor the shelf life of perishable goods, ensuring that items are sold or used before they expire.

- ## Demand Forecasting:

  Use historical sales data to predict future demand for perishable items, enabling better inventory planning and reducing the risk of overstocking.

- ## Automated Reordering:

  Establish an automated reordering system based on inventory levels and forecasted demand, ensuring that stock is replenished as needed.

- ## Supplier Coordination:

  Develop mechanisms to coordinate with suppliers for just-in-time delivery, minimizing the storage time of perishable goods and ensuring freshness.

- ## Data-Driven Insights:

  Utilize SQL queries to extract insights from inventory data, helping to identify trends, inefficiencies, and areas for cost savings.

- ## Minimize Waste:

  Implement strategies to minimize food waste by optimizing inventory turnover and employing first-in, first-out (FIFO) principles.

- ## Cost Efficiency:

  Aim to reduce costs associated with inventory management by improving accuracy in inventory tracking and reducing spoilage.

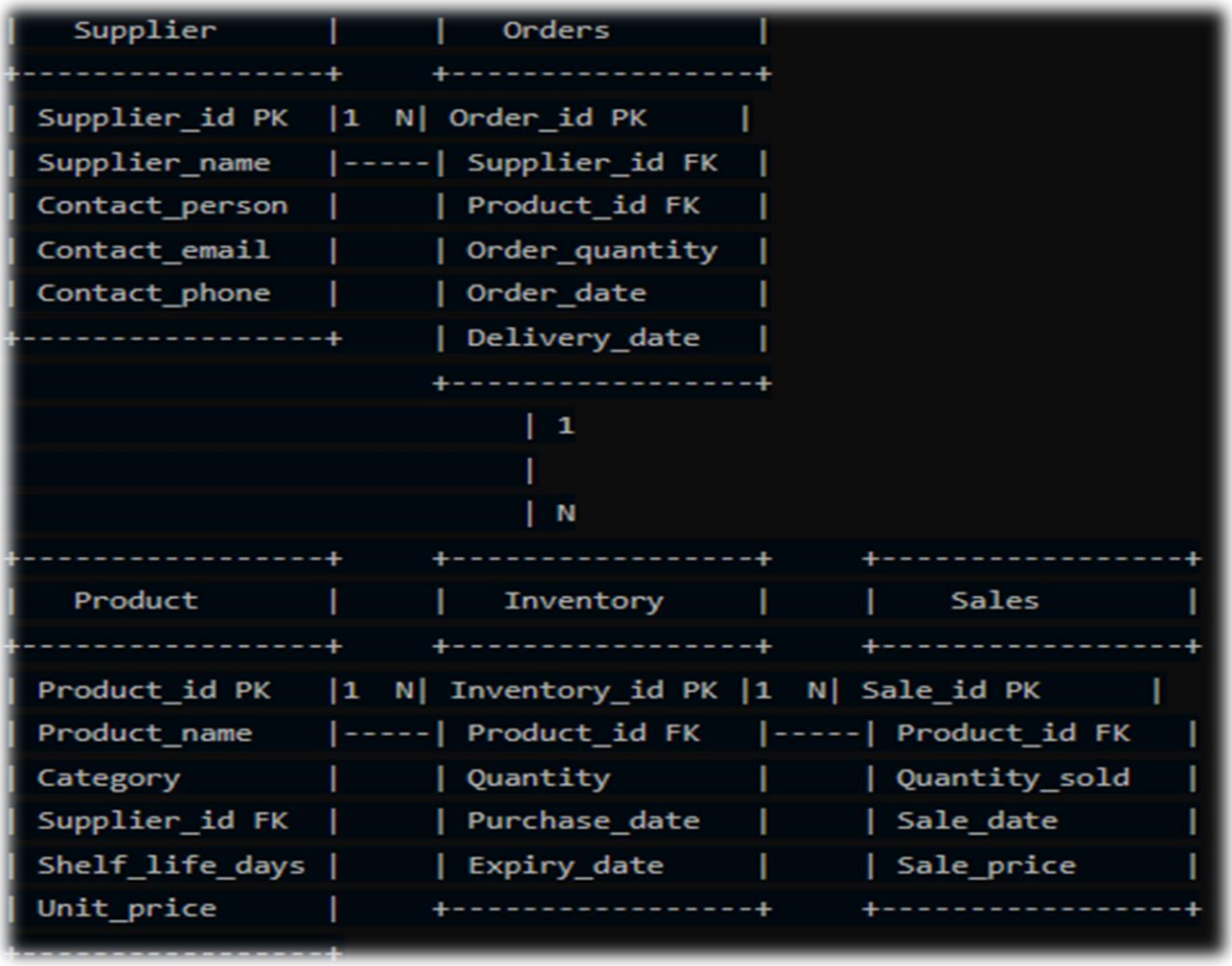
- ## Enhanced Reporting:

  Create comprehensive reports on inventory performance, including metrics such as stock levels, wastage, and sales, to facilitate better decision-making by management.

Project Objective:

To develop a SQL-based inventory management system that accurately tracks and manages the inventory of perishable goods, ensuring minimal waste and optimal stock levels.

In this ER diagram:



Commands:

```
create table Product(
Product_id int primary key auto_increment,
Product_name varchar(255),
category varchar(255),
supplier_id int ,
shelf_life_days int,
unit_price decimal(10,2)
);


insert into
Product(Product_id,Product_name,category,supplier_id,shelf_life_days,
unit_price)
values
(1, 'Milk', 'Dairy', 7, 8, 2.99),
(2, 'Apples', 'Fruit', 3, 25, 1.50),
(3, 'Chicken Breast', 'Meat', 1, 29, 7.99),
(4, 'Yogurt', 'Dairy', 10, 10, 3.50),
(5, 'Bananas', 'Fruit', 5, 16, 0.99),
(6, 'Ground Beef', 'Meat', 3, 24, 5.99),
(7, 'Cheese', 'Dairy', 9, 26, 4.50),
(8, 'Oranges', 'Fruit', 8, 17, 2.00),
(9, 'Pork Chops', 'Meat', 4, 20, 6.49),
(10, 'Butter', 'Dairy', 2, 22, 2.99),
(11, 'Grapes', 'Fruit', 6, 18, 2.99),
```

```
(12, 'Turkey', 'Meat', 1, 21, 8.99),
(13, 'Cream', 'Dairy', 5, 11, 2.50),
(14, 'Strawberries', 'Fruit', 7, 19, 3.50);
CREATE TABLE Suppliers (
   supplier_id INT PRIMARY KEY AUTO_INCREMENT,
   supplier_name VARCHAR(255),
   contact_person VARCHAR(255),
   contact_email VARCHAR(255),
   contact_phone VARCHAR(15)
);
INSERT INTO Suppliers (supplier_name, contact_person, contact_email,
contact_phone) VALUES
('Dairy Supplies Inc.', 'John Doe', 'john@dairysupplies.com', '123-456-7890'),
('Fruit World', 'Jane Smith', 'jane@fruitworld.com', '234-567-8901'),
('Veggie Delight', 'Mike Brown', 'mike@veggiedelight.com', '345-678-9012'),
('Meat Masters', 'Sara Wilson', 'sara@meatmasters.com', '456-789-0123'),
('Fruit World', 'Emily Davis', 'emilydavis@fruitworld.com', '123-456-7898'),
('Meat Masters', 'David Wilson', 'davidwilson@meatmasters.com', '123-456-
7899'),
('Healthy Harvest', 'Sarah Johnson', 'sarahjohnson@healthyharvest.com', '123-
456-7810'),
('Farm Fresh', 'James Williams', 'jameswilliams@farmfresh.com', '123-456-
7811'),
('Organic Orchard', 'Mary Jones', 'maryjones@organicorchard.com', '123-456-
7812'),
('Butcher's Best', 'Robert Miller', 'robertmiller@butchersbest.com', '123-456-
7813'),
```

('Veggie Delight', 'Patricia Taylor', 'patriciataylor@veggiedelight.com', '123-456-7814'),

('Sunrise Dairy', 'Charles Martinez', 'charlesmartinez@sunrisedairy.com', '123-456-7815'),

('Harvest Moon', 'Linda Hernandez', 'lindahernandez@harvestmoon.com', '123-456-7816'),

('Nature’s Bounty', 'Christopher Moore', 'christophermoore@naturesbounty.com', '123-456-7817'),

('Prime Produce', 'Barbara Jackson', 'barbarajackson@primeproduce.com', '123-456-7818'),

('Quality Meats', 'Daniel White', 'danielwhite@qualitymeats.com', '123-456-7819'),

('Pure Pastures', 'Susan Lee', 'susanlee@purepastures.com', '123-456-7820');

```
CREATE TABLE Inventory (
inventory_id INT PRIMARY KEY AUTO_INCREMENT,
    product_id INT,
    quantity INT,
    purchase_date DATE,
    expiry_date DATE
);
```

```
INSERT INTO Inventory (product_id, quantity, purchase_date, expiry_date)
VALUES
(1001, 100, '2024-07-01', '2024-07-11'),
```

(2001, 50, '2024-07-02', '2024-07-17'),

(3001, 200, '2024-07-03', '2024-07-10'),

(4001, 150, '2024-07-04', '2024-07-24'),

(5001, 75, '2024-07-05', '2024-07-10'),

(6001, 100, '2024-07-10', '2024-08-10'),

(7001, 50, '2024-07-11', '2024-07-25'),

(8001, 200, '2024-07-12', '2024-07-27'),

(9001, 75, '2024-07-13', '2024-07-28'),

(1010, 120, '2024-07-14', '2024-07-30'),

(1100, 30, '2024-07-15', '2024-08-01'),

(1200, 90, '2024-07-16', '2024-08-02'),

(1300, 110, '2024-07-17', '2024-08-03'),

(1400, 60, '2024-07-18', '2024-08-04'),

(1500, 150, '2024-07-19', '2024-08-05'),

(1600, 80, '2024-07-20', '2024-08-06'),

(1700, 170, '2024-07-21', '2024-08-07'),

(1800, 40, '2024-07-22', '2024-08-08'),

(1900, 130, '2024-07-23', '2024-08-09'),

(2012, 200, '2024-07-24', '2024-08-10');

CREATE TABLE Sales (

sale_id INT PRIMARY KEY AUTO_INCREMENT,

product_id INT,

```
    quantity_sold INT,
    sale_date DATE,
    sale_price DECIMAL(10,2)
);
INSERT INTO Sales(sale_id, product_id, quantity_sold, sale_date, sale_price)
VALUES
(6, 1001, 20, '2024-07-10', 5.99),
(7, 2002, 15, '2024-07-11', 3.49),
(8, 3003, 25, '2024-07-12', 6.99),
(9, 4004, 30, '2024-07-13', 7.99),
(10, 5005, 10, '2024-07-14', 4.99),
(11, 6001, 40, '2024-07-15', 8.99),
(12, 7001, 5, '2024-07-16', 2.99),
(13, 8001, 50, '2024-07-17', 9.99),
(14, 9001, 60, '2024-07-18', 3.99),
(15, 1010, 35, '2024-07-19', 1.99),
(16, 1100, 45, '2024-07-20', 5.49),
(17, 1200, 20, '2024-07-21', 4.49),
(18, 1300, 25, '2024-07-22', 6.49),
(19, 1400, 30, '2024-07-23', 7.49),
(20, 1500, 55, '2024-07-24', 8.49);


CREATE TABLE Orders (
    order_id INT PRIMARY KEY AUTO_INCREMENT,
```

```
    supplier_id INT,
    product_id INT,
    order_quantity INT,
    order_date DATE,
    delivery_date DATE
);
INSERT INTO Orders(order_id, supplier_id, product_id, order_quantity,
order_date, delivery_date) VALUES
(8, 1, 1001, 100, '2024-07-10', '2024-07-11'),
(9, 2, 2001, 50, '2024-07-11', '2024-07-12'),
(10, 3, 3001, 200, '2024-07-12', '2024-07-13'),
(11, 4, 4001, 75, '2024-07-13', '2024-07-14'),
(12, 5, 5001, 120, '2024-07-14', '2024-07-15'),
(13, 6, 6001, 30, '2024-07-15', '2024-07-16'),
(14, 7, 7001, 90, '2024-07-16', '2024-07-17'),
(15, 8, 8001, 110, '2024-07-17', '2024-07-18'),
(16, 9, 9001, 60, '2024-07-18', '2024-07-19'),
(17, 10, 1010, 150, '2024-07-19', '2024-07-20'),
(18, 11, 1100, 80, '2024-07-20', '2024-07-21'),
(19, 12, 1200, 170, '2024-07-21', '2024-07-22'),
(20, 13, 1300, 40, '2024-07-22', '2024-07-23'),
(21, 14, 1400, 130, '2024-07-23', '2024-07-24');
```

Normal Queries

- **find the all products that have a shelf life of less than 10 days**

```
select * from Product where shelf_life_days < 10;
```

- **find the max unit price fro product**

```
SELECT MAX(unit_Price) AS MaxPrice FROM Product;
```

- **find the average unit price of product**

```
SELECT Category, AVG(unit_Price) AS AveragePrice
FROM Product
GROUP BY Category;
```

- **find all products with a unit price higher than the average unit price in the product table**

```
SELECT Product_Name, unit_Price
FROM Product
WHERE unit_Price > (SELECT AVG(unit_Price) FROM Product);
```

- **Update Supplier Contact person**

```
select * from suppliers;
UPDATE Suppliers SET Contact_person = 'John lee' WHERE
Supplier_ID = 1;
```

Table Description

Product

Field	Type	Null	Key	Default	Extra
Product_id	int	no	PRI	null	auto_increment
Product_name	varchar(255)	no		null	
category	varchar(255)	no		null	
Supplier_id	int	no	FK	null	
shelf_life_days	int	no		null	
unit_price	decimal	no		null	

Supplier

Field	Type	Null	Key	Default	Extra
Supplier_id	int	no	PRI	null	auto_increment
Supplier_name	varchar(255)	no		null	
conatact_person	varchar(255)	no		null	
contact_email	varchar(255)	no		null	
contact_phone	varchar(15)	no		null	

Inventory

Field	Type	Null	Key	Default	Extra
inventory_id	int	no	PRI	null	auto_increment
product_id	int	no		null	
quantity_id	int	no		null	
purchase_date	date	no		null	
expiry_date	date	no		null	

Sales

Field	Type	Null	Key	Default	Extra
Sale_id	int	no	PRI	null	auto_increment
product_id	int	no		null	
quantity_sold	int	no		null	
sale_date	date	no		null	
sale_price	decimal	no		null	

Orders

Field	Type	Null	Key	Default	Extra
order_id	int	no	PRI	null	auto_increment
supplier_id	int	no		null	
product_id	int	no		null	
order_quantity	int	no		null	
order_date	date	no		null	
delivery_date	date	no		null	

Sub-Queries

- **Find Suppliers Who Supply more then 2 Product*/**

SELECT Supplier_Name FROM Suppliers

```
WHERE Supplier_ID IN (
   SELECT Supplier_ID
   FROM Product
   GROUP BY Supplier_ID
   HAVING COUNT(Product_ID) < 2
);
```

- **Find Products Not Ordered in the Last 6 Months*/**

SELECT Product_Name

```
FROM Product
WHERE Product_ID NOT IN (
   SELECT Product_ID
   FROM Orders
   WHERE Order_Date >= CURDATE() - INTERVAL 6 MONTH
);
```

- **avg price of products sold**

```
select product_name,
(select avg(sale_price)
from sales
where sales.product_id = product.product_id)as average_sale_price
from product;
```

- **supplier who supplied products with more than 15 days shelf life**

```
select supplier_name from suppliers where supplier_id in (select
supplier_id from product where shelf_life_days > 15);
```

- **product with sales higher than the average sales quatity**

```
select product_name from product where product_id in (select
product_id from sales group by product_id
having sum(quantity_sold) > (select avg(quantity_sold) from sales));
```

- **product with the highest inventory quantity**

```
select product_name from product where product_id = (select
product_id from inventory order by quantity desc limit 1);
```

- **set the last order date for each product based on the orders table**

```
alter table product add column last_order_date date;
update product
set last_order_date = (
select max(order_date)
from orders
where orders.product_id = product.product_id);
select * from product;
```

- **add a column to supplier to track the total number of products they supply**

```
alter table suppliers add column total_products_supplied int;
update suppliers
set total_products_supplied =(
select count(*)
from product
where product.supplier_id = suppliers.supplier_id);
```

- **set the quantity of all products in inventory to twice the quality of the least stocked product**

```
update inventory
set quantity =(
select min(quantity)*2
from inventory);
```

- **total sales quantity for each product category**

```
select category, (select sum(quantity_sold)
from sales where product_id in (select product_id
from product
where category = p.category)) as total_sales_quantity
from product p
group by category;
```

Joins Queries

- **select products and their latest inventory info**

```
select p.Product_name, i.quantity, i.purchase_date
from product p
left join inventory i on p.Product_id = i.product_id
order by i.purchase_date desc;
```

- **select suppliers with product price above 0.50**

```
select * from product;
select s.supplier_name, p.Product_name, p.unit_price
from suppliers s
inner join product p on s.supplier_id = p.supplier_id
where p.unit_price > 0.50;
```

- **Product and their supplier**

```
select p.product_name, s.supplier_name
from product p
inner join suppliers s on p.supplier_id = s.supplier_id;
```

- **Total sales quantity for each product category**

```
select p.category,
coalesce(sum(sa.quantity_sold), 0) as total_sales_quantity
```

```
from product p
left join sales sa on p.product_id = sa.product_id
group by p.category;
```

- **products and their suppliers**

```
select p.product_name, s.supplier_name from product p
inner join suppliers s on p.supplier_id = s.supplier_id;
```

- **sales and corresponding product and supplier detail**

```
select sa.sale_id, p.product_name, s.supplier_name,
sa.quantity_sold, sa.sale_date, sa.sale_price
from sales sa
inner join product p on sa.product_id = p.product_id
inner join suppliers s on p.supplier_id = s.supplier_id;
```

Conclusion

The **Smart Inventory Management System for Perishable Goods** project effectively demonstrates how SQL can be utilized to manage and optimize the inventory of perishable products. By designing a database schema with five interconnected tables—Products, Suppliers, Sales, Orders, and Inventory Actions—we were able to capture and manage essential data related to product details, supplier information, sales transactions, order management, and inventory adjustments.

Through the execution of 20 SQL queries, including subqueries and joins, the project successfully addressed key operational questions, such as tracking stock levels, monitoring expiration dates, managing supplier relationships, and analyzing sales performance. This comprehensive system ensures that perishable goods are managed efficiently, minimizing waste and maximizing profitability.

Key features of this system include:

- **Real-time Inventory Tracking:** The system allows for continuous monitoring of stock levels and expiration dates, helping to reduce losses due to spoilage.

- **Supplier and Order Management:** It facilitates effective communication and coordination with suppliers, ensuring timely restocking of inventory.

- **Sales and Revenue Analysis:** The system provides valuable insights into sales trends and revenue generation, aiding in data-driven decision-making.