INVENTORY MANAGEMENT SYSTEM DATABASE

OVERVIEW:

An Inventory Management System (IMS) database is a structured digital repository designed to efficiently track, organize, and manage an organization's inventory or stock of goods. It serves as the backbone of an IMS application, allowing users to store, retrieve, and manipulate data related to products, suppliers, and related inventory transactions.

DATABASE SCHEMA

The database schema typically includes tables for Customers, Products, Suppliers and Inventory transactions with well-defined relationships and attributes.

1. Customer Table: This table stores information about the different customers including their name, location and contact details and Customer ID which has Primary Key (unique identifier).

2. Products Table: This table stores information about individual inventory products, including Product ID which has Primary Key (unique identifiers) and identity (use to continue series of number), names, quantities in stock, prices, and category.

3. Suppliers Table: It maintains details about the suppliers or vendors who provide the inventory products. Supplier information includes supplier ID which has primary Key (unique identifiers) and identity (use to continue series of number), name, contact details.

4. Inventory Transactions Table: This table records all inventory related transactions, such as product purchases, sales. Each transaction entry contains information like transaction ID which has Primary Key (unique identifiers) and identity (use to continue series of number), product ID which is reference from product table, quantity, transaction type (e.g., purchase, sale) and possible payment terms.

SQL QUERIES

Step 1: Created Database and used it.

```
Create Database Inventory;
Use Inventory;
```

Step 2: Creation of table.

1) Customer Table

```
Create Table Customers(
 Customer_ID Varchar(5) Primary Key,
 First_Name Varchar(10),
 Last_Name Varchar(15),
 Customer_Location Varchar(50),
 Contact_Number Int
 );
```

2) Products Table

```
CREATE TABLE Products (
 Product_ID INT Primary Key Identity,
  Product_Name VARCHAR(50),
  Category VARCHAR(50),
  Quantity INT,
 Price Float
);
```

3) Suppliers Table

```
CREATE TABLE Suppliers (
   Supplier_ID Int Identity(101,1) PRIMARY KEY,
   Supplier_Name VARCHAR(100),
   Email VARCHAR(100),
   City VARCHAR(50),
   Country VARCHAR(50)
);
```

4) Inventory Transaction

```
CREATE TABLE Inventory_Transaction (
  Transaction_ID INT Primary Key Identity(101,1),
  Product_ID INT not null references Products(Product_ID),
 Transaction_Type VARCHAR(15) not null,
  Tax DECIMAL(10, 2),
 Shipping_Cost DECIMAL(10, 2),
 Quantity INT,
 Total_Cost DECIMAL(12, 2)
);
```

Step 3: Implementing CRUD Operations

1)Insertion of data in customer table

```
INSERT INTO Customers(Customer_ID,
First_Name,Last_Name,Customer_Location,Contact_Number)
VALUES ('ID 1','Jane','Doe','London',38769806),
       ('ID 2','Alice','Jones','London',85570007),
       ('ID 3','Noel','James','New York',43844108),
       ('ID 4','Alberto','Gonzales','London',5558787),
       ('ID 5','Jean','Havok','Miami',081234567),
       ('ID 6','Neha','Sharma','Delhi',0319876543),
       ('ID 7','Raj','Singh','Mumbai',5559931);
```

2) Insertion of data in product table

```
INSERT INTO Products(Category,Product_Name,Quantity, Price)
VALUES
    ('Electronics', 'Laptop', 10, 800),
    ('Electronics', 'Mouse', 50, 20),
    ('Clothing', 'T-shirt',  100, 15),
    ('Electronics', 'Smartphone', 30, 700),
    ('Electronics', 'Keyboard',20, 40),
    ('Clothing', 'Jeans', 25, 30),
    ('Electronics', 'Headphones', 40, 50),
    ('Electronics', 'Laptop', 15, 900),
    ('Clothing', 'T-shirt',70, 20),
    ('Electronics', 'Smartphone',45, 750),
    ('Clothing', 'Jeans',20, 35),
    ('Electronics', 'Keyboard',10, 25),
   ('Beverages','Chai',10,18),
   ('Beverages','Chang',24,19),
   ('Condiments','Aniseed Syrup',12,10),
    ('Seafood','Ikura',15,50);
```

3) Insertion of data in supplier table

```
    INSERT INTO Suppliers(Supplier_Name, Email, City, Country)
    VALUES
        ('John Doe', 'john.doe@example.com', 'New York', 'USA'),
        ('Jane Smith', 'jane.smith@example.com', 'London', 'UK'),
        ('Michael Johnson', 'michael.j@example.com', 'Los Angeles', 'USA'),
        ('Emily Williams', 'emily.w@example.com', 'Chicago', 'USA'),
        ('William Brown', 'will.brown@example.com', 'Toronto', 'Canada'),
        ('Olivia Jones', 'olivia.j@example.com', 'Sydney', 'Australia'),
        ('James Miller', 'james.m@example.com', 'Paris', 'France'),
        ('Sophia Davis', 'sophia.d@example.com', 'Berlin', 'Germany'),
        ('Benjamin Garcia', 'benjamin.g@example.com', 'Tokyo', 'Japan'),
        ('Isabella Martinez', 'isabella.m@example.com', 'Seoul', 'South Korea'),
        ('Alexander Rodriguez', 'alex.r@example.com', 'Moscow', 'Russia'),
        ('Mia Lopez', 'mia.l@example.com', 'Mumbai', 'India'),
        ('Ethan Hernandez', 'ethan.h@example.com', 'Sao Paulo', 'Brazil'),
        ('Ava Gonzalez', 'ava.g@example.com', 'Shanghai', 'China'),
        ('Daniel Wilson', 'daniel.w@example.com', 'Istanbul', 'Turkey'),
        ('Charlotte Anderson', 'charlotte.a@example.com', 'Lahore', 'Pakistan'),
        ('Matthew Taylor', 'matthew.t@example.com', 'Jakarta', 'Indonesia'),
        ('Amelia Lee', 'amelia.l@example.com', 'Lagos', 'Nigeria'),
         ('Oliver Wright', 'oliver.w@example.com', 'Cairo', 'Egypt');
```

4) Insertion of data in Inventory transaction table

```
   INSERT INTo
Inventory_Transaction(Product_ID,Transaction_Type,Tax,Shipping_Cost,Quantity,Tot
al_Cost)
VALUES
    (1,'Purchase', 12.50, 8.00,5,170.50),
    (2,'Purchase', 15.00, 10.00,3,205.00),
    (3,'Sale', 8.50, 5.00,2,113.50),
    (4,'Sale', 12.50, 8.00,5,170.50),
    (5,'Purchase', 15.00, 10.00,3,205.00),
    (6,'Sale',8.50, 5.00,2,113.50),
    (7,'Sale', 12.50, 8.00,5,170.50),
    (8,'Purchase', 15.00, 10.00,3,205.00),
     (9,'Sale', 8.50, 5.00,2,113.50),
     (10,'Purchase', 12.50, 8.00,5, 170.50);
```

5) Updating data in Product table

```
Update Products
set Price= 15000
where Product_ID=3;
```

6) Deleting data from Product table

```
Delete from Products
where Product_ID=8;
```

Step 4: Renamed column

```
EXEC sp_rename'Customers.Customer_Location','Location','Column';
```

Step 5: Retrieving and displaying information

1) To see how many table exist in database

```
Select * from sys.tables;
```

2) To see information in specific table

```
Select * from Products;
Select * from Suppliers;
Select * from Customers;
Select * from Inventory_Transaction;
```

3) To see information using specific condition

```
Select * from Customers
where Customer_ID = 'ID 4';


Select Transaction_ID,Quantity,Total_Cost
from Inventory_Transaction
where Product_ID = 5;


Select * from Suppliers
where Supplier_Name = 'Alexander Rodriguez';


Select * from Products
where Category= 'Electronics';
```

4) To see data of two different tables

```
select p.Product_ID,p.Product_Name,t.Total_Cost,t.Transaction_Type from Products
as p full join Inventory_Transaction as t
On p.Product_ID = t.Product_ID;


select p.Product_ID,p.Product_Name,t.Total_Cost,t.Transaction_Type from Products
as p inner join Inventory_Transaction as t
On p.Product_ID = t.Product_ID;
```

Step 6: Filtering and sorting Data

1)Using single table

```
Select * from Inventory_Transaction
```

```sql
where Total_Cost >= 150
order by Transaction_ID asc;
```

2) Using 2 different table

```sql
Select p.Product_ID,p.Product_Name,p.Price,t.Total_Cost
from Products as p Full Join Inventory_Transaction as t
on p.Product_ID = t.Product_ID
where Price > =750
order by Product_ID;
```

Testing

1. Data Insertion: Users can add new inventory, update existing product details, and record supplier information through the system's user interface.

2. Data Updates: The database continuously updates product quantities based on transactions, ensuring accurate real-time tracking of inventory levels.

3. Supplier Management: Supplier details can be retrieved and maintained, helping organizations manage their relationships with suppliers effectively.

4. Reporting: Users can generate reports based on the data in the database, such as inventory levels, transaction history, and supplier performance.

5. Scalability: The database can accommodate the growth of inventory and transactions over time.

Challenges Faced

- Designing an effective database schema that meets the project’s requirements.
- Identifying and resolving errors in SQL code.
- Maintaining accurate and up-to date documentation for the database schema and SQL queries.

Conclusion

In this project we developed a complete database in which we can update the stock, modify stock, we can forecast the stock, generate invoice. From this database we can get an update that if a particular inventory or stock is less than some pre-fixed quantity then it’ll be easy for the manager/owner to reorder the product from supplier to overcome the “Out of Stock” stage.

In addition to this it can also help us to manage or add customer details, products details, transaction details which can be proved as very useful feature. We can have complete customer details which can help us to retrieve the order details of regular customers. From this program we can also keep a track of transactions performed by different customers/clients. We can also get an idea that how much fund we received from different payment methodologies.

***