GLOBAL SUPPLY CHAIN MANAGEMENT SYSTEM

MAHVISH KOUNAIN (NF1014902)

Tharun Kumar Reddy, Devapatla (NF1015299)

SAI KUMAR GURRAM (NF1015899)

Masters in Data Analytics, University of Niagara Falls Canada Fall 2024

SQL Databases (CPSC-500-1)

Dr. Abbas Hamze

December 9,2024

INTRODUCTION

The global supply chain management system is a tool designed to make managing supply chains across industries easier and more efficient. With markets becoming more global, having a reliable and adaptable system is crucial. This project uses SQL to create a database that can store, retrieve, and analyze a lot of data about suppliers, manufacturers, logistics, and customers.

Using SQL, this system helps solve common problems in supply chain management, like keeping track of inventory, processing orders, managing supplier relationships, and handling logistics. The aim is to provide tools for managing data, so decisions can be made based on up-to-date and historical information. This project shows how powerful SQL can be in managing complex data and its potential to drive growth in supply chain management.

PROJECT OVERVIEW

Project Selection: The global supply chain management system was chosen to address the increasing complexity and globalization of supply chains. Efficient management of supply chains is crucial for businesses to remain competitive and meet customer demands. This project aims to tackle challenges such as inventory control, order processing, supplier relationship management, and transportation logistics by leveraging the capabilities of SQL.

ER Diagram creation: Entity-Relationship (ER) diagram was created to visually represent the database structure of the global supply chain management system. It identifies the key entities such as suppliers, manufacturers, logistics, and customers, and outlines the relationships between them. This diagram helps in understanding the data flow and ensuring the database design supports efficient data management and retrieval. By clearly mapping out these connections, the ER diagram serves as a blueprint for building a robust SQL database.

Schema creation: It defines the database structure, including tables, fields, and data types, to represent relationships between entities. It ensures efficient data storage and retrieval.

Table creation: A total of 10 tables were created within the global_supply_chain schema to store data related to suppliers, manufacturers, products, orders, inventory, logistics, customers, shipments, payments, and returns**.**

Data Insertion: Realistic data was inserted into each table to simulate a functioning global supply chain management system. This data includes information on suppliers, manufacturers, products, orders, inventory levels, logistics details, customer information, shipment records, payment transactions, and return processes**.**

Data Query Language Implementation: The Data Query Language (DQL) implementation involved executing various SQL queries to retrieve and analyze data from the Global Supply Chain Management System Database. These queries were designed to extract meaningful insights, such as tracking inventory levels, analyzing supplier performance, and monitoring order fulfillment rates. By leveraging DQL, we were able to generate valuable reports and visualizations that support data-driven decision-making and enhance the overall efficiency and performance of the supply chain management system.

ER DIAGRAM

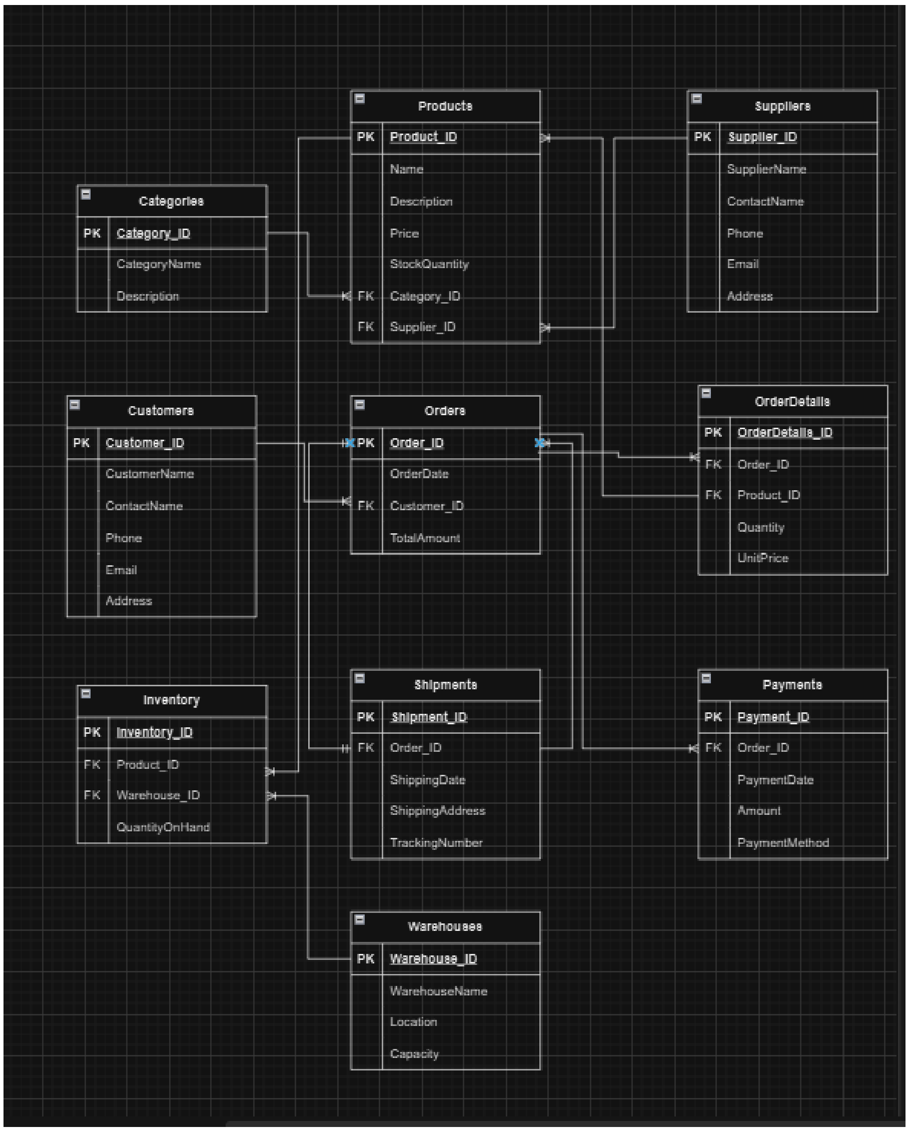
The ER diagram provides a clear visual representation of the structure and relationships within the Global Supply Chain Management System Database. It helps in understanding how different entities interact with each other, which is essential for designing and managing the database effectively. By analyzing these relationships, businesses can gain valuable insights into inventory levels, supplier performance, order fulfillment, and payment processing, ultimately contributing to the efficiency and success of supply chain operations.

Entities and Relationships:

1. **Categories:**
    - Represents the various categories of products.
    - **Attributes:** category_id, name, description.
2. **Products:**
    - Represents the products in the supply chain.
    - **Attributes:** product_id, name, category_id, description, price.
    - **Relationships:** Each product is linked to one category.
3. **Suppliers:**
    - Represents the suppliers providing goods and materials.
    - **Attributes:** supplier_id, name, contact_info, address, rating.
4. **Customers:**
    - Represents the customers placing orders.
    - **Attributes:** customer_id, name, email, phone_number, address.
    - **Relationships:** Each customer can place multiple orders.
5. **Orders:**
    - Represents the orders placed by customers.
    - **Attributes:** order_id, customer_id, order_date, status.
    - **Relationships:** Each order is linked to one customer and can contain multiple order details.
6. **Order Details:**

- Represents the details of each order.
- **Attributes:** order_detail_id, order_id, product_id, quantity, price.
- **Relationships:** Each order detail is linked to one order and one product.

7. **Inventory:**
    - Represents the stock levels of products.
    - **Attributes:** inventory_id, product_id, warehouse_id, quantity_on_hand, reorder_level.
    - **Relationships:** Each inventory record is associated with one product and one warehouse.
8. **Shipments:**
    - Represents the shipment details for orders.
    - **Attributes:** shipment_id, order_id, shipment_date, delivery_date, status.
    - **Relationships:** Each shipment is linked to one order.
9. **Payments:**
    - Represents the financial transactions for orders.
    - **Attributes:** payment_id, order_id, amount, payment_date, payment_method.
    - **Relationships:** Each payment is linked to one order.
10. **Warehouses:**
    - Represents the storage locations for inventory.
    - **Attributes:** warehouse_id, name, location, capacity.
    - **Relationships:** Each warehouse can store multiple inventory records.



ER diagram link:
https://drive.google.com/file/d/1WkMNvoNIA3xWGCJjP1dVSIfNebXpuUki/view?usp=sharing

SCHEMA DEFINATION

The schema serves as the blueprint of the database, outlining its structure and organization. For this project, the schema is named Global_Supply_Chain. This schema is designed to organize and manage all the tables and their relationships, ensuring a structured and efficient database design.

The schema creation process for the Global Supply Chain Management System Database involved defining the schema, creating tables, establishing relationships, and setting constraints. This structured approach ensures that the data is organized, consistent, and easily accessible, providing a solid foundation for managing and analyzing supply chain-related data.

TABLES CREATION

A total of 10 tables were created within the Global_Supply_Chain schema to store data related to various entities. Here are the tables and their primary attributes:

Table name	Primary attributes
Categories	category_id, name, description
Products	product_id, name, category_id, description, price
Suppliers	supplier_id, name, contact_info, address, rating
Customers	customer_id, name, email, phone_number, address
Orders	order_id, customer_id, order_date, status
Order Details	order_detail_id, order_id, product_id, quantity, price
Inventory	inventory_id, product_id, warehouse_id, quantity_on_hand, reorder_level
Shipments	shipment_id, order_id, shipment_date, delivery_date, status
Payments	payment_id, order_id, amount, payment_date, payment_method
Warehouses	warehouse_id, name, location, capacity

Key Constraints

The key constraints added to the Global Supply Chain Management System Database ensure data integrity, consistency, and uniqueness. Primary keys uniquely identify each record, foreign keys establish relationships between tables, unique constraints prevent duplicate entries, and not null constraints enforce the presence of data. These constraints provide a robust framework for managing and analyzing supply chain-related data effectively.

1. Primary Keys

Primary keys are unique identifiers for each record in a table. They ensure that each record can be uniquely identified and accessed. In this project, primary keys were added to the following tables:

- **Categories:** category_id
- **Products:** product_id
- **Suppliers:** supplier_id
- **Customers:** customer_id
- **Orders:** order_id
- **Order Details:** order_detail_id
- **Inventory:** inventory_id
- **Shipments:** shipment_id
- **Payments:** payment_id
- **Warehouses:** warehouse_id

2. Foreign Keys

Foreign keys establish relationships between tables, ensuring data integrity and consistency. They link one table to another by referencing the primary key of the related table. In this project, foreign keys were added as follows:

- **Products:**
  - category_id references Categories(category_id)
  - supplier_id references Suppliers(supplier_id)
- **Orders:**
  - customer_id references Customers(customer_id)
- **Order Details:**
  - order_id references Orders(order_id)
  - product_id references Products(product_id)
- **Inventory:**

    - product_id references Products(product_id)
    - warehouse_id references Warehouses(warehouse_id)
- **Shipments:**
    - order_id references Orders(order_id)
- **Payments:**
    - order_id references Orders(order_id)

3. Unique Constraints

Unique constraints ensure that certain columns have unique values, preventing duplicate entries. In this project, unique constraints were added to the following columns:

- **Suppliers:** name
- **Customers:** email

4. Not Null Constraints

Not null constraints ensure that certain columns cannot have null values, enforcing the presence of data in these columns. In this project, not null constraints were added to the following columns:

- **Categories:** name
- **Products:** name, price
- **Suppliers:** name, contact_info, address
- **Customers:** name, email, phone_number, address
- **Orders:** customer_id, order_date, status
- **Order Details:** order_id, product_id, quantity, price
- **Inventory:** product_id, warehouse_id, quantity_on_hand, reorder_level
- **Shipments:** order_id, shipment_date, delivery_date, status
- **Payments:** order_id, amount, payment_date, payment_method
- **Warehouses:** name, location, capacity

Relationship Establishment

Relationships between tables were defined using foreign keys to ensure data integrity and consistency. Here are some key relationships:

- **Products:**
    - category_id references Categories(category_id)
    - supplier_id references Suppliers(supplier_id)

- **Orders:**
    - customer_id references Customers(customer_id)
- **Order Details:**
    - order_id references Orders(order_id)
    - product_id references Products(product_id)
- **Inventory:**
    - product_id references Products(product_id)
    - warehouse_id references Warehouses(warehouse_id)
- **Shipments:**
    - order_id references Orders(order_id)
- **Payments:**
    - order_id references Orders(order_id)

DDL COMMANDS:

➨ DDL (Data Definition Language) Commands are SQL commands used to define, modify, or remove database structures, such as tables, indexes, and schemas. DDL commands do not manipulate data but focus on the structure of the database itself.

➨ DDL commands are essential for defining and structuring the database.

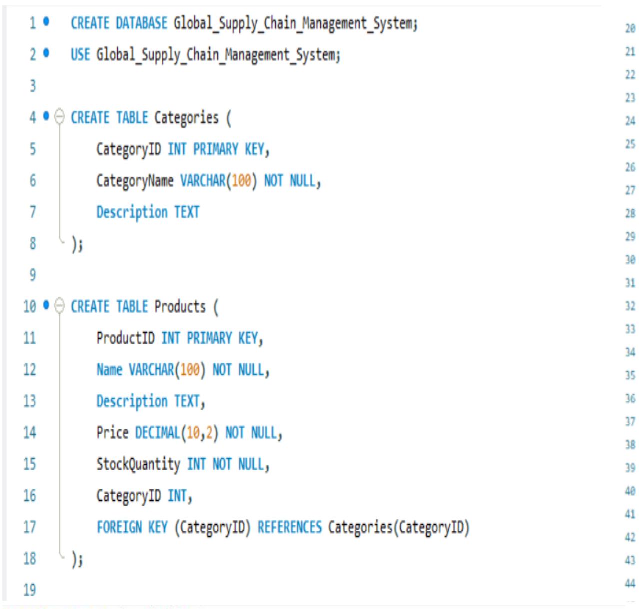
➨ Creating Tables and Defining Relationships:

- Utilized DDL statements to create the tables necessary for the supply chain management system.
- Defined relationships between tables using primary and foreign keys.

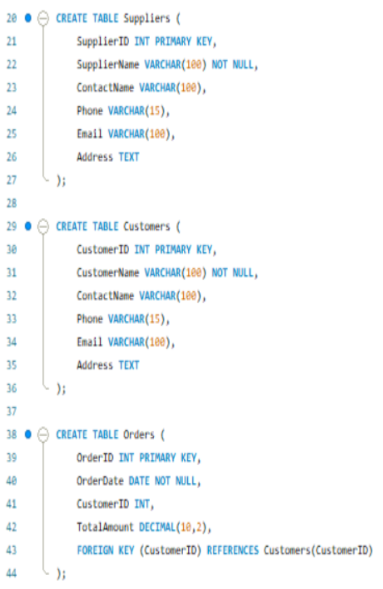
➨ Key Constraints Enforced:

- Primary Keys: Ensuring unique identification of records in each table.
- Foreign Keys: Establishing links between related tables.
- Unique Constraints: Guaranteeing the uniqueness of specific columns.
- Not Null Constraints: Ensuring essential fields are not left empty.
- Check Constraints: Enforcing domain-specific rules on data values.

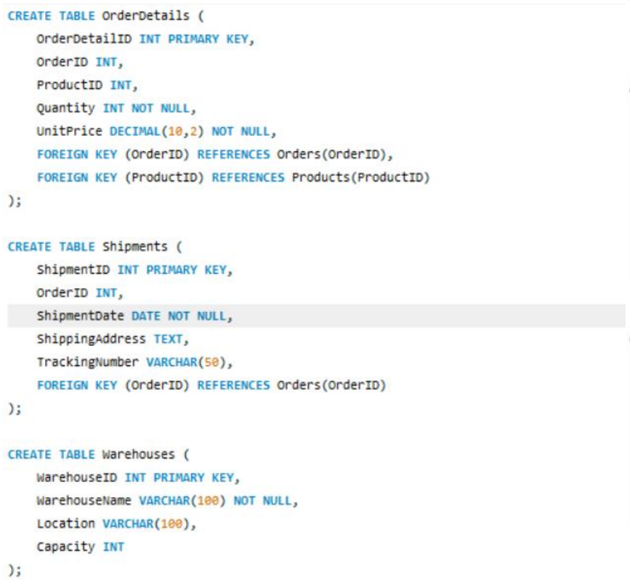
CREATING THE TABLES BY USING DDL COMMANDS



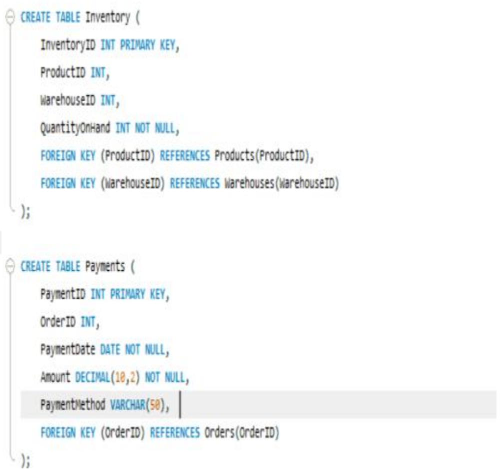
```sql
CREATE DATABASE Global_Supply_Chain_Management_System;
USE Global_Supply_Chain_Management_System;

CREATE TABLE Categories (
    CategoryID INT PRIMARY KEY,
    CategoryName VARCHAR(100) NOT NULL,
    Description TEXT
);

CREATE TABLE Products (
    ProductID INT PRIMARY KEY,
    Name VARCHAR(100) NOT NULL,
    Description TEXT,
    Price DECIMAL(10,2) NOT NULL,
    StockQuantity INT NOT NULL,
    CategoryID INT,
    FOREIGN KEY (CategoryID) REFERENCES Categories(CategoryID)
);

```



```sql
CREATE TABLE Suppliers (
    SupplierID INT PRIMARY KEY,
    SupplierName VARCHAR(100) NOT NULL,
    ContactName VARCHAR(100),
    Phone VARCHAR(15),
    Email VARCHAR(100),
    Address TEXT
);

CREATE TABLE Customers (
    CustomerID INT PRIMARY KEY,
    CustomerName VARCHAR(100) NOT NULL,
    ContactName VARCHAR(100),
    Phone VARCHAR(15),
    Email VARCHAR(100),
    Address TEXT
);

CREATE TABLE Orders (
    OrderID INT PRIMARY KEY,
    OrderDate DATE NOT NULL,
    CustomerID INT,
    TotalAmount DECIMAL(10,2),
    FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)
);
```



```sql
CREATE TABLE OrderDetails (
    OrderDetailID INT PRIMARY KEY,
    OrderID INT,
    ProductID INT,
    Quantity INT NOT NULL,
    UnitPrice DECIMAL(10,2) NOT NULL,
    FOREIGN KEY (OrderID) REFERENCES Orders(OrderID),
    FOREIGN KEY (ProductID) REFERENCES Products(ProductID)
);

CREATE TABLE Shipments (
    ShipmentID INT PRIMARY KEY,
    OrderID INT,
    ShipmentDate DATE NOT NULL,
    ShippingAddress TEXT,
    TrackingNumber VARCHAR(50),
    FOREIGN KEY (OrderID) REFERENCES Orders(OrderID)
);

CREATE TABLE Warehouses (
    WarehouseID INT PRIMARY KEY,
    WarehouseName VARCHAR(100) NOT NULL,
    Location VARCHAR(100),
    Capacity INT
);
```



```sql
CREATE TABLE Inventory (
    InventoryID INT PRIMARY KEY,
    ProductID INT,
    WarehouseID INT,
    QuantityOnHand INT NOT NULL,
    FOREIGN KEY (ProductID) REFERENCES Products(ProductID),
    FOREIGN KEY (WarehouseID) REFERENCES Warehouses(WarehouseID)
);

CREATE TABLE Payments (
    PaymentID INT PRIMARY KEY,
    OrderID INT,
    PaymentDate DATE NOT NULL,
    Amount DECIMAL(10,2) NOT NULL,
    PaymentMethod VARCHAR(50),
    FOREIGN KEY (OrderID) REFERENCES Orders(OrderID)
);
```

DML COMMANDS:

- DML (Data Manipulation Language) commands are SQL commands used for managing and manipulating data within database tables. Unlike DDL (Data Definition Language) commands, which define and structure the database, DML commands are used to insert, update, delete, and retrieve data stored in those tables.
- DML commands helps to perform operations on actual data within the database tables.
- Populating Tables with Realistic Data:

- Insert data into tables to simulate real-world scenarios.
- Ensure data reflects dynamic management within the supply chain.

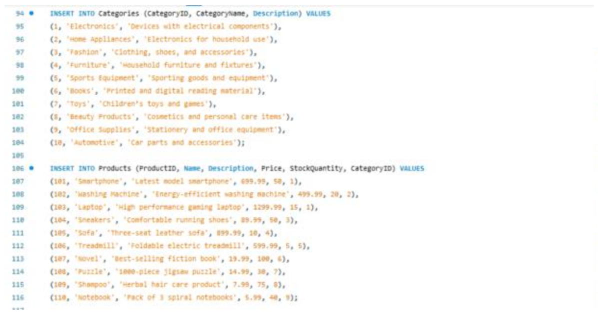
➨ Using DML Statements:

- Insert Records: Add new records to the database.
- Update Records: Modify existing records to reflect changes.
- Delete Records: Remove records that are no longer needed.

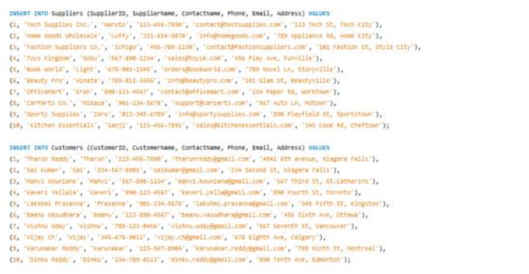
DATA INSERTION

The data insertion phase is pivotal in creating a functional and reliable Global Supply Chain Management System. By populating the database with realistic data, the system is equipped to handle real-world supply chain operations, providing a robust platform for managing and analyzing supply chain activities. The comprehensive dataset ensures that the database design is structured and efficient, supporting effective decision-making and enhancing overall supply chain performance.

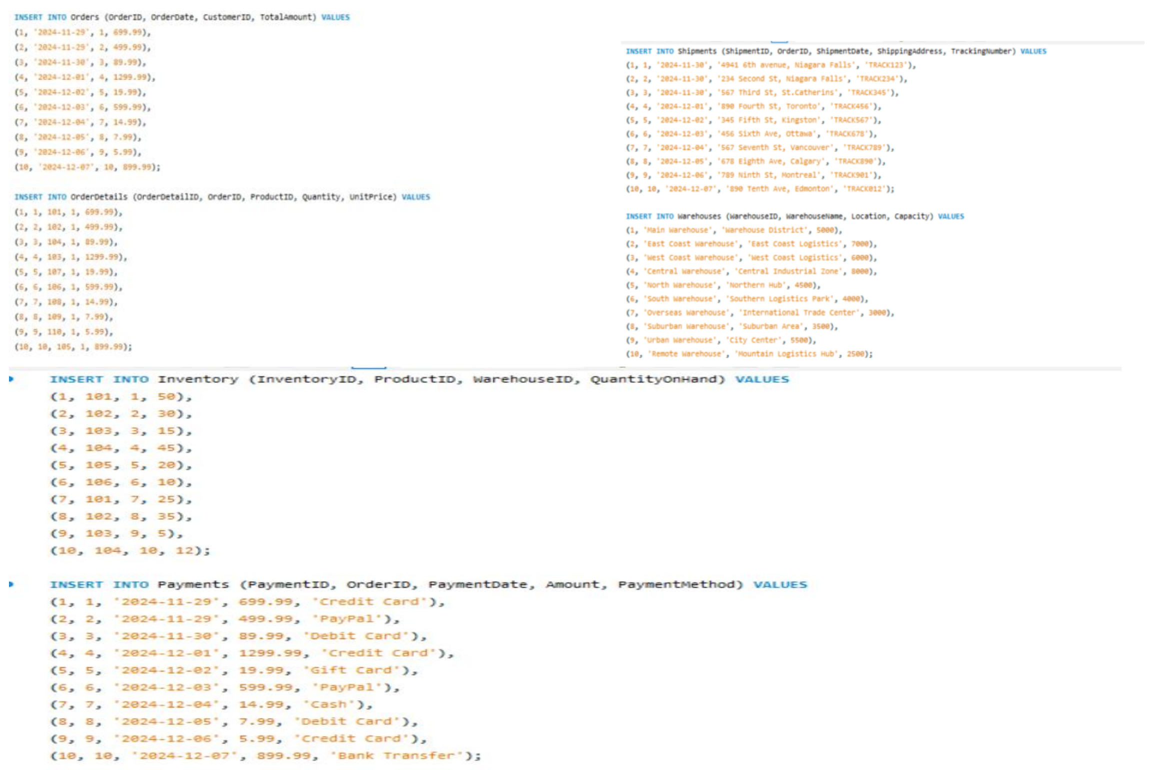
INERTING THE VALUES INTO THE TABLES BY USING DML COMMANDS:



```
INSERT INTO Categories (CategoryID, CategoryName, Description) VALUES
(1, 'Electronics', 'Devices with electrical components'),
(2, 'Home Appliances', 'Electronics for household use'),
(3, 'Fashion', 'Clothing, shoes, and accessories'),
(4, 'Furniture', 'Household furniture and fixtures'),
(5, 'Sports Equipment', 'Sporting goods and equipment'),
(6, 'Books', 'Printed and digital reading material'),
(7, 'Toys', 'Children's toys and games'),
(8, 'Beauty Products', 'Cosmetics and personal care items'),
(9, 'Office Supplies', 'Stationery and office equipment'),
(10, 'Automotive', 'Car parts and accessories');

INSERT INTO Products (ProductID, Name, Description, Price, StockQuantity, CategoryID) VALUES
(101, 'Smartphone', 'Latest model smartphone', 699.99, 50, 1),
(102, 'Washing Machine', 'Energy-efficient washing machine', 499.99, 20, 2),
(103, 'Laptop', 'High performance gaming laptop', 1299.99, 15, 1),
(104, 'Sneakers', 'Comfortable running shoes', 89.99, 50, 3),
(105, 'Sofa', 'Three-seat leather sofa', 899.99, 10, 4),
(106, 'Treadmill', 'Foldable electric treadmill', 599.99, 5, 5),
(107, 'Novel', 'Best-selling fiction book', 19.99, 100, 6),
(108, 'Puzzle', '1000-piece jigsaw puzzle', 14.99, 30, 7),
(109, 'Shampoo', 'Herbal hair care product', 7.99, 75, 8),
(110, 'Notebook', 'Pack of 3 spiral notebooks', 5.99, 40, 9);
```



```
INSERT INTO Suppliers (SupplierID, SupplierName, ContactName, Phone, Email, Address) VALUES
(1, 'Tech Supplies Inc.', 'Naruto', '123-456-7890', 'contact@techsupplies.com', '123 Tech St, Tech City'),
(2, 'Home Goods Wholesale', 'Luffy', '321-654-9870', 'info@homegoods.com', '789 Appliance Rd, Home City'),
(3, 'Fashion Suppliers Co.', 'Ichigo', '456-789-1230', 'contact@fashionsuppliers.com', '101 Fashion St, Style City'),
(4, 'Toys Kingdom', 'Goku', '567-890-1234', 'sales@toysk.com', '456 Play Ave, Funville'),
(5, 'Book World', 'Light', '678-901-2345', 'orders@bookworld.com', '789 Novel Ln, Storyville'),
(6, 'Beauty Pro', 'Hinata', '789-012-3456', 'info@beautypro.com', '101 Glam St, Beautyville'),
(7, 'OfficeMart', 'Eren', '890-123-4567', 'contact@officemart.com', '234 Paper Rd, Worktown'),
(8, 'CarParts Co.', 'Mikasa', '901-234-5678', 'support@carparts.com', '567 Auto Ln, Motown'),
(9, 'Sporty Supplies', 'Zoro', '012-345-6789', 'info@sportysupplies.com', '890 Playfield St, Sportstown'),
(10, 'Kitchen Essentials', 'Sanji', '123-456-7891', 'sales@kitchenessentials.com', '345 Cook Rd, Cheftown');

INSERT INTO Customers (CustomerID, CustomerName, ContactName, Phone, Email, Address) VALUES
(1, 'Tharun Reddy', 'Tharun', '123-456-7890', 'tharunreddy@gmail.com', '4941 6th avenue, Niagara Falls'),
(2, 'Sai Kumar', 'Sai', '234-567-8901', 'saikumar@gmail.com', '234 Second St, Niagara Falls'),
(3, 'Mahvi Kouniane', 'Mahvi', '567-890-1234', 'mahvi.kouniane@gmail.com', '567 Third St, St.Catherins'),
(4, 'Kaveri Yellale', 'Kaveri', '890-123-4567', 'kaveri.yella@gmail.com', '890 Fourth St, Toronto'),
(5, 'Lakshmi Prasanna', 'Prasanna', '901-234-5678', 'lakshmi.prasanna@gmail.com', '345 Fifth St, Kingston'),
(6, 'Baanu Vasudhara', 'Baanu', '123-890-4567', 'baanu.vasudhara@gmail.com', '456 Sixth Ave, Ottawa'),
(7, 'Vishnu Uday', 'Vishnu', '789-123-0456', 'vishnu.uday@gmail.com', '567 Seventh St, Vancouver'),
(8, 'Vijay Ch', 'Vijay', '345-678-9012', 'vijay.ch@gmail.com', '678 Eighth Ave, Calgary'),
(9, 'Karunakar Reddy', 'Karunakar', '123-567-8904', 'karunakar.reddy@gmail.com', '789 Ninth St, Montreal'),
(10, 'Dinku Reddy', 'Dinku', '234-789-0123', 'dinku.reddy@gmail.com', '890 Tenth Ave, Edmonton');
```



```
INSERT INTO Orders (OrderID, OrderDate, CustomerID, TotalAmount) VALUES
(1, '2024-11-29', 1, 699.99),
(2, '2024-11-29', 2, 499.99),
(3, '2024-11-30', 3, 89.99),
(4, '2024-12-01', 4, 1299.99),
(5, '2024-12-02', 5, 19.99),
(6, '2024-12-03', 6, 599.99),
(7, '2024-12-04', 7, 14.99),
(8, '2024-12-05', 8, 7.99),
(9, '2024-12-06', 9, 5.99),
(10, '2024-12-07', 10, 899.99);

INSERT INTO OrderDetails (OrderDetailID, OrderID, ProductID, Quantity, UnitPrice) VALUES
(1, 1, 101, 1, 699.99),
(2, 2, 102, 1, 499.99),
(3, 3, 104, 1, 89.99),
(4, 4, 103, 1, 1299.99),
(5, 5, 107, 1, 19.99),
(6, 6, 106, 1, 599.99),
(7, 7, 108, 1, 14.99),
(8, 8, 109, 1, 7.99),
(9, 9, 110, 1, 5.99),
(10, 10, 105, 1, 899.99);
```

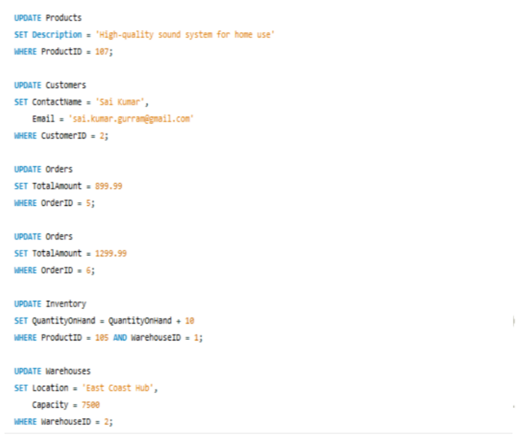
```
INSERT INTO Shipments (ShipmentID, OrderID, ShipmentDate, ShippingAddress, TrackingNumber) VALUES
(1, 1, '2024-11-30', '4941 6th avenue, Niagara Falls', 'TRACK123'),
(2, 2, '2024-11-30', '234 Second St, Niagara Falls', 'TRACK234'),
(3, 3, '2024-11-30', '567 Third St, St.Catherins', 'TRACK345'),
(4, 4, '2024-12-01', '890 Fourth St, Toronto', 'TRACK456'),
(5, 5, '2024-12-02', '345 Fifth St, Kingston', 'TRACK567'),
(6, 6, '2024-12-03', '456 Sixth Ave, Ottawa', 'TRACK678'),
(7, 7, '2024-12-04', '567 Seventh St, Vancouver', 'TRACK789'),
(8, 8, '2024-12-05', '678 Eighth Ave, Calgary', 'TRACK890'),
(9, 9, '2024-12-06', '789 Ninth St, Montreal', 'TRACK901'),
(10, 10, '2024-12-07', '890 Tenth Ave, Edmonton', 'TRACK012');

INSERT INTO Warehouses (WarehouseID, WarehouseName, Location, Capacity) VALUES
(1, 'Main Warehouse', 'Warehouse District', 5000),
(2, 'East Coast Warehouse', 'East Coast Logistics', 7000),
(3, 'West Coast Warehouse', 'West Coast Logistics', 6000),
(4, 'Central Warehouse', 'Central Industrial Zone', 8000),
(5, 'North Warehouse', 'Northern Hub', 4500),
(6, 'South Warehouse', 'Southern Logistics Park', 4000),
(7, 'Overseas Warehouse', 'International Trade Center', 3000),
(8, 'Suburban Warehouse', 'Suburban Area', 3500),
(9, 'Urban Warehouse', 'City Center', 5500),
(10, 'Remote Warehouse', 'Mountain Logistics Hub', 2500);
```

```
INSERT INTO Inventory (InventoryID, ProductID, WarehouseID, QuantityOnHand) VALUES
(1, 101, 1, 50),
(2, 102, 2, 30),
(3, 103, 3, 15),
(4, 104, 4, 45),
(5, 105, 5, 20),
(6, 106, 6, 10),
(7, 101, 7, 25),
(8, 102, 8, 35),
(9, 103, 9, 5),
(10, 104, 10, 12);

INSERT INTO Payments (PaymentID, OrderID, PaymentDate, Amount, PaymentMethod) VALUES
(1, 1, '2024-11-29', 699.99, 'Credit Card'),
(2, 2, '2024-11-29', 499.99, 'PayPal'),
(3, 3, '2024-11-30', 89.99, 'Debit Card'),
(4, 4, '2024-12-01', 1299.99, 'Credit Card'),
(5, 5, '2024-12-02', 19.99, 'Gift Card'),
(6, 6, '2024-12-03', 599.99, 'PayPal'),
(7, 7, '2024-12-04', 14.99, 'Cash'),
(8, 8, '2024-12-05', 7.99, 'Debit Card'),
(9, 9, '2024-12-06', 5.99, 'Credit Card'),
(10, 10, '2024-12-07', 899.99, 'Bank Transfer');
```

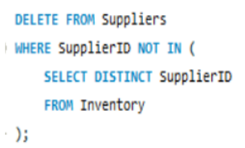
DML COMMANDS FOR UPDATE AND DELETE THE TABLES

/* update command for DML*/

/* product*/



```
UPDATE Products
SET Description = 'High-quality sound system for home use'
WHERE ProductID = 107;

UPDATE Customers
SET ContactName = 'Sai Kumar',
    Email = 'sai.kumar.gurram@gmail.com'
WHERE CustomerID = 2;

UPDATE Orders
SET TotalAmount = 899.99
WHERE OrderID = 5;

UPDATE Orders
SET TotalAmount = 1299.99
WHERE OrderID = 6;

UPDATE Inventory
SET QuantityOnHand = QuantityOnHand + 10
WHERE ProductID = 105 AND WarehouseID = 1;

UPDATE Warehouses
SET Location = 'East Coast Hub',
    Capacity = 7500
WHERE WarehouseID = 2;
```



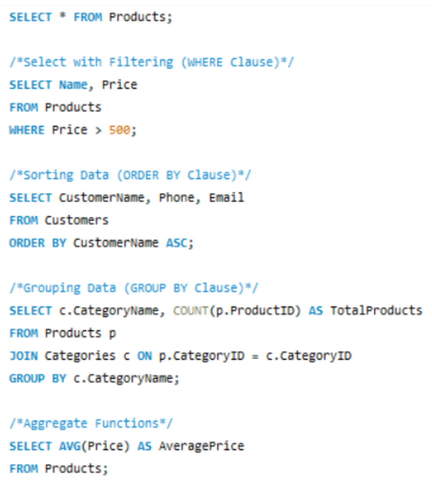
```
DELETE FROM Suppliers
WHERE SupplierID NOT IN (
    SELECT DISTINCT SupplierID
    FROM Inventory
);
```

DQL COMMANDS:

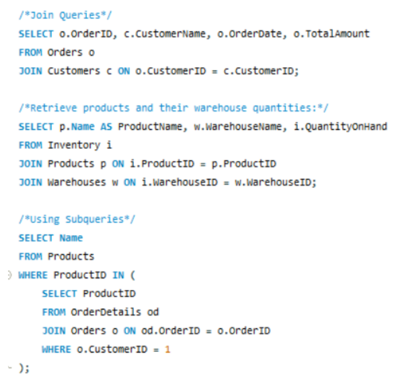
- DQL (Data Query Language) is a subset of SQL that is used to query and retrieve data from a database. It focuses on SELECT statements to extract data from one or more tables in a database. DQL does not modify data, but it helps in querying and analyzing data.

➨ Crafting SQL Queries to Retrieve Data:

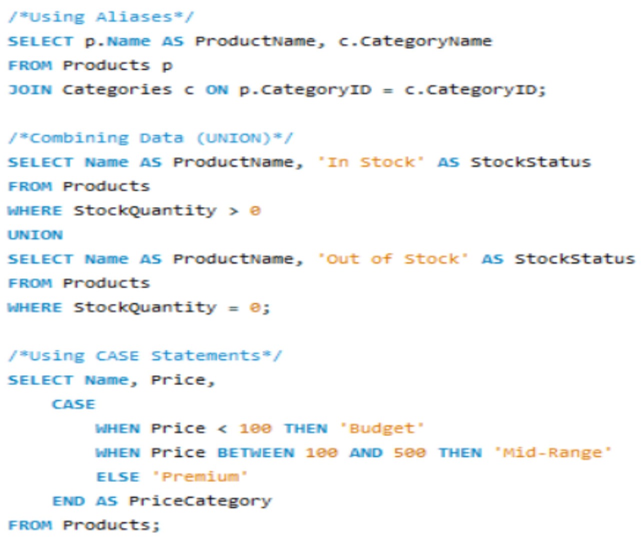
- Selecting Specific Records: Use SELECT statements to retrieve data from one or more tables.
- Joining Multiple Tables: Use JOIN operations to combine data from related tables.
- Grouping Data: Use GROUP BY clauses to organize data into groups for aggregation.
- Performing Aggregate Calculations: Use functions like COUNT, SUM, AVG, MAX, and MIN to perform calculations on grouped data.



```sql
SELECT * FROM Products;

/*Select with Filtering (WHERE Clause)*/
SELECT Name, Price
FROM Products
WHERE Price > 500;

/*Sorting Data (ORDER BY Clause)*/
SELECT CustomerName, Phone, Email
FROM Customers
ORDER BY CustomerName ASC;

/*Grouping Data (GROUP BY Clause)*/
SELECT c.CategoryName, COUNT(p.ProductID) AS TotalProducts
FROM Products p
JOIN Categories c ON p.CategoryID = c.CategoryID
GROUP BY c.CategoryName;

/*Aggregate Functions*/
SELECT AVG(Price) AS AveragePrice
FROM Products;
```

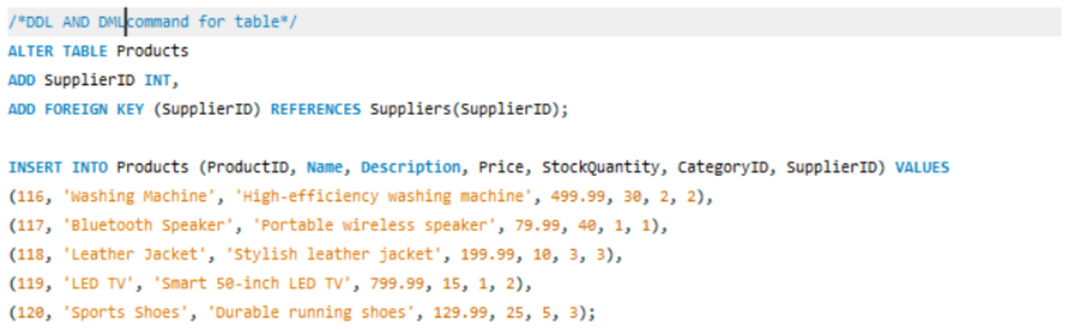


```sql
/*Join Queries*/
SELECT o.OrderID, c.CustomerName, o.OrderDate, o.TotalAmount
FROM Orders o
JOIN Customers c ON o.CustomerID = c.CustomerID;

/*Retrieve products and their warehouse quantities:*/
SELECT p.Name AS ProductName, w.WarehouseName, i.QuantityOnHand
FROM Inventory i
JOIN Products p ON i.ProductID = p.ProductID
JOIN Warehouses w ON i.WarehouseID = w.WarehouseID;

/*Using Subqueries*/
SELECT Name
FROM Products
WHERE ProductID IN (
    SELECT ProductID
    FROM OrderDetails od
    JOIN Orders o ON od.OrderID = o.OrderID
    WHERE o.CustomerID = 1
);
```



```sql
/*Using Aliases*/
SELECT p.Name AS ProductName, c.CategoryName
FROM Products p
JOIN Categories c ON p.CategoryID = c.CategoryID;

/*Combining Data (UNION)*/
SELECT Name AS ProductName, 'In Stock' AS StockStatus
FROM Products
WHERE StockQuantity > 0
UNION
SELECT Name AS ProductName, 'Out of Stock' AS StockStatus
FROM Products
WHERE StockQuantity = 0;

/*Using CASE Statements*/
SELECT Name, Price,
    CASE
        WHEN Price < 100 THEN 'Budget'
        WHEN Price BETWEEN 100 AND 500 THEN 'Mid-Range'
        ELSE 'Premium'
    END AS PriceCategory
FROM Products;
```

USING DDL AND DML COMMAND TO ALTER THE TABLE



```
/*DDL AND DML command for table*/
ALTER TABLE Products
ADD SupplierID INT,
ADD FOREIGN KEY (SupplierID) REFERENCES Suppliers(SupplierID);

INSERT INTO Products (ProductID, Name, Description, Price, StockQuantity, CategoryID, SupplierID) VALUES
(116, 'Washing Machine', 'High-efficiency washing machine', 499.99, 30, 2, 2),
(117, 'Bluetooth Speaker', 'Portable wireless speaker', 79.99, 40, 1, 1),
(118, 'Leather Jacket', 'Stylish leather jacket', 199.99, 10, 3, 3),
(119, 'LED TV', 'Smart 50-inch LED TV', 799.99, 15, 1, 2),
(120, 'Sports Shoes', 'Durable running shoes', 129.99, 25, 5, 3);
```

CONCLUSION

The Global Supply Chain Management System project successfully demonstrates the power and flexibility of SQL in managing complex supply chain operations. By carefully designing and implementing a structured database schema, creating comprehensive tables, and ensuring data integrity through key constraints, we have built a robust foundation for efficient supply chain management. The realistic data inserted into the system provides a valuable simulation of real-world operations, allowing for detailed analysis and insights into various aspects of the supply chain, such as supplier performance, inventory levels, and order fulfillment.

The well-defined relationships between entities ensure consistency and enable seamless data retrieval and analysis. This project highlights the importance of a well-designed database in streamlining supply chain processes and enhancing overall efficiency. The Global Supply Chain Management System is now equipped to support data-driven decision-making, improve operational performance, and adapt to the evolving needs of the business environment.

In conclusion, this project serves as a testament to the capabilities of SQL in creating effective and scalable solutions for supply chain management. By leveraging the strengths of structured data management, the system is poised to contribute significantly to the success and growth of any business operating within the global supply chain. The comprehensive dataset and structured approach ensure that the database design is efficient and capable of handling the complexities of modern supply chain operations.