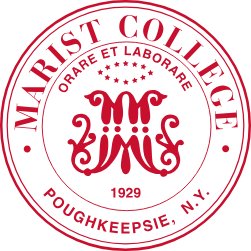
**Warehouse Management System – Phase 6.1**

**Database Management Systems**

**MSCS 542L-256**

**Embrace Database**



Marist College

School of Computer Science and Mathematics

Submitted To:

Dr. Reza Sadeghi

11/15/2023

**Project Report of Warehouse Management System – Phase 6.1**

**Team Name**

Embrace Database

**Team Members**

1. Reem Ooka (Team Head) [Reem.Ooka1@marist.edu](mailto:Reem.Ooka1@marist.edu)

2. Frank Seelmann (Team Member) [Frank.Seelmann1@marist.edu](mailto:frank.seelmann1@marist.edu)

**Description of Team Members**

1. Reem Ooka

Hailing from the Incredible country of India, I proudly call Mumbai my home - a vibrant metropolitan city renowned as the economic and entertainment heartbeat of our nation. I graduated in May 2022 with a bachelor's degree in computer science. During my professional career, I gained expertise in developing websites tailored for citizen-centric solutions while working at an e-governance company as a Full Stack Developer. Presently, I am furthering my academic pursuits as a graduate student at Marist, where I am specializing in Cloud Computing.

Frank voted Reem to be the team lead since he already has a job, so putting “Team Lead” on his resume will not be as useful for him.

2. Frank Seelmann

Hudson Valley native, graduated from SUNY New Paltz in May 2022 with a Bachelor of Computer Engineering. Works at IBM in Poughkeepsie as a Verification Engineer. Has some experience using database management systems, specifically MySQL and MongoDB, but in both cases the DBMS was supplemental to the project, not its focus.

**Table of Contents**

[1. Project Objective 4](#_Toc364801042)

[2. Review the Related Work 5](#_Toc542432329)

[3. The Merits of Our Project 6](#_Toc649099280)

[4. GitHub Repository 6](#_Toc1279003252)

[5. External Models 6](#_Toc648381143)

[6. Entity Relationship Model 7](#_Toc49795513)

[6.a ER Diagram 7](#_Toc1274814176)

[Figure 6.a - Entity Relationship Diagram 8](#_Toc1297428313)

[6.b ER Model 9](#_Toc215456758)

[7. Enhanced Entity Relationship Model 10](#_Toc275648397)

[Figure 6 – Enhanced Entity Relationship Model 11](#_Toc1722903521)

[8. Database Development 11](#_Toc450588709)

[Table 8.1 – Data Insertion 12](#_Toc553918875)

[9. Loading Data 14](#_Toc38041626)

[Table 9.1 – Data Insertion 15](#_Toc328949567)

[Works Cited 18](#_Toc995301015)

[Appendix 19](#_Toc1037859550)

[SQL to Generate and Insert into Tables 20](#_Toc805319218)

**Table of Figures**

[1. Project Objective 4](#_Toc805014347)

[2. Review the Related Work 5](#_Toc215042955)

[3. The Merits of Our Project 6](#_Toc941531926)

[4. GitHub Repository 6](#_Toc360308193)

[5. Entity Relationship Model 6](#_Toc120291976)

[5.a ER Diagram 6](#_Toc338795510)

[Figure 5.a - Entity Relationship Diagram 7](#_Toc1365113691)

[5.b ER Model 8](#_Toc491734337)

[Figure 5.b - Entity Relationship Model 9](#_Toc430587628)

[6. Enhanced Entity Relationship Model 9](#_Toc835529983)

[Figure 6 – Enhanced Entity Relationship Model 10](#_Toc411824058)

[Works Cited 10](#_Toc1662178320)

# 1. Project Objective

The core objective of implementing the General Warehouse Management System (WMS) is to effectively oversee and streamline product information management. This encompasses attributes such as product name, identification, storage specifications, pricing, weight, height, and other relevant data points. The WMS aims to enable users to efficiently search for items and initiate requests for either borrowing or purchasing. It strives to achieve accurate product tracking, foster seamless user interactions, offer a user-friendly interface for inventory management and transactions, all while upholding data integrity and security.

For this class, our WMS has four minimum criteria:

1. An admin user who has several exclusive capabilities, such as adding a guest user, removing users, adding/deleting/editing items from the warehouse, and accepting/rejecting borrowing requests.
2. Users must be able search the warehouse, request to borrow/buy items for a specific time and view the history of borrowed items.
3. The WMS should have several user-friendly features, such as a welcome page, a menu of all functions available to a user, tabularized reports, an exit function, and providing warning/error messages.
4. The WMS should protect user information, particularly passwords.

The WMS will use MySQL as the DBMS, and Java to handle the front-end.

# 2. Review the Related Work

Our project is a prototype of a real-world WMS. To better understand what is expected of WMS, and what might make one better than another, we examined other existing solutions.

Using Capterra [1], a website for comparing software solutions, we were able to compare several WMS providers and see some of the key features that set them apart. Capterra has user ratings for 10 popular WMS providers, as well as listing seven key features for comparison. These features are Barcoding/RFID, Forecasting, Inventory Management, Purchasing & Receiving, Receiving/Put away Management, Returns Management, and Shipping Management.

The WMS with the most reviews, 1335, is NetSuite from Oracle, which has all 7 features. This product is also very expensive, with a pricing range of $1,299 to $9,999 a month, depending on the subscription package [2]. Users critique it for its high cost and complexity.

The second most reviewed WMS, with 300 reviews, is “Inventory Cloud.” Users praise its short learning curve but criticize its lack of customization.

“Carton Cloud” has the highest rating at 4.8 stars. It lacks 2 of the 7 features, Forecasting and Returns Shipment. Users praise it for its ease of use and affordability (pricing starts at $99/week).

# 3. The Merits of Our Project

The best reason to choose our WMS is that we offer a cost-effective solution. Having a small team (2 developers) means we can keep costs low. We expect to provide 5 of the 7 features considered by Capterra: Inventory Management, Purchasing & Receiving, Receiving/Put away Management, Returns Management, and Shipping Management.

From reading the reviews, a “nice-to-have" for our WMS would be adding customizability for the user. This could take the form of having the reports show up in different ways, such as list view vs grid view, which would be remembered by the systems as the preferred viewing option for the user.

# 4. GitHub Repository

To provide version control for our project, we are using GitHub:

<https://github.com/ReemPatel13/MSCS-542L-256_WarehouseManagementSystem_EmbraceDatabase>

# 5. External Models

To how our database will be used, we created two tables to show some of the actions they can take. One is for the employees, and one is for the customers.

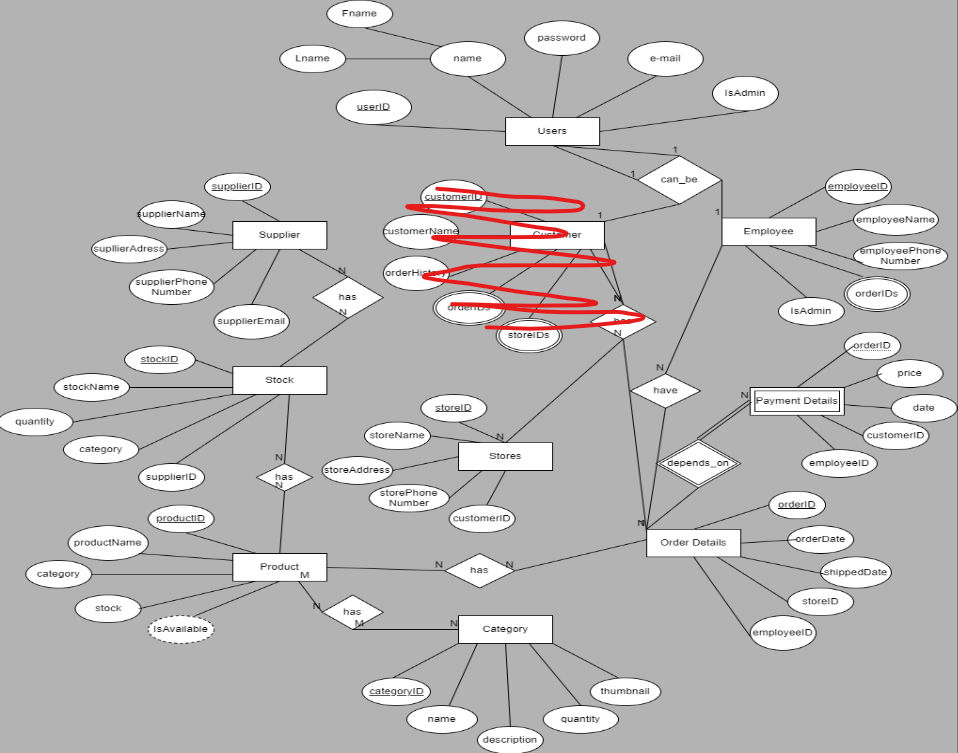


Figure 5.a - Employee External Model

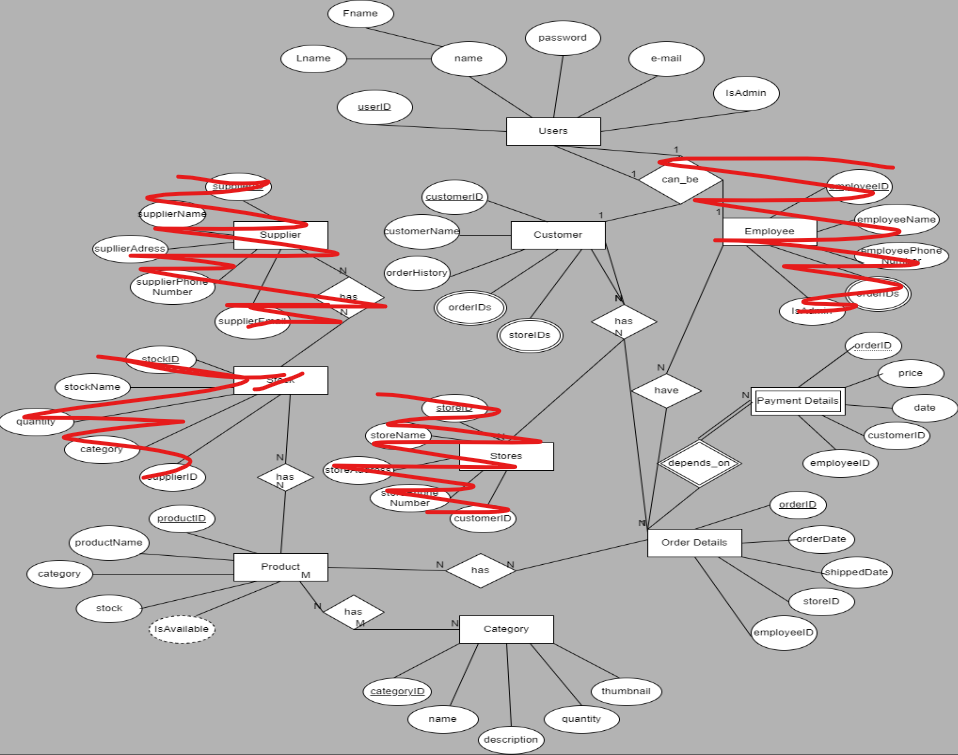


Figure 5.b - Customer External Model

# 6. Entity Relationship Model

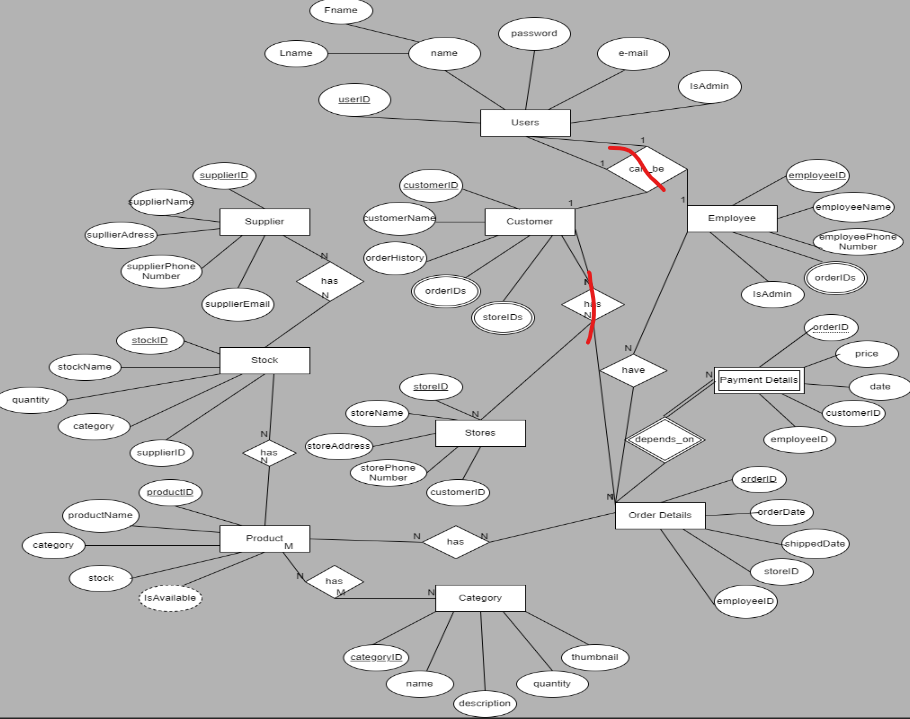
To illustrate how our database will be constructed, we created an Entity Relationship Diagram and an Entity Relationship Model.

## 6.a ER Diagram

To create this mini world, we used [draw.io](http://draw.io) and the entities, attributes, the relationships between them, the participations and cardinality was mutually discussed adhering to the pointers given.

The following is a short description about each entity and its attributes:

1. Users: Our Warehouse Management System will have users who can be either employees or customers. An employee could also be an admin. It has a Primary key called userID, a composite attribute such as name composing of Fname and Lname, other attributes like password and e-mail as well as a boolean attribute called IsAdmin. The user entity has 1 to 1 relationship with the employee entity and customer entity.
2. Employee: An employee is responsible for handling orders and could also be the admin. Here, the primary key is employeeID. It has attributes like employeeName, employeePhoneNumber, a multivalued attribute like the orderID and a boolean attribute like the IsAdmin. The employee entity has 1 to 1 relationship with the user entity and 1 to Many relationship with the order details entity.
3. Customer: A customer is basically the store owner who is our final client. The customer entity has a Primary key called customerID. Other attributes include customerName and orderHistory. Two multivalued attributes here are orderID and storeID as a customer can have multiple orders and stores. This entity has 1 to 1 relationship with the user entity. It has 1 to Many relationship with the store entity. It has 1 to Many relationship with the orderDetails entity.
4. Supplier: A supplier supplies products to the warehouse. It has a supplierID, supplierName, supplierAddress, supplierPhoneNumber and supplierEmail. It has 1 to many relationship with the stock entity.
5. Stock: The stock keeps a records of all available products. It has a stockID, stockName, quantity, category and a supplierID. It has 1 to Many relationship with the product entity.
6. Product: The product entity has a productID, productName, category, stock and a derived attribute called IsAvailable. It has Many to Many relationship with the category entity.
7. Category: A category will have a categoryID, name, description, quantity and thumbnail. It has a Many to Many relationship with the product entity.
8. Stores: The stores entity has a storeID, storeName, storeAddress, storePhoneNumber and customerID. It has a Many to 1 relationship with the customer entity.
9. OrderDetails: This entity has attributes like orderID, orderDate, shippedDate, storeID and employeeID. It has Many to 1 relationships with the employee and customer entities. It also has a 1 to 1 relationship with the weak payment entity.
10. WEAK PaymentDetails: This is a weak entity which does not have its own primary key and depends on the primary key of the orderDetails entity taking its orderID. It has attributes like price, date, customerID and employeeID.



### Figure 6.a - Entity Relationship Diagram

## 6.b ER Model

The Entity Relationship model was created using [draw.io](https://app.diagrams.net/#G1aTKjF4hdOrcInVLw2XpZXm5cGy09vSpO), as it is a free online CAD tool with the necessary symbols for creating an ER model. Figure 5.b shows the ER model.

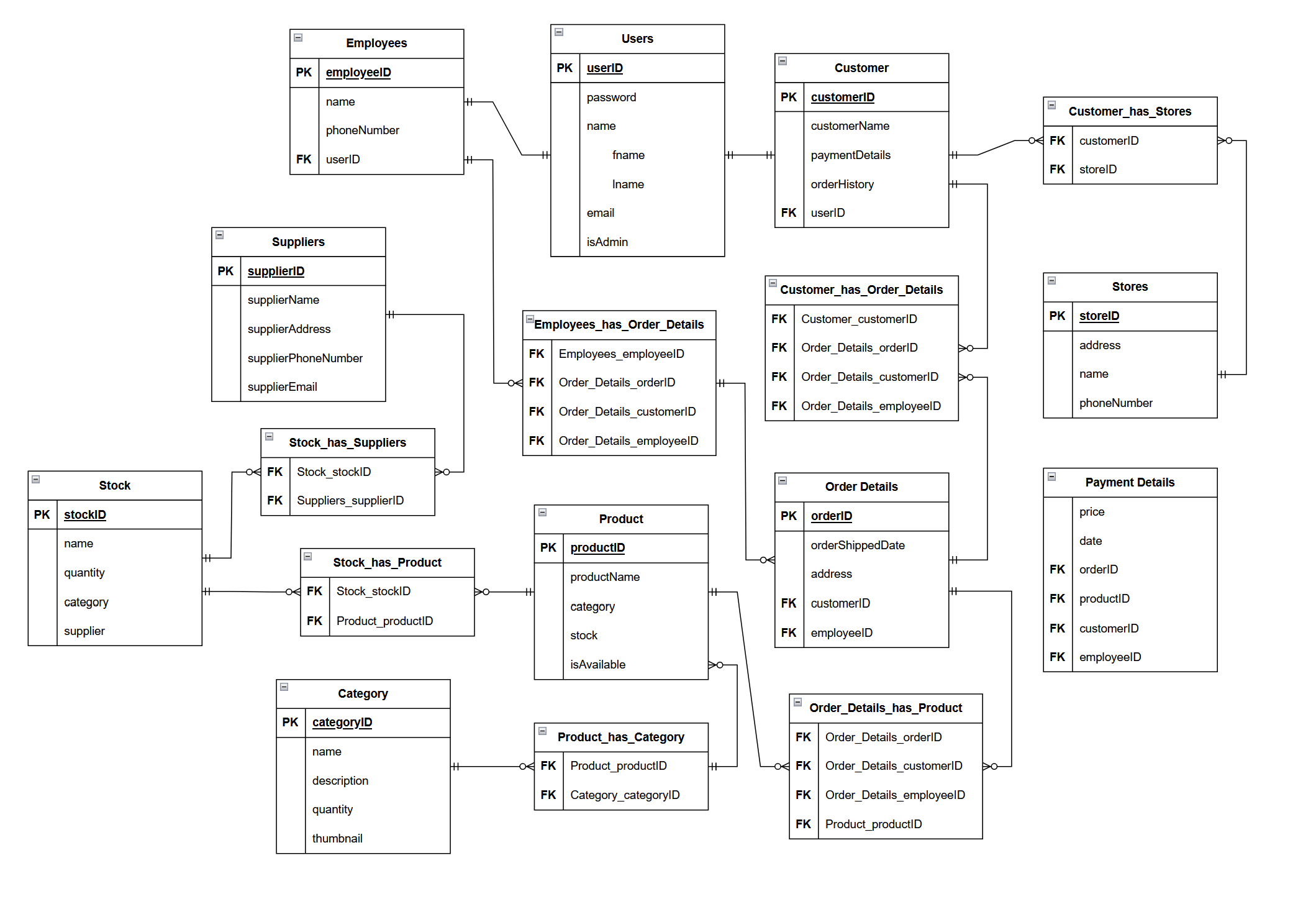
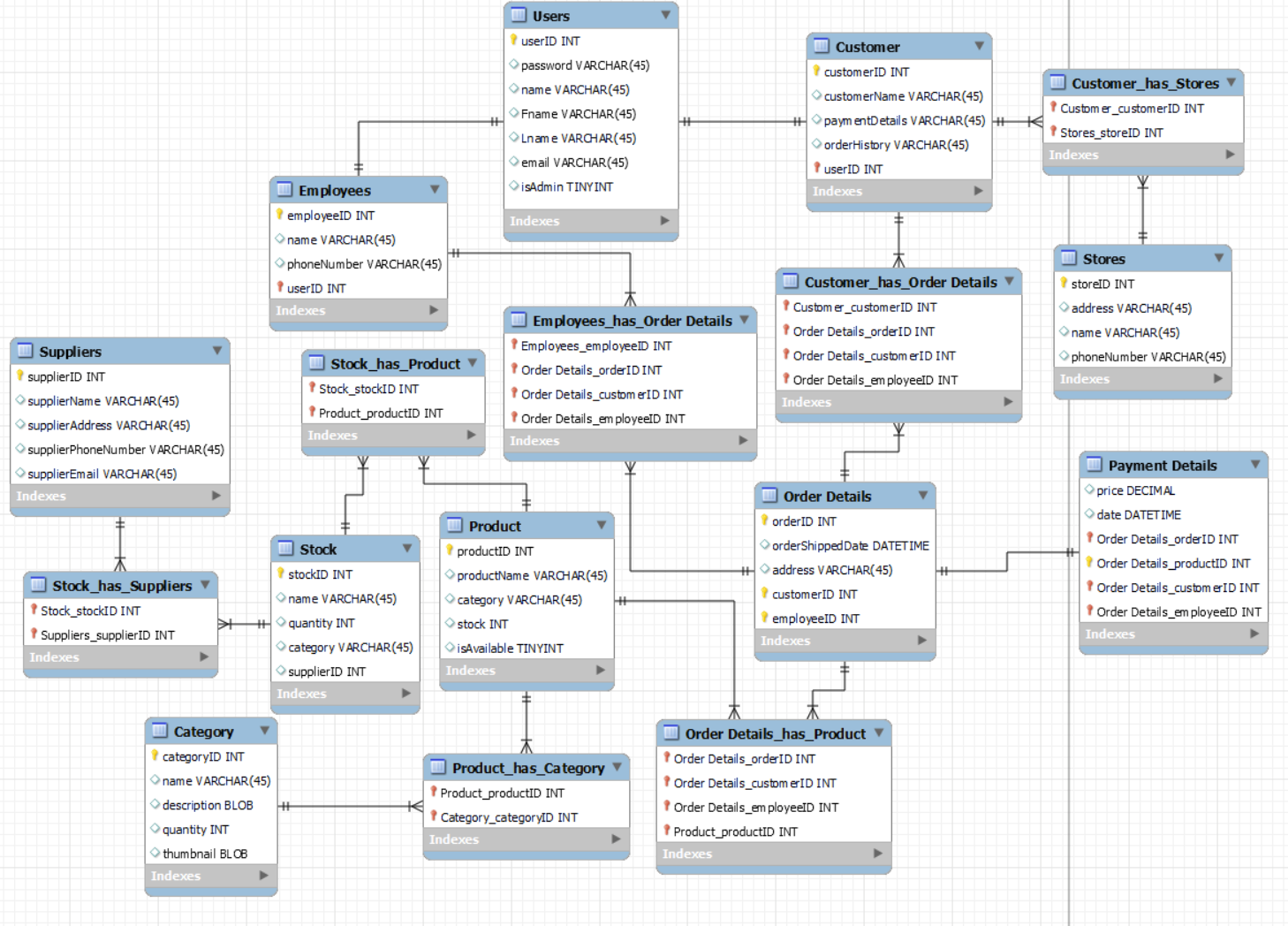


Figure 6.b - Entity Relationship Model

# 7. Enhanced Entity Relationship Model

For this project we are using the MySQL Workbench. One feature of the Workbench is it allows the creation of Enhanced Entity Relationship Models (EER) models. Figure 6 shows the results of using the tool to illustrate our entity, attribute, and relationship information.



### Figure 6 – Enhanced Entity Relationship Model

Keys in a relational database are attributes (or sets of attributes) are used to uniquely identify records in a table. A primary key is used to uniquely identify a record in the table itself, while a foreign key is used to establish a relationship with the primary key of another table. One nice feature of the EER tool is that it will automatically add foreign keys when you create relationships. These are denoted by the orange pin-like symbol to the left of an attribute name. One example is the userID foreign key in the Employee entity.

Relationships in a relational database refer to associations between tables based on keys. Relationships can be one-to-one, one-to-many, and many-to-many. Examples of these relationships would be Order Details to Payment, Customer to Order Details, and Product to Category, respectively. As a second useful feature, the tool will automatically create an associate entity when appropriate, such as when defining a relationship as many-to-many. The Product to Category relationship is an example of many-to-many since a product could have multiple categories, and a category may have multiple products.

# 8. Database Development

To create our database, we first used the SQL commands to create and use it.

CREATE SCHEMA IF NOT EXISTS warehouse\_management;

USE warehouse\_management;

Then, we created the tables using CREATE TABLE IF NOT EXISTS <Table\_Name>. The code in its entirety can be found in the Appendix, but the code for creating these tables is shown below in Table 8.1, divided up by statement.

### Table 8.1 – Data Insertion

|  |  |  |
| --- | --- | --- |
| Table | SQL | Description |
| Users | CREATE TABLE IF NOT EXISTS Users (  userID INT PRIMARY KEY AUTO\_INCREMENT,  password VARCHAR(45) NULL,  name VARCHAR(50) NULL,  Fname VARCHAR(25) NULL,  Lname VARCHAR(25) NULL,  email VARCHAR(45) NULL,  isAdmin TINYINT NULL  ); | The Users table has attributes like userID with is an integer Primary Key and will auto-increment. It has a password with the datatype of variable character and size 45, name with size 50, composite attributes like Fname and Lname, email as well as a Boolean attribute called isAdmin. |
| Customers | CREATE TABLE IF NOT EXISTS Customer (  customerID INT AUTO\_INCREMENT,  customerName VARCHAR(45) NULL,  paymentDetails VARCHAR(45) NULL,  orderHistory VARCHAR(45) NULL,  orderDetails VARCHAR(45) NULL,  userID INT NOT NULL,  PRIMARY KEY (customerID, userID),  INDEX fk\_Customer\_Users1\_idx (userID ASC),  CONSTRAINT fk\_Customer\_Users1  FOREIGN KEY (userID)  REFERENCES Users (userID)  ); | The Customer table has customerID which is an integer Primary key will be incremented automatically. It has a customerName which is a string basically varchar with size 45, paymentDetails attribute with varchar and size 45, orderHistory and orderDeatils . t also has an integer userID which is a foreign key referring to the userID in the users’ table. |
| Product | CREATE TABLE IF NOT EXISTS Product (  productID INT PRIMARY KEY,  productName VARCHAR(45) NULL,  category VARCHAR(45) NULL,  stock INT NULL,  isAvailable TINYINT NULL  ); | The table of Product has productid as an integer Primary key, a productName with variable character data type and size 45, category, stock with data type as an integer and a Boolean value called isAvailable. |
| Employees | CREATE TABLE IF NOT EXISTS Employees (  employeeID INT NOT NULL,  name VARCHAR(45) NULL,  orderID VARCHAR(45) NULL,  phoneNumber VARCHAR(45) NULL,  userID INT NOT NULL,  orderIDs INT NULL,  PRIMARY KEY (employeeID, userID),  INDEX fk\_Employees\_Users\_idx (userID ASC),  CONSTRAINT fk\_Employees\_Users  FOREIGN KEY (userID)  REFERENCES Users (userID)  ); | The table of Employees consists of employeeID which again is an integer Primary key, name with data type varchar, orderID with data type varchar, phoneNumber which is an integer, userID which is a foreign key. |
| Order\_Details | CREATE TABLE IF NOT EXISTS Order\_Details (  orderID INT NOT NULL,  orderShippedDate DATETIME NULL,  address VARCHAR(45) NULL,  customerID INT NOT NULL,  employeeID INT NOT NULL,  PRIMARY KEY (orderID, customerID, employeeID),  INDEX fk\_Order\_Details\_Customer1\_idx (customerID ASC),  INDEX fk\_Order\_Details\_Employees1\_idx (employeeID ASC),  CONSTRAINT fk\_Order\_Details\_Customer1  FOREIGN KEY (customerID)  REFERENCES Customer (customerID),  CONSTRAINT fk\_Order\_Details\_Employees1  FOREIGN KEY (employeeID)  REFERENCES Employees (employeeID)  ); | The table of Order\_Details has an orderID which is a integer Primary key, orderShippedDate with the data type of Datetime, address which is a varchar, customerID and employeeID which are integers and foreign keys. |
| Category | CREATE TABLE IF NOT EXISTS Category (  categoryID INT PRIMARY KEY AUTO\_INCREMENT,  name VARCHAR(45) NULL,  description BLOB NULL,  quantity INT NULL,  thumbnail BLOB NULL  ); | The table Category contains a categoryID which is an integer Primary key, name which is a varchar, description which is of data type BLOB, quantity with data type int, and a thumbnail with data type BLOB. |
| Suppliers | CREATE TABLE IF NOT EXISTS Suppliers (  supplierID INT PRIMARY KEY,  supplierName VARCHAR(45) NULL,  supplierAddress VARCHAR(45) NULL,  supplierPhoneNumber VARCHAR(45) NULL,  supplierEmail VARCHAR(45) NULL  ); | The table of Suppliers consist of an integer Primary key called supplierID, supplierName with data type varchar, supplierAddress, supplierPhoneNumber and a supplierEmail. |
| Stores | CREATE TABLE IF NOT EXISTS Stores (  storeID INT NOT NULL AUTO\_INCREMENT,  address VARCHAR(45) NULL,  name VARCHAR(45) NULL,  phoneNumber VARCHAR(45) NULL,  customerID INT NOT NULL,  PRIMARY KEY (storeID, customerID),  INDEX fk\_Stores\_Customer1\_idx (customerID ASC),  CONSTRAINT fk\_Stores\_Customer1  FOREIGN KEY (customerID)  REFERENCES Customer (customerID)  ); | The table of Stores contains storeID which is an integer, address with data type varchar, name, phoneNumber, and customerID which is an integer foreign key. |
| Stock | CREATE TABLE IF NOT EXISTS Stock (  stockID INT PRIMARY KEY,  name VARCHAR(45) NULL,  quantity INT NULL,  category VARCHAR(45) NULL,  supplierID INT NULL  ); | The table Stock has stockID as integer Primary key, name which is varchar, integer quantity, category as varchar, supplierID as int. |
| Payment\_Details | CREATE TABLE IF NOT EXISTS Payment\_Details (  price DECIMAL NULL,  date DATETIME NULL,  Order\_Details\_orderID INT NOT NULL,  Order\_Details\_productID INT NOT NULL,  Order\_Details\_customerID INT NOT NULL,  Order\_Details\_employeeID INT NOT NULL,  PRIMARY KEY (Order\_Details\_orderID, Order\_Details\_productID, Order\_Details\_customerID,  Order\_Details\_employeeID),  INDEX fk\_Payment\_Order\_Details1\_idx (Order\_Details\_orderID ASC, Order\_Details\_productID ASC,  Order\_Details\_customerID ASC, Order\_Details\_employeeID ASC),  CONSTRAINT fk\_Payment\_Order\_Details1  FOREIGN KEY (Order\_Details\_orderID , Order\_Details\_customerID , Order\_Details\_employeeID)  REFERENCES Order\_Details (orderID , customerID , employeeID)  ); | The table Payment\_Details has price with data type decimal, date with data type as Datetime, order\_details\_orderID as int, productID, customer\_ID, employee\_ID. |
| Product\_has\_Category | CREATE TABLE IF NOT EXISTS Product\_has\_Category (  Product\_productID INT NOT NULL,  Category\_categoryID INT NOT NULL,  PRIMARY KEY (Product\_productID, Category\_categoryID),  INDEX fk\_Product\_has\_Category\_Category1\_idx (Category\_categoryID ASC),  INDEX fk\_Product\_has\_Category\_Product1\_idx (Product\_productID ASC),  CONSTRAINT fk\_Product\_has\_Category\_Product1  FOREIGN KEY (Product\_productID)  REFERENCES Product (productID),  CONSTRAINT fk\_Product\_has\_Category\_Category1  FOREIGN KEY (Category\_categoryID)  REFERENCES Category (categoryID)  ); | The table Product\_has\_Category has attributes like Product\_productID and Category\_categoryID which are integer values and primary keys as well as foreign constraints. |
| Stock\_has\_Product | CREATE TABLE IF NOT EXISTS Stock\_has\_Product (  Stock\_stockID INT NOT NULL,  Product\_productID INT NOT NULL,  PRIMARY KEY (Stock\_stockID, Product\_productID),  INDEX fk\_Stock\_has\_Product\_Product1\_idx (Product\_productID ASC),  INDEX fk\_Stock\_has\_Product\_Stock1\_idx (Stock\_stockID ASC),  CONSTRAINT fk\_Stock\_has\_Product\_Stock1  FOREIGN KEY (Stock\_stockID)  REFERENCES Stock (stockID),  CONSTRAINT fk\_Stock\_has\_Product\_Product1  FOREIGN KEY (Product\_productID)  REFERENCES Product (productID)  ); | The table Stock\_has\_Product has attributes like Product\_productID and Stock\_stockID which are integer values and primary keys as well as foreign constraints. |
| Stock\_has\_Suppliers | CREATE TABLE IF NOT EXISTS Stock\_has\_Suppliers (  Stock\_stockID INT NOT NULL,  Suppliers\_supplierID INT NOT NULL,  PRIMARY KEY (Stock\_stockID, Suppliers\_supplierID),  INDEX fk\_Stock\_has\_Suppliers\_Suppliers1\_idx (Suppliers\_supplierID ASC),  INDEX fk\_Stock\_has\_Suppliers\_Stock1\_idx (Stock\_stockID ASC),  CONSTRAINT fk\_Stock\_has\_Suppliers\_Stock1  FOREIGN KEY (Stock\_stockID)  REFERENCES Stock (stockID),  CONSTRAINT fk\_Stock\_has\_Suppliers\_Suppliers1  FOREIGN KEY (Suppliers\_supplierID)  REFERENCES Suppliers (supplierID)  ); | The table Stock\_has\_Suppliers has attributes like Stock\_stockID and Suppliers\_supplierID which are integer values and primary keys as well as foreign constraints. |
| Order\_Details\_has\_Product | CREATE TABLE IF NOT EXISTS Order\_Details\_has\_Product (  Order\_Details\_orderID INT NOT NULL,  Order\_Details\_customerID INT NOT NULL,  Order\_Details\_employeeID INT NOT NULL,  Product\_productID INT NOT NULL,  PRIMARY KEY (Order\_Details\_orderID, Order\_Details\_customerID, Order\_Details\_employeeID,  Product\_productID),  INDEX fk\_Order\_Details\_has\_Product\_Product1\_idx (Product\_productID ASC),  INDEX fk\_Order\_Details\_has\_Product\_Order\_Details1\_idx (Order\_Details\_orderID ASC,  Order\_Details\_customerID ASC, Order\_Details\_employeeID ASC),  CONSTRAINT fk\_Order\_Details\_has\_Product\_Order\_Details1  FOREIGN KEY (Order\_Details\_orderID , Order\_Details\_customerID , Order\_Details\_employeeID)  REFERENCES Order\_Details (orderID , customerID , employeeID),  CONSTRAINT fk\_Order\_Details\_has\_Product\_Product1  FOREIGN KEY (Product\_productID)  REFERENCES Product (productID)  ); | The table Order\_Details\_has\_Product has attributes like Order\_Details\_orderID, Order\_Details\_customerID, Order\_Details\_employeeID and Product\_productID which are integer values and primary keys as well as foreign constraints. |

# 9. Loading Data

When initially inserting sample data into our database, we ran into the foreign key constraint for some tables. To get around this, we used the MySQL setting to temporarily disable the foreign key constraint. This is done with the following SQL command:

SET FOREIGN\_KEY\_CHECKS=1;

To speed up insertion, we made one insert statement per table, inserting all of the sample data at once. The code in its entirety can be found in the Appendix, but the following Table 9.1 shows the SQL for insertion per-table. We attempted to use the “explain” keyword to see how we could further optimize our insertion in the MySQL Workbench, but the ‘Execution Plan’ was always blank, and query status showed extremely low execution times.

### Table 9.1 – Data Insertion

|  |  |
| --- | --- |
| Table | SQL |
| Users | INSERT INTO Users (password, name, Fname, Lname, email, isAdmin) VALUES  ('password123', 'JohnDoe', 'John', 'Doe', 'john.doe@example.com', 0),  ('password456', 'JaneSmith', 'Jane', 'Smith', 'jane.smith@example.com', 0),  ('password789', 'MichaelBrown', 'Michael', 'Brown', 'michael.brown@example.com', 1),  ('password101', 'EmilyJohnson', 'Emily', 'Johnson', 'emily.johnson@example.com', 0),  ('password202', 'DavidWilson', 'David', 'Wilson', 'david.wilson@example.com', 0),  ('password303', 'SarahLee', 'Sarah', 'Lee', 'sarah.lee@example.com', 1),  ('password404', 'RobertTaylor', 'Robert', 'Taylor', 'robert.taylor@example.com', 0),  ('password505', 'SusanAnderson', 'Susan', 'Anderson', 'susan.anderson@example.com', 0),  ('password606', 'JamesHarris', 'James', 'Harris', 'james.harris@example.com', 0),  ('password707', 'LindaMartinez', 'Linda', 'Martinez', 'linda.martinez@example.com', 1); |
| Customer | INSERT INTO Customer (customerName, paymentDetails, orderHistory, orderDetails, userID) VALUES  ('JohnDoe', 'Payment 1', 'OrderHistory 1', 'OrderDetails 1', 1),  ('JaneSmith', 'Payment 2', 'OrderHistory 2', 'OrderDetails 2', 2),  ('MichaelBrown', 'Payment 3', 'OrderHistory 3', 'OrderDetails 3', 3),  ('EmilyJohnson', 'Payment 4', 'OrderHistory 4', 'OrderDetails 4', 4),  ('DavidWilson', 'Payment 5', 'OrderHistory 5', 'OrderDetails 5', 5),  ('SarahLee', 'Payment 6', 'OrderHistory 6', 'OrderDetails 6', 6),  ('RobertTaylor', 'Payment 7', 'OrderHistory 7', 'OrderDetails 7', 7),  ('SusanAnderson', 'Payment 8', 'OrderHistory 8', 'OrderDetails 8', 8),  ('JamesHarris', 'Payment 9', 'OrderHistory 9', 'OrderDetails 9', 9),  ('LindaMartinez', 'Payment 10', 'OrderHistory 10', 'OrderDetails 10', 10); |
| Product | INSERT INTO Product (productID, productName, category, stock, isAvailable) VALUES  (1, 'Product A', 'Category 1', 100, 1),  (2, 'Product B', 'Category 2', 75, 1),  (3, 'Product C', 'Category 1', 50, 0),  (4, 'Product D', 'Category 3', 120, 1),  (5, 'Product E', 'Category 2', 90, 1),  (6, 'Product F', 'Category 1', 60, 0),  (7, 'Product G', 'Category 3', 80, 1),  (8, 'Product H', 'Category 2', 110, 1),  (9, 'Product I', 'Category 1', 40, 0),  (10, 'Product J', 'Category 3', 70, 1); |
| Employees | INSERT INTO Employees (employeeID, name, orderID, phoneNumber, userID, orderIDs) VALUES  (1, 'Employee 1', 'Order001', '123-456-7890', 1, 101),  (2, 'Employee 2', 'Order002', '234-567-8901', 2, 102),  (3, 'Employee 3', 'Order003', '345-678-9012', 3, 103),  (4, 'Employee 4', 'Order004', '456-789-0123', 4, 104),  (5, 'Employee 5', 'Order005', '567-890-1234', 5, 105),  (6, 'Employee 6', 'Order006', '678-901-2345', 6, 106),  (7, 'Employee 7', 'Order007', '789-012-3456', 7, 107),  (8, 'Employee 8', 'Order008', '890-123-4567', 8, 108),  (9, 'Employee 9', 'Order009', '901-234-5678', 9, 109),  (10, 'Employee 10', 'Order010', '012-345-6789', 10, 110); |
| Order\_Details | INSERT INTO Order\_Details (orderID, orderShippedDate, address, customerID, employeeID) VALUES  (1, '2023-01-01 08:00:00', '123 Main St', 1, 1),  (2, '2023-01-02 09:30:00', '456 Elm St', 2, 2),  (3, '2023-01-03 10:45:00', '789 Oak St', 3, 3),  (4, '2023-01-04 12:15:00', '101 Pine St', 4, 4),  (5, '2023-01-05 14:20:00', '222 Maple St', 5, 5),  (6, '2023-01-06 16:30:00', '333 Cedar St', 6, 6),  (7, '2023-01-07 18:45:00', '444 Birch St', 7, 7),  (8, '2023-01-08 21:00:00', '555 Redwood St', 8, 8),  (9, '2023-01-09 23:15:00', '666 Walnut St', 9, 9),  (10, '2023-01-10 01:30:00', '777 Spruce St', 10, 10); |
| Category | INSERT INTO Category (name, description, quantity, thumbnail) VALUES  ('Category 1', 'Description for Category 1', 100, NULL),  ('Category 2', 'Description for Category 2', 75, NULL),  ('Category 3', 'Description for Category 3', 50, NULL),  ('Category 4', 'Description for Category 4', 120, NULL),  ('Category 5', 'Description for Category 5', 90, NULL),  ('Category 6', 'Description for Category 6', 60, NULL),  ('Category 7', 'Description for Category 7', 80, NULL),  ('Category 8', 'Description for Category 8', 110, NULL),  ('Category 9', 'Description for Category 9', 40, NULL),  ('Category 10', 'Description for Category 10', 70, NULL); |
| Suppliers | INSERT INTO Suppliers (supplierID, supplierName, supplierAddress, supplierPhoneNumber, supplierEmail)  VALUES  (1, 'Supplier A', '123 Main St', '123-456-7890', 'supplierA@example.com'),  (2, 'Supplier B', '456 Elm St', '234-567-8901', 'supplierB@example.com'),  (3, 'Supplier C', '789 Oak St', '345-678-9012', 'supplierC@example.com'),  (4, 'Supplier D', '101 Pine St', '456-789-0123', 'supplierD@example.com'),  (5, 'Supplier E', '222 Maple St', '567-890-1234', 'supplierE@example.com'),  (6, 'Supplier F', '333 Cedar St', '678-901-2345', 'supplierF@example.com'),  (7, 'Supplier G', '444 Birch St', '789-012-3456', 'supplierG@example.com'),  (8, 'Supplier H', '555 Redwood St', '890-123-4567', 'supplierH@example.com'),  (9, 'Supplier I', '666 Walnut St', '901-234-5678', 'supplierI@example.com'),  (10, 'Supplier J', '777 Spruce St', '012-345-6789', 'supplierJ@example.com'); |
| Stores | INSERT INTO Stores (address, name, phoneNumber, customerID) VALUES  ('123 Main St', 'Store 1', '123-456-7890', 1),  ('456 Elm St', 'Store 2', '234-567-8901', 2),  ('789 Oak St', 'Store 3', '345-678-9012', 3),  ('101 Pine St', 'Store 4', '456-789-0123', 4),  ('222 Maple St', 'Store 5', '567-890-1234', 5),  ('333 Cedar St', 'Store 6', '678-901-2345', 6),  ('444 Birch St', 'Store 7', '789-012-3456', 7),  ('555 Redwood St', 'Store 8', '890-123-4567', 8),  ('666 Walnut St', 'Store 9', '901-234-5678', 9),  ('777 Spruce St', 'Store 10', '012-345-6789', 10); |
| Stock | INSERT INTO Stock (stockID, name, quantity, category, supplierID) VALUES  (1, 'Product A', 100, 'Category 1', 1),  (2, 'Product B', 75, 'Category 2', 2),  (3, 'Product C', 50, 'Category 1', 3),  (4, 'Product D', 120, 'Category 3', 4),  (5, 'Product E', 90, 'Category 2', 5),  (6, 'Product F', 60, 'Category 1', 6),  (7, 'Product G', 80, 'Category 3', 7),  (8, 'Product H', 110, 'Category 2', 8),  (9, 'Product I', 40, 'Category 1', 9),  (10, 'Product J', 70, 'Category 3', 10); |
| Payment\_Details | INSERT INTO Payment\_Details (price, date, Order\_Details\_orderID, Order\_Details\_productID, Order\_Details\_customerID, Order\_Details\_employeeID)  VALUES  (50.0, '2023-01-01 08:00:00', 1, 1, 1, 1),  (65.0, '2023-01-02 09:30:00', 2, 2, 2, 2),  (75.0, '2023-01-03 10:45:00', 3, 3, 3, 3),  (40.0, '2023-01-04 12:15:00', 4, 4, 4, 4),  (95.0, '2023-01-05 14:20:00', 5, 5, 5, 5),  (60.0, '2023-01-06 16:30:00', 6, 6, 6, 6),  (80.0, '2023-01-07 18:45:00', 7, 7, 7, 7),  (110.0, '2023-01-08 21:00:00', 8, 8, 8, 8),  (45.0, '2023-01-09 23:15:00', 9, 9, 9, 9),  (70.0, '2023-01-10 01:30:00', 10, 10, 10, 10); |
| Product\_has\_Category | INSERT INTO Product\_has\_Category (Product\_productID, Category\_categoryID) VALUES  (1, 1),  (2, 2),  (3, 1),  (4, 3),  (5, 2),  (6, 1),  (7, 3),  (8, 2),  (9, 1),  (10, 3); |
| Stock\_has\_Product | INSERT INTO Stock\_has\_Product (Stock\_stockID, Product\_productID) VALUES  (1, 1),  (2, 2),  (3, 3),  (4, 4),  (5, 5),  (6, 6),  (7, 7),  (8, 8),  (9, 9),  (10, 10); |
| Stock\_has\_Suppliers | INSERT INTO Stock\_has\_Suppliers (Stock\_stockID, Suppliers\_supplierID) VALUES  (1, 1),  (2, 2),  (3, 3),  (4, 4),  (5, 5),  (6, 6),  (7, 7),  (8, 8),  (9, 9),  (10, 10); |
| Order\_Details\_has\_Product | INSERT INTO Order\_Details\_has\_Product (Order\_Details\_orderID, Order\_Details\_customerID, Order\_Details\_employeeID, Product\_productID)  VALUES  (1, 1, 1, 1),  (2, 2, 2, 2),  (3, 3, 3, 3),  (4, 4, 4, 4),  (5, 5, 5, 5),  (6, 6, 6, 6),  (7, 7, 7, 7),  (8, 8, 8, 8),  (9, 9, 9, 9),  (10, 10, 10, 10); |

# 10. Normalization Check

Higher normalization forms are considered better in database design for many reasons, including reduced data redundancy and improved data integrity. The normal forms we are concerned with are 1st, 2nd, and 3rd. The definitions are as follows:

1. First Normal Form:
   1. Each table cell should contain a single value.
   2. Each record needs to be unique.
2. Second Normal Form:
   1. Be in 1NF
   2. No non-prime attributes that are functionally dependent on any other non-prime attribute A non-prime is one that is not part of the key
3. Third Normal Form
   1. Be in 2NF
   2. Has no transitive functional dependencies (i.e., no chain of dependencies)

One of the best ways to improve the normalization of the database is splitting up the many-to-many relationships with a ‘junction table.’ Specifically, this helps to prevent transitive functional dependencies. Our database uses the naming convention of ‘Table1\_has\_Table2’ to denote these junction tables. The following Table 10.1 shows the normal form and a short justification for each entity.

### Table 10.1 – Normalization

|  |  |  |
| --- | --- | --- |
| Table | NF | Justification |
| Users | 3 | Primary key, atomic attributes, no non-prime attributes functionally dependent on another, no transitive functional dependencies. While the ‘name’ attribute is derived from ‘Fname’ and ‘Lname,’ the derived attribute is still dependent on the primary key |
| Customer | 3 | Primary key, atomic attributes, no non-prime attributes functionally dependent on another, no transitive functional dependencies |
| Product | 3 | Primary key, atomic attributes, no non-prime attributes functionally dependent on another, no transitive functional dependencies |
| Employees | 3 | Primary key, atomic attributes, no non-prime attributes functionally dependent on another, no transitive functional dependencies |
| Order\_Details | 3 | Primary key, atomic attributes, no non-prime attributes functionally dependent on another, no transitive functional dependencies |
| Category | 3 | Primary key, atomic attributes, no non-prime attributes functionally dependent on another, no transitive functional dependencies. While the ‘quantity’ attribute is calculated, the attribute is still dependent on the primary key |
| Suppliers | 3 | Primary key, atomic attributes, no non-prime attributes functionally dependent on another, no transitive functional dependencies |
| Stores | 3 | Primary key, atomic attributes, no non-prime attributes functionally dependent on another, no transitive functional dependencies |
| Stock | 3 | Primary key, atomic attributes, no non-prime attributes functionally dependent on another, no transitive functional dependencies. While the ‘quantity’ attribute is calculated, the attribute is still dependent on the primary key |
| Payment\_Details | 2 | Composite primary key, atomic attributes, no non-prime attributes functionally dependent on another, but is a weak entity, so cannot be 3rd NF (dependent on another entity for its identity) |
| Product\_has\_Category | 3 | Composite primary key, atomic attributes, no non-prime attributes functionally dependent on another, no transitive functional dependencies |
| Stock\_has\_Product | 3 | Composite primary key, atomic attributes, no non-prime attributes functionally dependent on another, no transitive functional dependencies |
| Stock\_has\_Suppliers | 3 | Composite primary key, atomic attributes, no non-prime attributes functionally dependent on another, no transitive functional dependencies |
| Order\_Details\_has\_Product | 3 | Composite primary key, atomic attributes, no non-prime attributes functionally dependent on another, no transitive functional dependencies |

# 11. Application development

a. UX design:

### Figure 11.1 – UX Design

1. Login Page: On the Start page i.e the Login page, user will be authenticated based on a valid username and password. Upon authentication, the user will be directed to different dashboards depending on whether the user is a customer or an admin.

2) Customer Dashboard: The Customer Dashboard will have tabs like Order Management, Search Item, Items, My Profile.

2.1) Order Management: The Customer can manage their order by viewing the stores they own and then respective order details for each store. Furthermore, they can view payment details as well.

2.2) Search Item: The Customer can search for an item by the name of the item.

2.3) Items: Under the items tab, the Customer can view different categories and then products within each category. Then they can add the items they want to purchase to a cart.

2.4) My Profile: Finally, the Customer can view their profile and update the details as well.

3) Admin Dashboard: The Admin Dashboard consists of tabs like Items, Search Item, Suppliers, Customers, Employees.

3.1) Items: In the items tab, the Admin can view all the different products in his warehouse.

3.2) Search Item: The Admin can search for an item by the name of the item.

3.3) Suppliers: Firstly, the Admin can add, remove or update the details of a supplier. Moreover, in the supplier's tab, the Admin can view the stock delivered by a supplier and the products in each stock delivery.

3.4) Customers: The Admin can add, remove or update the details of a customer. Additionally, in the customers tab, the Amin can view the stores of a particular customer and the respective order details of each store.

3.5) Employees: The Admin can add, remove or update the details of an employee. Furthermore, the Admin can view the orders handled by a particular employee and the respective payment details.

# 12. View Data Preparation

Views are a feature of MySQL that can simplify development, enhance security, and provide flexibility in data access and management.

Many of our tables succinctly cover the necessary information for its given page. For example, the profile page matches the information in the Users table.

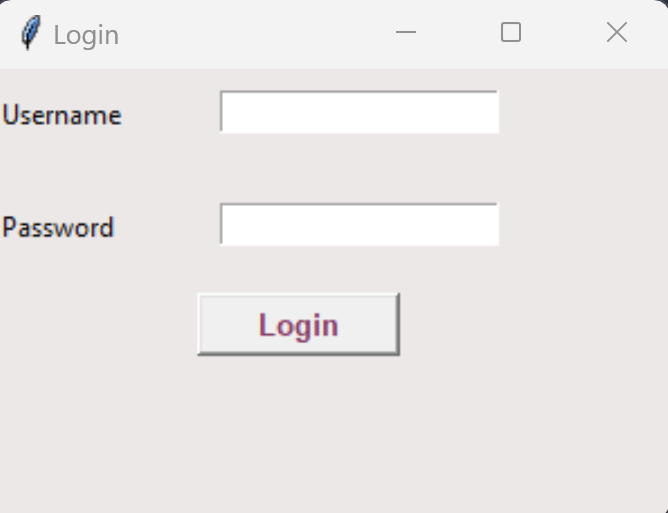
### Table 11.1 – View Creation

|  |
| --- |
| CREATE VIEW admin\_dashboard\_view AS  SELECT  Order\_Details.orderID,  Order\_Details.orderShippedDate,  Customer.customerID,  Customer.customerName,  Product.productID,  Product.productName,  Stock.quantity,  Product.category  FROM  Order\_Details  JOIN  Customer ON Order\_details.customerID = Customer.customerID  JOIN  Order\_Details\_has\_Product ON Order\_Details\_has\_Product.Order\_Details\_orderID = Order\_Details.orderID  JOIN  Product ON Order\_Details\_has\_Product.Product\_productID = Product.productID  JOIN  Stock\_has\_Product ON Stock\_has\_Product.Product\_productID = Product.productID  JOIN  Stock ON Stock.stockID = Stock\_has\_Product.Stock\_stockID; |
| CREATE VIEW customer\_dashboard\_view AS  SELECT  Customer.customerID,  Customer.customerName,  Customer.paymentDetails,  Customer.orderHistory,  Order\_Details.orderID,  Order\_Details.orderShippedDate,  Order\_Details.address,  Product.productID,  Product.productName,  Payment\_Details.price  FROM  Customer  JOIN  Order\_Details ON Customer.customerID = Order\_Details.customerID  JOIN  Order\_Details\_has\_Product ON Order\_Details\_has\_Product.Order\_Details\_orderID = Order\_Details.orderID  JOIN  Product ON Order\_Details\_has\_Product.Product\_productID = Product.productID  JOIN  Payment\_Details ON Payment\_Details.Order\_Details\_orderID = Order\_Details.orderID  JOIN  Product ON Order\_Details.productID = Product.productID; |
| CREATE VIEW search\_items\_view AS  SELECT  Product.productID,  Product.productName,  Product.category,  Product.stock,  Product.isAvailable,  Category.categoryID,  Category.name AS categoryName,  Category.description,  Category.quantity,  Category.thumbnail  FROM  Product  JOIN  Product\_has\_Category ON Product\_has\_Category.Product\_productID = Product.productID  JOIN  Category ON Category.categoryID = Product\_has\_Category.Category\_categoryID; |

# 13. Graphical User Interface Design

To create our graphical user interface (GUI), we use the importing MySQL connector, a self-contained Python driver for communicating with MySQL servers. The design philosophy of the code is that there is one script, main.py, with the different pages in separate files. For example, the suppliers can be seen in suppliers.py.

Currently, we have a simple login page, Figure 13.1, and main menu page, Figure 13.2.



### Figure 13.1 – Login Page



### Figure 13.2 – Main menu

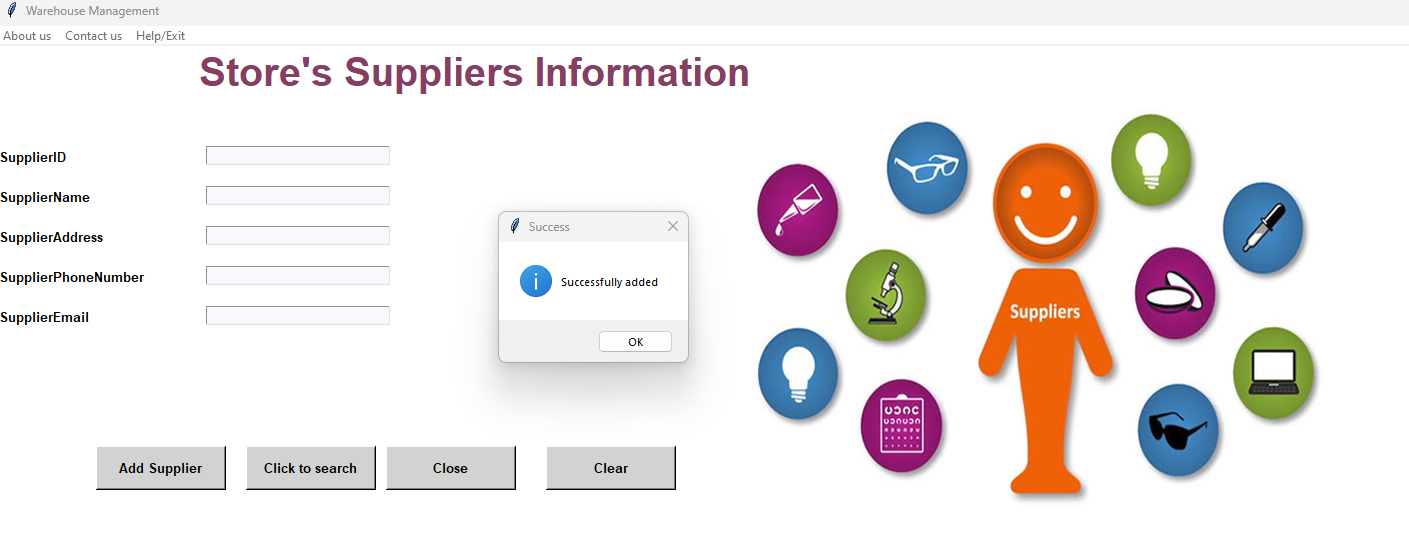
One important feature of the database is the ability to search. Figures 13.3 and 13.4 show the Suppliers page and the result of executed the search.



### Figure 13.3 – Suppliers Page



### Figure 13.4 – Adding Supplier

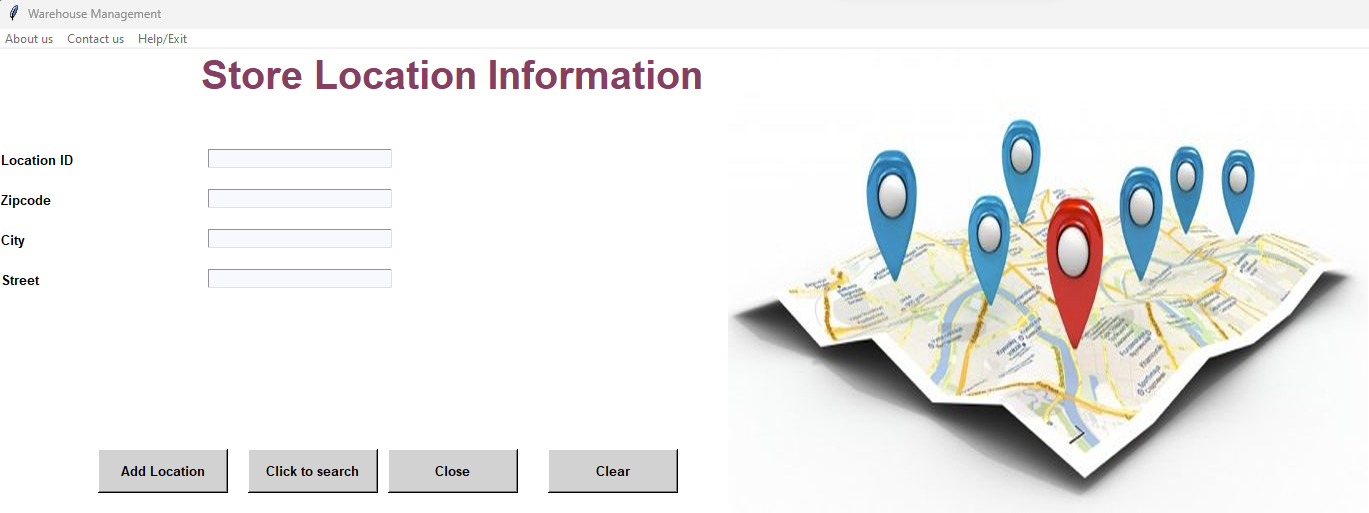


### Figure 13.5 – Successful Adding Pop-Up

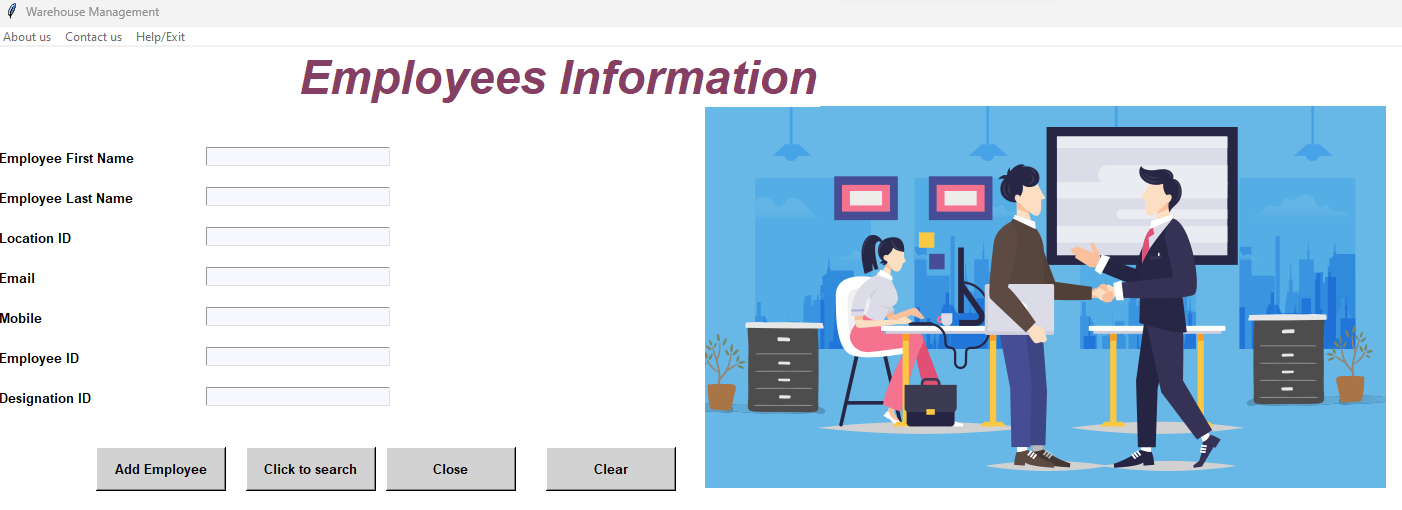


### Figure 13.6 – Search for a Supplier

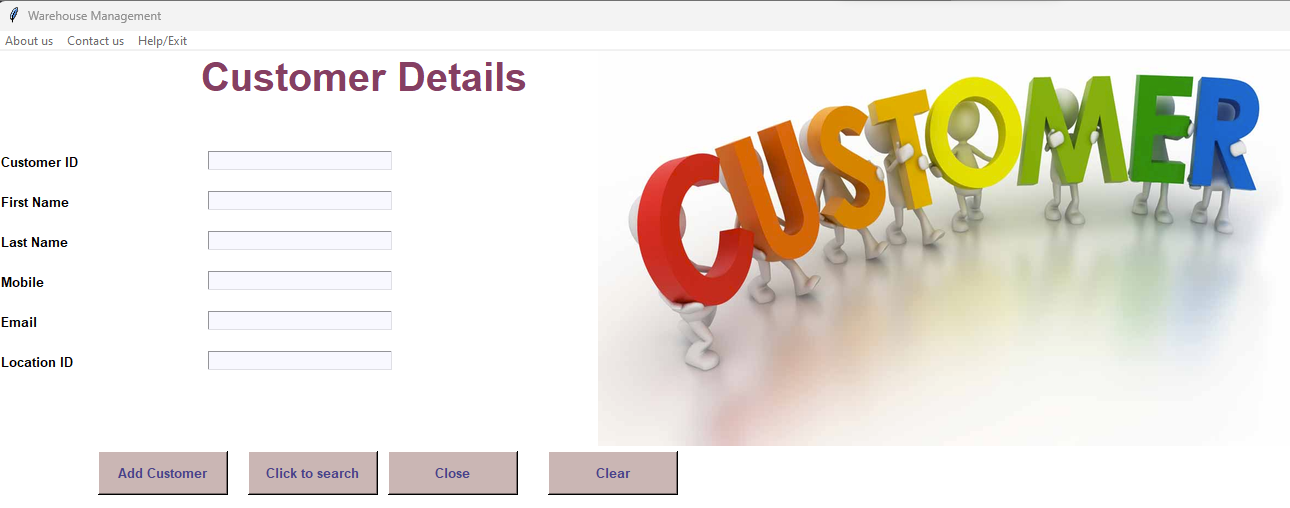
The remaining figures in this section show off some of our other pages:



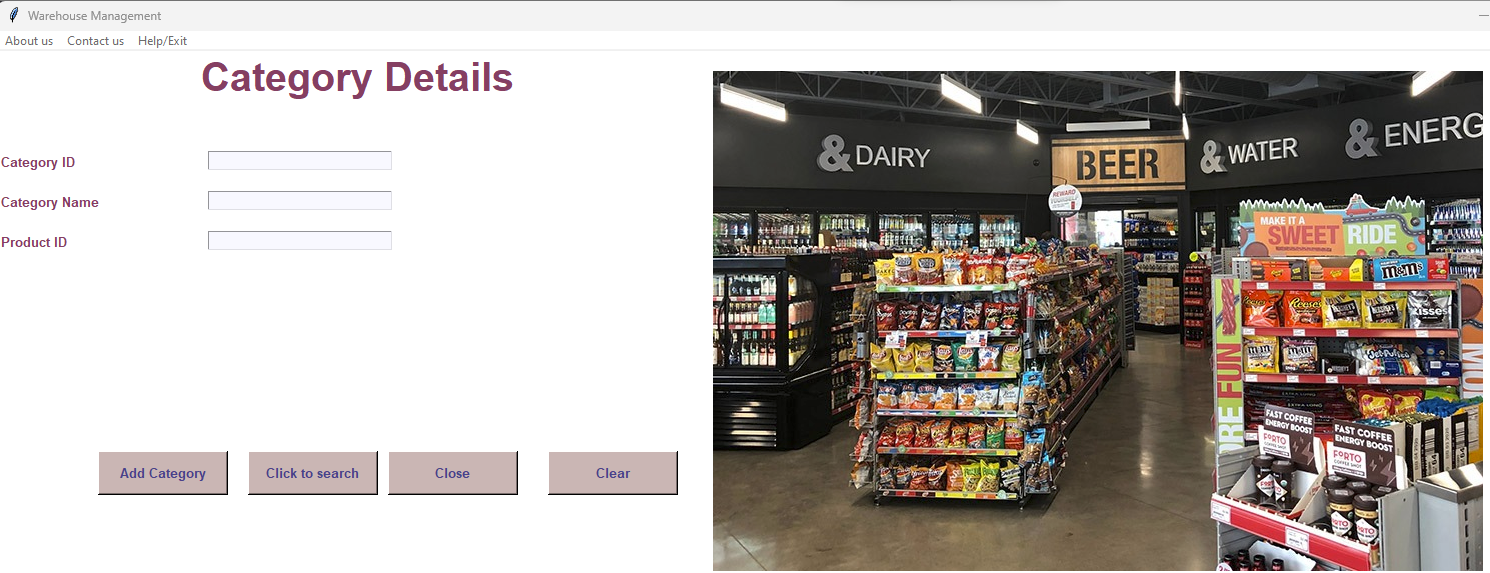
### Figure 13.7 – Store Location



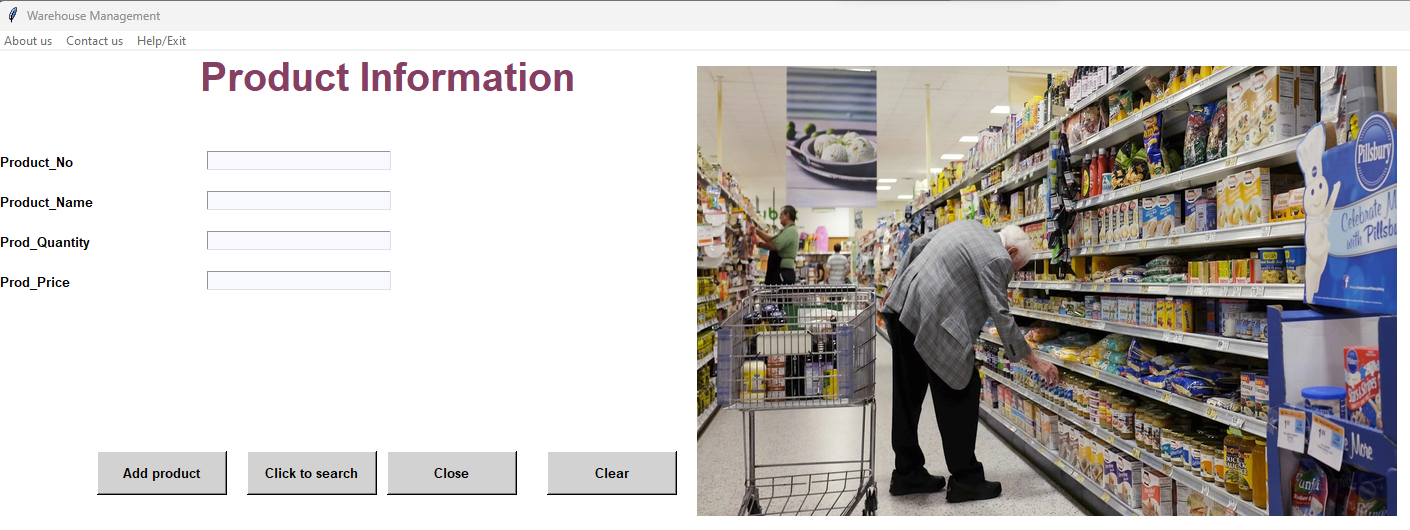
### Figure 13.8 – Employee Information



### Figure 13.9 – Customer Details



### Figure 13.10 – Customer Details



### Figure 13.11 – Product Details

# Works Cited

[1] “Capterra: Find & evaluate top software & business services,” Warehouse Management Software, <https://www.capterra.com/sem-compare/warehouse-management-software/?utm_source=ps-google&utm_medium=ppc&utm_campaign=:1:CAP:2:COM:3:All:4:US:5:BAU:6:SOF:7:Desktop:8:BR:9:Warehouse_Management&network=g&gclid=EAIaIQobChMI3o6cj5umgQMVmklyCh310AN7EAAYAyAAEgLcOfD_BwE> (accessed Sep. 12, 2023).

[2] “Capterra: Find & evaluate top software & business services,” Warehouse Management Software, <https://www.capterra.com/> (accessed Sep. 12, 2023)

# Appendix

## SQL to Generate and Insert into Tables

CREATE SCHEMA IF NOT EXISTS warehouse\_management;

USE warehouse\_management;

CREATE TABLE IF NOT EXISTS Users (

userID INT PRIMARY KEY AUTO\_INCREMENT,

password VARCHAR(45) NULL,

name VARCHAR(50) NULL,

Fname VARCHAR(25) NULL,

Lname VARCHAR(25) NULL,

email VARCHAR(45) NULL,

isAdmin TINYINT NULL

);

CREATE TABLE IF NOT EXISTS Customer (

customerID INT AUTO\_INCREMENT,

customerName VARCHAR(45) NULL,

paymentDetails VARCHAR(45) NULL,

orderHistory VARCHAR(45) NULL,

orderDetails VARCHAR(45) NULL,

userID INT NOT NULL,

PRIMARY KEY (customerID, userID),

INDEX fk\_Customer\_Users1\_idx (userID ASC),

CONSTRAINT fk\_Customer\_Users1

FOREIGN KEY (userID)

REFERENCES Users (userID)

);

CREATE TABLE IF NOT EXISTS Product (

productID INT PRIMARY KEY,

productName VARCHAR(45) NULL,

category VARCHAR(45) NULL,

stock INT NULL,

isAvailable TINYINT NULL

);

CREATE TABLE IF NOT EXISTS Employees (

employeeID INT NOT NULL,

name VARCHAR(45) NULL,

orderID VARCHAR(45) NULL,

phoneNumber VARCHAR(45) NULL,

userID INT NOT NULL,

orderIDs INT NULL,

PRIMARY KEY (employeeID, userID),

INDEX fk\_Employees\_Users\_idx (userID ASC),

CONSTRAINT fk\_Employees\_Users

FOREIGN KEY (userID)

REFERENCES Users (userID)

);

CREATE TABLE IF NOT EXISTS Order\_Details (

orderID INT NOT NULL,

orderShippedDate DATETIME NULL,

address VARCHAR(45) NULL,

customerID INT NOT NULL,

employeeID INT NOT NULL,

PRIMARY KEY (orderID, customerID, employeeID),

INDEX fk\_Order\_Details\_Customer1\_idx (customerID ASC),

INDEX fk\_Order\_Details\_Employees1\_idx (employeeID ASC),

CONSTRAINT fk\_Order\_Details\_Customer1

FOREIGN KEY (customerID)

REFERENCES Customer (customerID),

CONSTRAINT fk\_Order\_Details\_Employees1

FOREIGN KEY (employeeID)

REFERENCES Employees (employeeID)

);

CREATE TABLE IF NOT EXISTS Category (

categoryID INT PRIMARY KEY AUTO\_INCREMENT,

name VARCHAR(45) NULL,

description BLOB NULL,

quantity INT NULL,

thumbnail BLOB NULL

);

CREATE TABLE IF NOT EXISTS Suppliers (

supplierID INT PRIMARY KEY,

supplierName VARCHAR(45) NULL,

supplierAddress VARCHAR(45) NULL,

supplierPhoneNumber VARCHAR(45) NULL,

supplierEmail VARCHAR(45) NULL

);

CREATE TABLE IF NOT EXISTS Stores (

storeID INT NOT NULL AUTO\_INCREMENT,

address VARCHAR(45) NULL,

name VARCHAR(45) NULL,

phoneNumber VARCHAR(45) NULL,

customerID INT NOT NULL,

PRIMARY KEY (storeID, customerID),

INDEX fk\_Stores\_Customer1\_idx (customerID ASC),

CONSTRAINT fk\_Stores\_Customer1

FOREIGN KEY (customerID)

REFERENCES Customer (customerID)

);

CREATE TABLE IF NOT EXISTS Stock (

stockID INT PRIMARY KEY,

name VARCHAR(45) NULL,

quantity INT NULL,

category VARCHAR(45) NULL,

supplierID INT NULL

);

CREATE TABLE IF NOT EXISTS Payment\_Details (

price DECIMAL NULL,

date DATETIME NULL,

Order\_Details\_orderID INT NOT NULL,

Order\_Details\_productID INT NOT NULL,

Order\_Details\_customerID INT NOT NULL,

Order\_Details\_employeeID INT NOT NULL,

PRIMARY KEY (Order\_Details\_orderID, Order\_Details\_productID, Order\_Details\_customerID,

Order\_Details\_employeeID),

INDEX fk\_Payment\_Order\_Details1\_idx (Order\_Details\_orderID ASC, Order\_Details\_productID ASC,

Order\_Details\_customerID ASC, Order\_Details\_employeeID ASC),

CONSTRAINT fk\_Payment\_Order\_Details1

FOREIGN KEY (Order\_Details\_orderID , Order\_Details\_customerID , Order\_Details\_employeeID)

REFERENCES Order\_Details (orderID , customerID , employeeID)

);

CREATE TABLE IF NOT EXISTS Product\_has\_Category (

Product\_productID INT NOT NULL,

Category\_categoryID INT NOT NULL,

PRIMARY KEY (Product\_productID, Category\_categoryID),

INDEX fk\_Product\_has\_Category\_Category1\_idx (Category\_categoryID ASC),

INDEX fk\_Product\_has\_Category\_Product1\_idx (Product\_productID ASC),

CONSTRAINT fk\_Product\_has\_Category\_Product1

FOREIGN KEY (Product\_productID)

REFERENCES Product (productID),

CONSTRAINT fk\_Product\_has\_Category\_Category1

FOREIGN KEY (Category\_categoryID)

REFERENCES Category (categoryID)

);

CREATE TABLE IF NOT EXISTS Stock\_has\_Product (

Stock\_stockID INT NOT NULL,

Product\_productID INT NOT NULL,

PRIMARY KEY (Stock\_stockID, Product\_productID),

INDEX fk\_Stock\_has\_Product\_Product1\_idx (Product\_productID ASC),

INDEX fk\_Stock\_has\_Product\_Stock1\_idx (Stock\_stockID ASC),

CONSTRAINT fk\_Stock\_has\_Product\_Stock1

FOREIGN KEY (Stock\_stockID)

REFERENCES Stock (stockID),

CONSTRAINT fk\_Stock\_has\_Product\_Product1

FOREIGN KEY (Product\_productID)

REFERENCES Product (productID)

);

CREATE TABLE IF NOT EXISTS Stock\_has\_Suppliers (

Stock\_stockID INT NOT NULL,

Suppliers\_supplierID INT NOT NULL,

PRIMARY KEY (Stock\_stockID, Suppliers\_supplierID),

INDEX fk\_Stock\_has\_Suppliers\_Suppliers1\_idx (Suppliers\_supplierID ASC),

INDEX fk\_Stock\_has\_Suppliers\_Stock1\_idx (Stock\_stockID ASC),

CONSTRAINT fk\_Stock\_has\_Suppliers\_Stock1

FOREIGN KEY (Stock\_stockID)

REFERENCES Stock (stockID),

CONSTRAINT fk\_Stock\_has\_Suppliers\_Suppliers1

FOREIGN KEY (Suppliers\_supplierID)

REFERENCES Suppliers (supplierID)

);

CREATE TABLE IF NOT EXISTS Order\_Details\_has\_Product (

Order\_Details\_orderID INT NOT NULL,

Order\_Details\_customerID INT NOT NULL,

Order\_Details\_employeeID INT NOT NULL,

Product\_productID INT NOT NULL,

PRIMARY KEY (Order\_Details\_orderID, Order\_Details\_customerID, Order\_Details\_employeeID,

Product\_productID),

INDEX fk\_Order\_Details\_has\_Product\_Product1\_idx (Product\_productID ASC),

INDEX fk\_Order\_Details\_has\_Product\_Order\_Details1\_idx (Order\_Details\_orderID ASC,

Order\_Details\_customerID ASC, Order\_Details\_employeeID ASC),

CONSTRAINT fk\_Order\_Details\_has\_Product\_Order\_Details1

FOREIGN KEY (Order\_Details\_orderID , Order\_Details\_customerID , Order\_Details\_employeeID)

REFERENCES Order\_Details (orderID , customerID , employeeID),

CONSTRAINT fk\_Order\_Details\_has\_Product\_Product1

FOREIGN KEY (Product\_productID)

REFERENCES Product (productID)

);

SET FOREIGN\_KEY\_CHECKS=0;

INSERT INTO Users (password, name, Fname, Lname, email, isAdmin) VALUES

('password123', 'JohnDoe', 'John', 'Doe', 'john.doe@example.com', 0),

('password456', 'JaneSmith', 'Jane', 'Smith', 'jane.smith@example.com', 0),

('password789', 'MichaelBrown', 'Michael', 'Brown', 'michael.brown@example.com', 1),

('password101', 'EmilyJohnson', 'Emily', 'Johnson', 'emily.johnson@example.com', 0),

('password202', 'DavidWilson', 'David', 'Wilson', 'david.wilson@example.com', 0),

('password303', 'SarahLee', 'Sarah', 'Lee', 'sarah.lee@example.com', 1),

('password404', 'RobertTaylor', 'Robert', 'Taylor', 'robert.taylor@example.com', 0),

('password505', 'SusanAnderson', 'Susan', 'Anderson', 'susan.anderson@example.com', 0),

('password606', 'JamesHarris', 'James', 'Harris', 'james.harris@example.com', 0),

('password707', 'LindaMartinez', 'Linda', 'Martinez', 'linda.martinez@example.com', 1);

INSERT INTO Customer (customerName, paymentDetails, orderHistory, orderDetails, userID) VALUES

('JohnDoe', 'Payment 1', 'OrderHistory 1', 'OrderDetails 1', 1),

('JaneSmith', 'Payment 2', 'OrderHistory 2', 'OrderDetails 2', 2),

('MichaelBrown', 'Payment 3', 'OrderHistory 3', 'OrderDetails 3', 3),

('EmilyJohnson', 'Payment 4', 'OrderHistory 4', 'OrderDetails 4', 4),

('DavidWilson', 'Payment 5', 'OrderHistory 5', 'OrderDetails 5', 5),

('SarahLee', 'Payment 6', 'OrderHistory 6', 'OrderDetails 6', 6),

('RobertTaylor', 'Payment 7', 'OrderHistory 7', 'OrderDetails 7', 7),

('SusanAnderson', 'Payment 8', 'OrderHistory 8', 'OrderDetails 8', 8),

('JamesHarris', 'Payment 9', 'OrderHistory 9', 'OrderDetails 9', 9),

('LindaMartinez', 'Payment 10', 'OrderHistory 10', 'OrderDetails 10', 10);

INSERT INTO Product (productID, productName, category, stock, isAvailable) VALUES

(1, 'Product A', 'Category 1', 100, 1),

(2, 'Product B', 'Category 2', 75, 1),

(3, 'Product C', 'Category 1', 50, 0),

(4, 'Product D', 'Category 3', 120, 1),

(5, 'Product E', 'Category 2', 90, 1),

(6, 'Product F', 'Category 1', 60, 0),

(7, 'Product G', 'Category 3', 80, 1),

(8, 'Product H', 'Category 2', 110, 1),

(9, 'Product I', 'Category 1', 40, 0),

(10, 'Product J', 'Category 3', 70, 1);

INSERT INTO Employees (employeeID, name, orderID, phoneNumber, userID, orderIDs) VALUES

(1, 'Employee 1', 'Order001', '123-456-7890', 1, 101),

(2, 'Employee 2', 'Order002', '234-567-8901', 2, 102),

(3, 'Employee 3', 'Order003', '345-678-9012', 3, 103),

(4, 'Employee 4', 'Order004', '456-789-0123', 4, 104),

(5, 'Employee 5', 'Order005', '567-890-1234', 5, 105),

(6, 'Employee 6', 'Order006', '678-901-2345', 6, 106),

(7, 'Employee 7', 'Order007', '789-012-3456', 7, 107),

(8, 'Employee 8', 'Order008', '890-123-4567', 8, 108),

(9, 'Employee 9', 'Order009', '901-234-5678', 9, 109),

(10, 'Employee 10', 'Order010', '012-345-6789', 10, 110);

INSERT INTO Order\_Details (orderID, orderShippedDate, address, customerID, employeeID) VALUES

(1, '2023-01-01 08:00:00', '123 Main St', 1, 1),

(2, '2023-01-02 09:30:00', '456 Elm St', 2, 2),

(3, '2023-01-03 10:45:00', '789 Oak St', 3, 3),

(4, '2023-01-04 12:15:00', '101 Pine St', 4, 4),

(5, '2023-01-05 14:20:00', '222 Maple St', 5, 5),

(6, '2023-01-06 16:30:00', '333 Cedar St', 6, 6),

(7, '2023-01-07 18:45:00', '444 Birch St', 7, 7),

(8, '2023-01-08 21:00:00', '555 Redwood St', 8, 8),

(9, '2023-01-09 23:15:00', '666 Walnut St', 9, 9),

(10, '2023-01-10 01:30:00', '777 Spruce St', 10, 10);

INSERT INTO Category (name, description, quantity, thumbnail) VALUES

('Category 1', 'Description for Category 1', 100, NULL),

('Category 2', 'Description for Category 2', 75, NULL),

('Category 3', 'Description for Category 3', 50, NULL),

('Category 4', 'Description for Category 4', 120, NULL),

('Category 5', 'Description for Category 5', 90, NULL),

('Category 6', 'Description for Category 6', 60, NULL),

('Category 7', 'Description for Category 7', 80, NULL),

('Category 8', 'Description for Category 8', 110, NULL),

('Category 9', 'Description for Category 9', 40, NULL),

('Category 10', 'Description for Category 10', 70, NULL);

INSERT INTO Suppliers (supplierID, supplierName, supplierAddress, supplierPhoneNumber, supplierEmail)

VALUES

(1, 'Supplier A', '123 Main St', '123-456-7890', 'supplierA@example.com'),

(2, 'Supplier B', '456 Elm St', '234-567-8901', 'supplierB@example.com'),

(3, 'Supplier C', '789 Oak St', '345-678-9012', 'supplierC@example.com'),

(4, 'Supplier D', '101 Pine St', '456-789-0123', 'supplierD@example.com'),

(5, 'Supplier E', '222 Maple St', '567-890-1234', 'supplierE@example.com'),

(6, 'Supplier F', '333 Cedar St', '678-901-2345', 'supplierF@example.com'),

(7, 'Supplier G', '444 Birch St', '789-012-3456', 'supplierG@example.com'),

(8, 'Supplier H', '555 Redwood St', '890-123-4567', 'supplierH@example.com'),

(9, 'Supplier I', '666 Walnut St', '901-234-5678', 'supplierI@example.com'),

(10, 'Supplier J', '777 Spruce St', '012-345-6789', 'supplierJ@example.com');

INSERT INTO Stores (address, name, phoneNumber, customerID) VALUES

('123 Main St', 'Store 1', '123-456-7890', 1),

('456 Elm St', 'Store 2', '234-567-8901', 2),

('789 Oak St', 'Store 3', '345-678-9012', 3),

('101 Pine St', 'Store 4', '456-789-0123', 4),

('222 Maple St', 'Store 5', '567-890-1234', 5),

('333 Cedar St', 'Store 6', '678-901-2345', 6),

('444 Birch St', 'Store 7', '789-012-3456', 7),

('555 Redwood St', 'Store 8', '890-123-4567', 8),

('666 Walnut St', 'Store 9', '901-234-5678', 9),

('777 Spruce St', 'Store 10', '012-345-6789', 10);

INSERT INTO Stock (stockID, name, quantity, category, supplierID) VALUES

(1, 'Product A', 100, 'Category 1', 1),

(2, 'Product B', 75, 'Category 2', 2),

(3, 'Product C', 50, 'Category 1', 3),

(4, 'Product D', 120, 'Category 3', 4),

(5, 'Product E', 90, 'Category 2', 5),

(6, 'Product F', 60, 'Category 1', 6),

(7, 'Product G', 80, 'Category 3', 7),

(8, 'Product H', 110, 'Category 2', 8),

(9, 'Product I', 40, 'Category 1', 9),

(10, 'Product J', 70, 'Category 3', 10);

INSERT INTO Payment\_Details (price, date, Order\_Details\_orderID, Order\_Details\_productID, Order\_Details\_customerID, Order\_Details\_employeeID)

VALUES

(50.0, '2023-01-01 08:00:00', 1, 1, 1, 1),

(65.0, '2023-01-02 09:30:00', 2, 2, 2, 2),

(75.0, '2023-01-03 10:45:00', 3, 3, 3, 3),

(40.0, '2023-01-04 12:15:00', 4, 4, 4, 4),

(95.0, '2023-01-05 14:20:00', 5, 5, 5, 5),

(60.0, '2023-01-06 16:30:00', 6, 6, 6, 6),

(80.0, '2023-01-07 18:45:00', 7, 7, 7, 7),

(110.0, '2023-01-08 21:00:00', 8, 8, 8, 8),

(45.0, '2023-01-09 23:15:00', 9, 9, 9, 9),

(70.0, '2023-01-10 01:30:00', 10, 10, 10, 10);

INSERT INTO Product\_has\_Category (Product\_productID, Category\_categoryID) VALUES

(1, 1),

(2, 2),

(3, 1),

(4, 3),

(5, 2),

(6, 1),

(7, 3),

(8, 2),

(9, 1),

(10, 3);

INSERT INTO Stock\_has\_Product (Stock\_stockID, Product\_productID) VALUES

(1, 1),

(2, 2),

(3, 3),

(4, 4),

(5, 5),

(6, 6),

(7, 7),

(8, 8),

(9, 9),

(10, 10);

INSERT INTO Stock\_has\_Suppliers (Stock\_stockID, Suppliers\_supplierID) VALUES

(1, 1),

(2, 2),

(3, 3),

(4, 4),

(5, 5),

(6, 6),

(7, 7),

(8, 8),

(9, 9),

(10, 10);

INSERT INTO Order\_Details\_has\_Product (Order\_Details\_orderID, Order\_Details\_customerID, Order\_Details\_employeeID, Product\_productID)

VALUES

(1, 1, 1, 1),

(2, 2, 2, 2),

(3, 3, 3, 3),

(4, 4, 4, 4),

(5, 5, 5, 5),

(6, 6, 6, 6),

(7, 7, 7, 7),

(8, 8, 8, 8),

(9, 9, 9, 9),

(10, 10, 10, 10);

SET FOREIGN\_KEY\_CHECKS=1;