Case Study- Managing a Furniture Database

**Introduction:**

Managing a furniture database involves organizing, storing, and retrieving data related to customers, orders, products, and order items in an efficient and structured manner. A well-designed database ensures seamless operations within a furniture store, facilitating inventory management, sales tracking, customer relationship management, and overall business analysis. This guide provides an overview of the essential components and best practices for managing a furniture database.

A furniture database efficiently is crucial for the smooth operation of a furniture store. By organizing data into structured tables and following best practices, you can ensure accurate tracking of inventory, orders, and customer information, leading to better decision-making and enhanced customer service.

**Problem Statement:**

Background:

A furniture store needs an efficient and robust database management system to handle its growing business operations. The store deals with numerous customers, processes various orders daily, and maintains an extensive inventory of furniture products. Accurate and efficient data management is crucial for the store to track customer information, manage inventory levels, process orders, and analyze sales data.

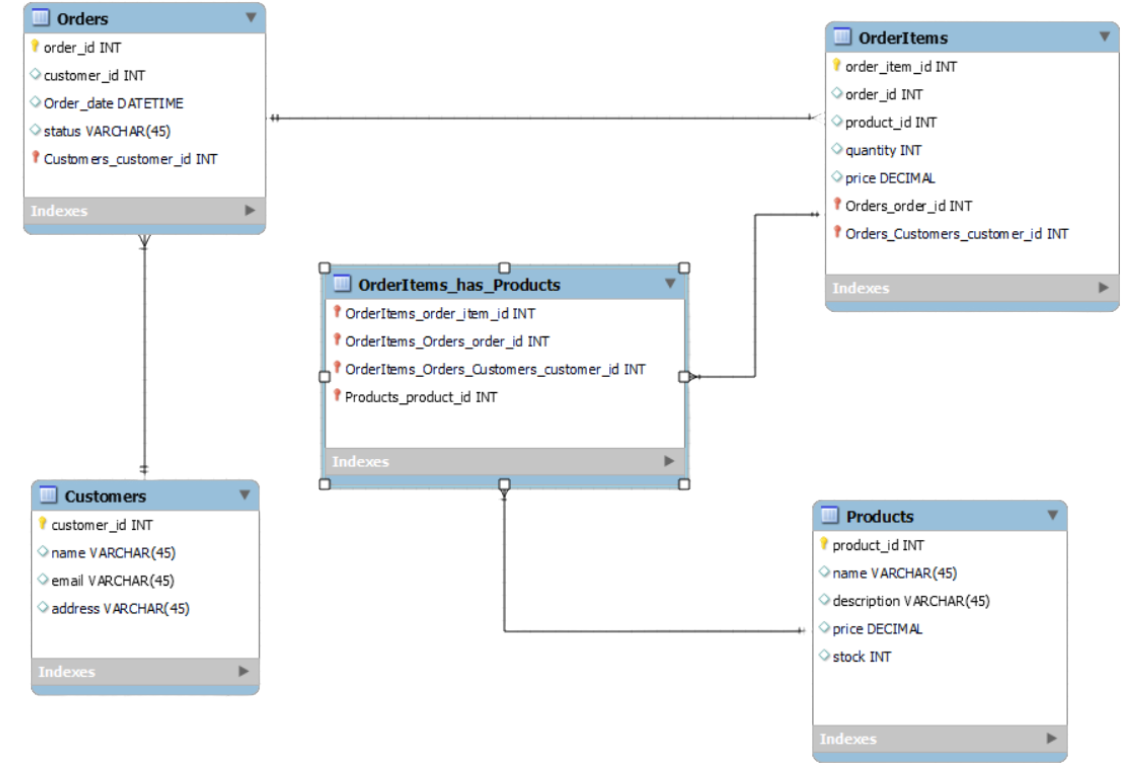
Objective:

Develop and manage a furniture database using MySQL that will organize and streamline the store's operations. The database should include tables for customers, orders, products, and order items, ensuring data integrity and facilitating efficient data retrieval and updates.

The 4 key datasets to be used in the case study are: -

* Customers
* Order Items
* Orders
* Products

**Entity Relationship Diagram:**



**Dataset:**

Create Database case\_study;

Use case\_study;

select \* from customers;

select \* from orderitems;

select \* from order2;

select \* from products;

/\*--------------------------------------------**Customers Table**------------------------------------------\*/

**Create Table & Insert Data:**

create table Customers(

customer\_id int,

name varchar(30),

email varchar(30),

address varchar(50));

insert into Customers(customer\_id,name,email,address)

values (1, 'John Doe', 'john.doe@example.com', '123 Elm St'),

(2, 'Jane Smith', 'jane.smith@example.com', '456 Oak St'),

(3, 'Michael Johnson', 'michael.johnson@example.com', '789 Pine St'),

(4, 'Emily Davis', 'emily.davis@example.com', '101 Maple St'),

(5, 'Christopher Brown', 'christopher.brown@example.com', '202 Birch St'),

(6, 'Amanda Wilson', 'amanda.wilson@example.com', '303 Cedar St'),

(7, 'Joshua Miller', 'joshua.miller@example.com', '404 Spruce St'),

(8, 'Sarah Taylor', 'sarah.taylor@example.com', '505 Ash St'),

(9, 'Daniel Anderson', 'daniel.anderson@example.com', '606 Cherry St'),

(10, 'Jessica Thomas', 'jessica.thomas@example.com', '707 Walnut St'),

(11, 'Matthew Moore', 'matthew.moore@example.com', '808 Willow St'),

(12, 'Olivia Jackson', 'olivia.jackson@example.com', '909 Sycamore St'),

(13, 'David White', 'david.white@example.com', '1010 Redwood St'),

(14, 'Sophia Harris', 'sophia.harris@example.com', '1111 Hickory St'),

(15, 'James Martinez', 'james.martinez@example.com', '1212 Chestnut St'),

(16, 'Isabella Robinson', 'isabella.robinson@example.com', '1313 Aspen St'),

n\(17, 'Benjamin Clark', 'benjamin.clark@example.com', '1414 Beech St'),

(18, 'Mia Rodriguez', 'mia.rodriguez@example.com', '1515 Dogwood St'),

(19, 'Ethan Lewis', 'ethan.lewis@example.com', '1616 Magnolia St'),

(20, 'Ava Lee', 'ava.lee@example.com', '1717 Cypress St');

/\*------------------------------------------- **Orderitems Table**------------------------------------------\*/

**Create Table & Insert Data:**

create table OrderItems(

order\_item\_id int,

order\_id int,

product\_id int,

quantity int,

price int);

INSERT INTO OrderItems (order\_item\_id, order\_id, product\_id, quantity, price)

VALUES

(1, 1, 101, 2, 19.99),

(2, 1, 102, 1, 9.99),

(3, 2, 103, 5, 14.99),

(4, 3, 104, 3, 29.99),

(5, 3, 105, 4, 24.99),

(6, 4, 106, 1, 49.99),

(7, 5, 107, 2, 39.99),

(8, 6, 108, 6, 11.99),

(9, 7, 109, 7, 7.99),

(10, 8, 110, 8, 5.99),

(11, 9, 111, 2, 19.99),

(12, 10, 112, 1, 9.99),

(13, 11, 113, 5, 14.99),

(14, 12, 114, 3, 29.99),

(15, 13, 115, 4, 24.99),

(16, 14, 116, 1, 49.99),

(17, 15, 117, 2, 39.99),

(18, 16, 118, 6, 11.99),

(19, 17, 119, 7, 7.99),

(20, 18, 120, 8, 5.99);

/\*------------------------------------------- **Order2 Table**------------------------------------------\*/

create table Orders2(

order\_id int,

customer\_id int,

order\_date datetime,

status varchar(30));

INSERT INTO Orders2 (order\_id, customer\_id, order\_date, status)

VALUES(1, 1, '2023-01-15', 'Shipped'),

(2, 2, '2023-01-16', 'Processing'),

(3, 3, '2023-01-17', 'Delivered'),

(4, 4, '2023-01-18', 'Cancelled'),

(5, 5, '2023-01-19', 'Shipped'),

(6, 6, '2023-01-20', 'Processing'),

(7, 7, '2023-01-21', 'Delivered'),

(8, 8, '2023-01-22', 'Cancelled'),

(9, 9, '2023-01-23', 'Shipped'),

(10, 10, '2023-01-24', 'Processing'),

(11, 11, '2023-01-25', 'Delivered'),

(12, 12, '2023-01-26', 'Cancelled'),

(13, 13, '2023-01-27', 'Shipped'),

(14, 14, '2023-01-28', 'Processing'),

(15, 15, '2023-01-29', 'Delivered'),

(16, 16, '2023-01-30', 'Cancelled'),

(17, 17, '2023-01-31', 'Shipped'),

(18, 18, '2023-02-01', 'Processing'),

(19, 19, '2023-02-02', 'Delivered'),

(20, 20, '2023-02-03', 'Cancelled');

/\*------------------------------------------- **Products Table**------------------------------------------\*/

create table products(

product\_id int,

name varchar(20),

description varchar(30),

price int,

stock int);

INSERT INTO Products (product\_id, name, description, price, stock)

VALUES(101, 'Product A', 'Description of Product A', 19.99, 100),

(102, 'Product B', 'Description of Product B', 9.99, 150),

(103, 'Product C', 'Description of Product C', 14.99, 200),

(104, 'Product D', 'Description of Product D', 29.99, 250),

(105, 'Product E', 'Description of Product E', 24.99, 300),

(106, 'Product F', 'Description of Product F', 49.99, 350),

(107, 'Product G', 'Description of Product G', 39.99, 400),

(108, 'Product H', 'Description of Product H', 11.99, 450),

(109, 'Product I', 'Description of Product I', 7.99, 500),

(110, 'Product J', 'Description of Product J', 5.99, 550),

(111, 'Product K', 'Description of Product K', 19.99, 600),

(112, 'Product L', 'Description of Product L', 9.99, 650),

(113, 'Product M', 'Description of Product M', 14.99, 700),

(114, 'Product N', 'Description of Product N', 29.99, 750),

(115, 'Product O', 'Description of Product O', 24.99, 800),

(116, 'Product P', 'Description of Product P', 49.99, 850),

(117, 'Product Q', 'Description of Product Q', 39.99, 900),

(118, 'Product R', 'Description of Product R', 11.99, 950),

(119, 'Product S', 'Description of Product S', 7.99, 1000),

(120, 'Product T', 'Description of Product T', 5.99, 1050);

**Case Study Questions & Answers:**

**1.How to retrieve all orders made by a specific customer?**

SELECT \* FROM Orders

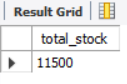
WHERE customer\_id = 1;



**2.How to get the total number of products in stock?**

SELECT SUM(stock) AS total\_stock

FROM Products;



**3. How to list all products along with their order quantities?**

SELECT Products.name, SUM(OrderItems.quantity) AS total\_ordered

FROM Products

JOIN OrderItems ON Products.product\_id = OrderItems.product\_id

GROUP BY Products.name;



**4.How to find customers who have placed orders?**

SELECT DISTINCT Customers.name, Customers.email

FROM Customers

JOIN Orders2 ON Customers.customer\_id = Orders2.customer\_id;



**5.Find the most popular product (the product with the highest quantity sold)**

SELECT p.product\_id, p.product\_name, SUM(oi.quantity) AS total\_quantity

FROM products p

JOIN orderitems oi ON p.product\_id = oi.product\_id

GROUP BY p.product\_id, p.product\_name

ORDER BY total\_quantity DESC

LIMIT 1;



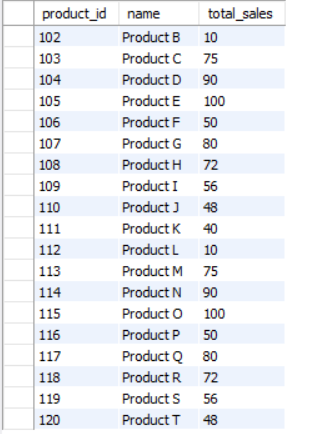
**6.Get the total sales amount for each product.**

SELECT p.product\_id, p.product\_name, SUM(oi.quantity \* oi.price) AS total\_sales

FROM products p

JOIN orderitems oi ON p.product\_id = oi.product\_id

GROUP BY p.product\_id, p.product\_name;



**7.Categorize products as 'Low Stock', 'Medium Stock', or 'High Stock' based on their stock levels**.

SELECT product\_id, name, stock,

CASE

WHEN stock < 20 THEN 'Low Stock'

WHEN stock BETWEEN 20 AND 50 THEN 'Medium Stock'

ELSE 'High Stock'

END AS Stock\_Category

FROM Products;



**8.** **Which orders have a total amount greater than $50?**

SELECT order\_id, customer\_id, order\_date, status

FROM Orders2

WHERE order\_id IN (

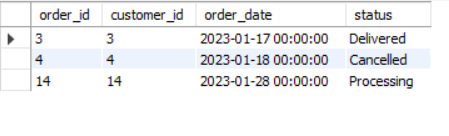
SELECT order\_id

FROM OrderItems

GROUP BY order\_id

HAVING SUM(price) > 50

);



**9.Identify if an order is 'Completed' (Shipped) or 'In Progress' (Pending).**

SELECT order\_id, customer\_id, order\_date, status,

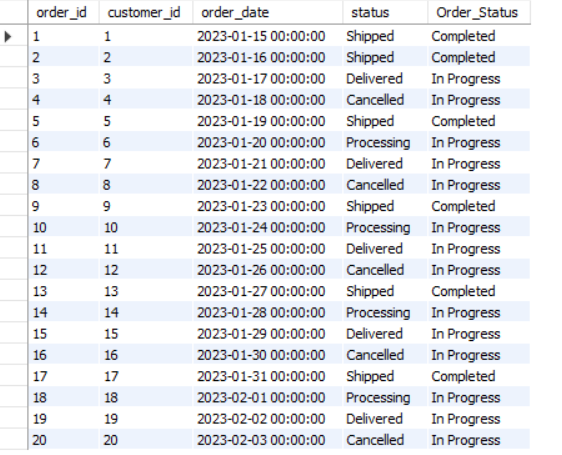
CASE

WHEN status = 'Shipped' THEN 'Completed'

ELSE 'In Progress'

END AS Order\_Status

FROM Orders2;



**10.What is the total quantity of products ordered by each customer?**

SELECT name, (

SELECT SUM(quantity)

FROM OrderItems oi

WHERE oi.order\_id IN (

SELECT order\_id

FROM Orders2 o

WHERE o.customer\_id = c.customer\_id

)

) AS total\_quantity

FROM Customers c;

