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.

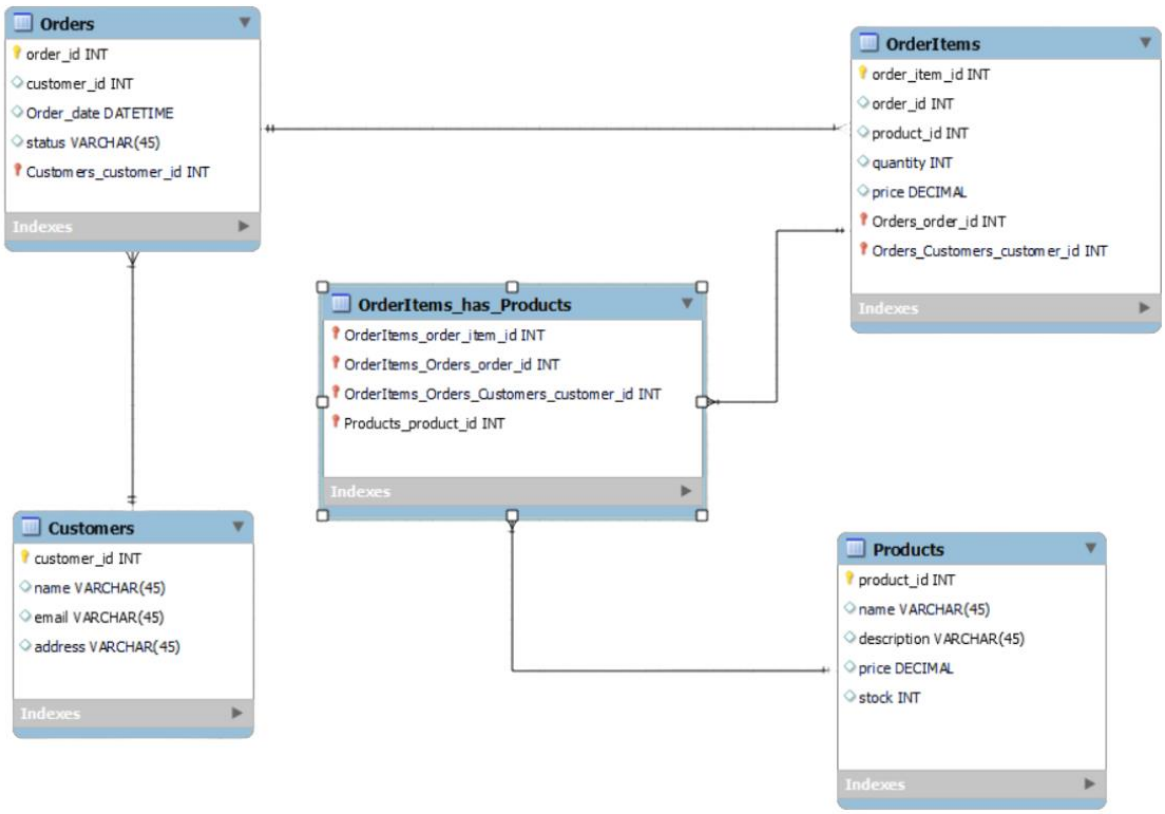
Objective:

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;
```

order_id	customer_id	order_date	status
1	1	2023-01-15 00:00:00	Shipped

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

```
SELECT SUM(stock) AS total_stock
FROM Products;
```

Result Grid

total_stock
11500

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

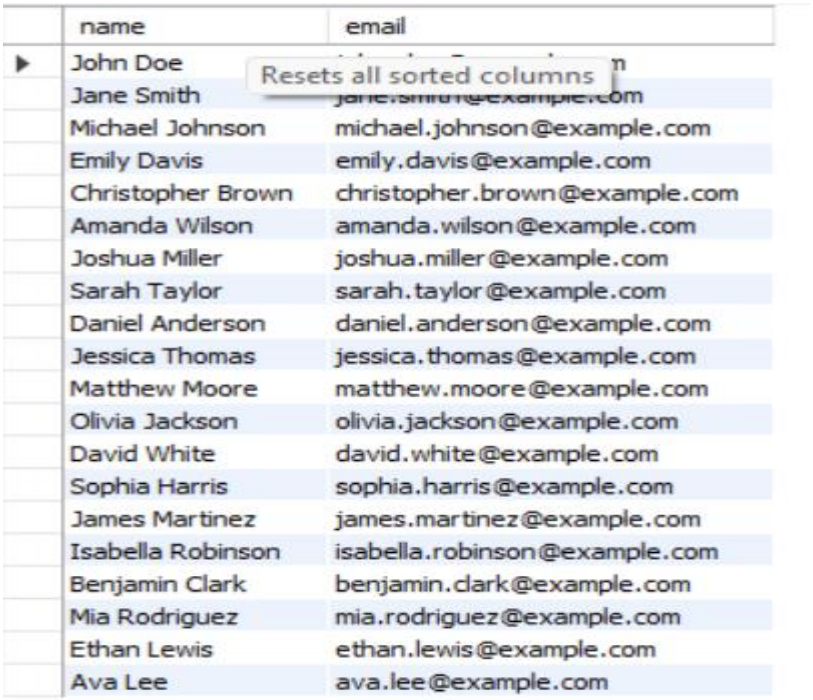
```
SELECT Products.name, SUM(OrderItems.quantity) AS total_ordered
FROM Products
JOIN OrderItems ON Products.product_id = OrderItems.product_id
GROUP BY Products.name;
```

Result Grid | Filter Rows:

name	total_ordered
Product A	2
Product B	1
Product C	5
Product D	3
Product E	4
Product F	1
Product G	2
Product H	6
Product I	7
Product J	8
Product K	2
Product L	1
Product M	5
Product N	3
Product O	4
Product P	1
Product Q	2
Product R	6
Product S	7
Product T	8

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;
```



name	email
John Doe	
Jane Smith	
Michael Johnson	michael.johnson@example.com
Emily Davis	emily.davis@example.com
Christopher Brown	christopher.brown@example.com
Amanda Wilson	amanda.wilson@example.com
Joshua Miller	joshua.miller@example.com
Sarah Taylor	sarah.taylor@example.com
Daniel Anderson	daniel.anderson@example.com
Jessica Thomas	jessica.thomas@example.com
Matthew Moore	matthew.moore@example.com
Olivia Jackson	olivia.jackson@example.com
David White	david.white@example.com
Sophia Harris	sophia.harris@example.com
James Martinez	james.martinez@example.com
Isabella Robinson	isabella.robinson@example.com
Benjamin Clark	benjamin.clark@example.com
Mia Rodriguez	mia.rodriguez@example.com
Ethan Lewis	ethan.lewis@example.com
Ava Lee	ava.lee@example.com

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;
```

	product_id	name	total_quantity
▶	110	Product J	8

6.Get the total sales amount for each product.

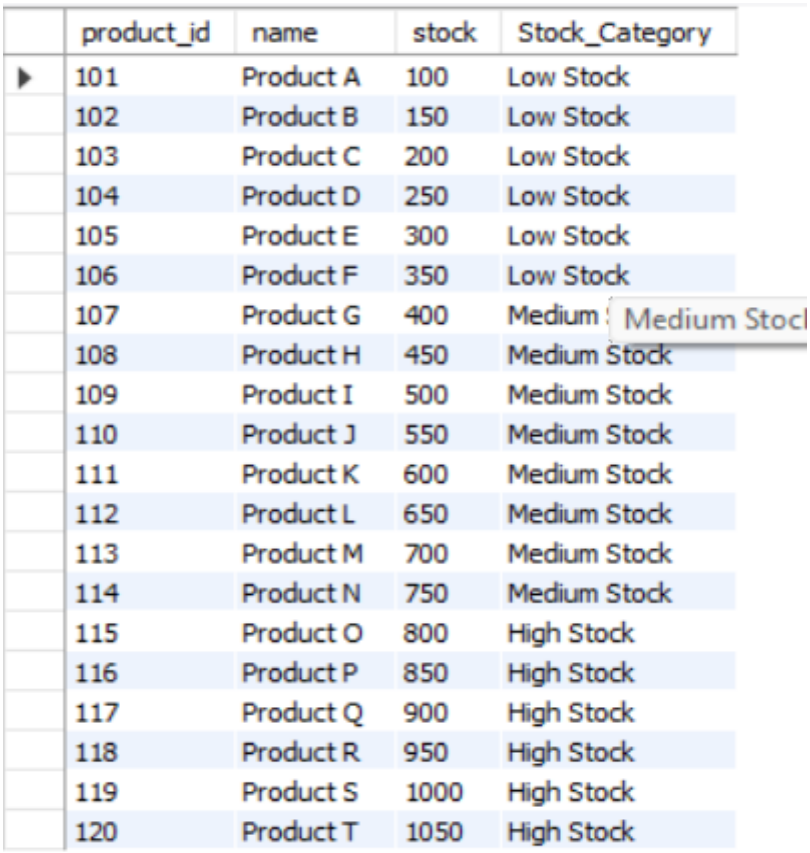
```
SELECT p.product_id, p.product_name, SUM(oi.quantity * oi.price) AS total_sales
FROM products p
JOIN orderitems oi ON p.product_id = oi.product_id
GROUP BY p.product_id, p.product_name;
```

	product_id	name	total_sales
	102	Product B	10
	103	Product C	75
	104	Product D	90
	105	Product E	100
	106	Product F	50
	107	Product G	80
	108	Product H	72
	109	Product I	56
	110	Product J	48
	111	Product K	40
	112	Product L	10
	113	Product M	75
	114	Product N	90
	115	Product O	100
	116	Product P	50
	117	Product Q	80
	118	Product R	72
	119	Product S	56
	120	Product T	48

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;
```



product_id	name	stock	Stock_Category
101	Product A	100	Low Stock
102	Product B	150	Low Stock
103	Product C	200	Low Stock
104	Product D	250	Low Stock
105	Product E	300	Low Stock
106	Product F	350	Low Stock
107	Product G	400	Medium Stock
108	Product H	450	Medium Stock
109	Product I	500	Medium Stock
110	Product J	550	Medium Stock
111	Product K	600	Medium Stock
112	Product L	650	Medium Stock
113	Product M	700	Medium Stock
114	Product N	750	Medium Stock
115	Product O	800	High Stock
116	Product P	850	High Stock
117	Product Q	900	High Stock
118	Product R	950	High Stock
119	Product S	1000	High Stock
120	Product T	1050	High Stock

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
);
```

	order_id	customer_id	order_date	status
▶	3	3	2023-01-17 00:00:00	Delivered
	4	4	2023-01-18 00:00:00	Cancelled
	14	14	2023-01-28 00:00:00	Processing

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;
```

	order_id	customer_id	order_date	status	Order_Status
▶	1	1	2023-01-15 00:00:00	Shipped	Completed
	2	2	2023-01-16 00:00:00	Shipped	Completed
	3	3	2023-01-17 00:00:00	Delivered	In Progress
	4	4	2023-01-18 00:00:00	Cancelled	In Progress
	5	5	2023-01-19 00:00:00	Shipped	Completed
	6	6	2023-01-20 00:00:00	Processing	In Progress
	7	7	2023-01-21 00:00:00	Delivered	In Progress
	8	8	2023-01-22 00:00:00	Cancelled	In Progress
	9	9	2023-01-23 00:00:00	Shipped	Completed
	10	10	2023-01-24 00:00:00	Processing	In Progress
	11	11	2023-01-25 00:00:00	Delivered	In Progress
	12	12	2023-01-26 00:00:00	Cancelled	In Progress
	13	13	2023-01-27 00:00:00	Shipped	Completed
	14	14	2023-01-28 00:00:00	Processing	In Progress
	15	15	2023-01-29 00:00:00	Delivered	In Progress
	16	16	2023-01-30 00:00:00	Cancelled	In Progress
	17	17	2023-01-31 00:00:00	Shipped	Completed
	18	18	2023-02-01 00:00:00	Processing	In Progress
	19	19	2023-02-02 00:00:00	Delivered	In Progress
	20	20	2023-02-03 00:00:00	Cancelled	In Progress

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;
```

name	total_quantity
John Doe	3
Jane Smith	5
Michael Johnson	7
Emily Davis	1
Christopher Brown	2
Amanda Wilson	6
Joshua Miller	7
Sarah Taylor	8
Daniel Anderson	2
Jessica Thomas	1
Matthew Moore	5
Olivia Jackson	3
David White	4
Sophia Harris	1
James Martinez	2
Isabella Robinson	6
Benjamin Clark	7
Mia Rodriguez	8
Ethan Lewis	NULL
Ava Lee	NULL