Our SQL implementation for a restaurant management system covers all of the essential features of a restaurant management system, including:

* Customer management
* Employee management
* Menu management
* Order management
* Reservation management
* Inventory management

We have also used a variety of SQL features in our implementation, such as:

* Foreign keys to enforce data integrity
* Joins to combine data from multiple tables
* Subqueries to filter data
* Aggregate functions to summarize data
* Transactions to ensure data consistency
* Views and procedures to make your code more reusable and efficient
* Security access control to restrict access to sensitive data

Here is a brief explanation of how our implementation works:

Database tables

We have created seven database tables:

* Customers: This table stores information about the restaurant's customers, such as their name, contact information, and dietary restrictions.
* Employees: This table stores information about the restaurant's employees, such as their name, contact information, and role.
* Menu: This table stores information about the restaurant's menu items, such as their name, description, price, and ingredients.
* Orders: This table stores information about the restaurant's orders, such as the customer's name, the order date and time, and the menu items ordered.
* OrderItems: This table stores information about the individual items in each order, such as the menu item, the quantity ordered, and the subtotal price.
* Reservations: This table stores information about the restaurant's reservations, such as the customer's name, the reservation date and time, and the table number.
* Inventory: This table stores information about the restaurant's inventory levels, such as the menu item and the quantity in stock.

SQL queries

we have also provided a variety of SQL queries to demonstrate how our implementation can be used. For example, our query to retrieve all menu items with their current stock quantity shows how you can use a view to simplify complex queries. our parameterized procedure to add a new customer to the database shows how we can make your code more reusable and efficient. And our trigger to generate audit information whenever a new order is placed shows how we can use triggers to enforce data integrity and security.

Security access control

we have also implemented basic security access control by granting different permissions to different users. For example, we have granted read permission to all users on the Menu table, but we have only granted write permission to the Manager role on the Orders table. This helps to ensure that only authorized users can access and modify sensitive data.

Overall, our SQL implementation for a restaurant management system is comprehensive, efficient, and secure.

Database diagram

