Designing a database for a café management system involves creating tables to manage the various aspects of the café’s operations, such as customers, orders, menu items, employees, and inventory. Below is a high-level outline of the database schema, including tables and their relationships.

**High-Level Functionality**

**1. Customer Management**

* **Customer Profiles**: Create, view, update, and delete customer profiles including personal information and contact details.
* **Loyalty Programs**: Enroll customers in loyalty programs, track loyalty points, and manage redemption of rewards.
* **Reservations**: Allow customers to make, view, and cancel table reservations.

**2. Order Management**

* **Order Processing**: Create, update, and cancel orders. Track order status (e.g., pending, completed, cancelled).
* **Order Details**: Add, update, or remove items from an order. Manage special instructions and preferences.
* **Payments**: Process payments using various methods (cash, credit card, mobile payments). Issue receipts.
* **Order History**: Maintain a history of all orders for each customer for reference and analysis.

**3. Menu Management**

* **Menu Items**: Add, update, and remove menu items. Include details such as name, description, price, category, and availability status.
* **Categories**: Manage categories for menu items (e.g., beverages, food, desserts).

**4. Employee Management**

* **Employee Profiles**: Create, view, update, and delete employee profiles including personal information and contact details.
* **Scheduling**: Manage employee work schedules, track shifts, and ensure adequate staffing.
* **Roles and Permissions**: Define and assign roles (e.g., Barista, Manager, Cashier) with specific permissions.

**5. Inventory Management**

* **Inventory Tracking**: Track inventory levels for all items, update quantities, and manage units of measure.
* **Reordering**: Set reorder levels and generate alerts or automatic orders when inventory falls below thresholds.
* **Supplier Management**: Manage supplier information and maintain a history of supplier orders and deliveries.

**6. Reporting and Analytics**

* **Sales Reports**: Generate daily, weekly, monthly, and yearly sales reports.
* **Inventory Reports**: Track inventory usage, current levels, and reordering needs.
* **Employee Performance**: Monitor employee performance metrics such as sales handled, hours worked, and efficiency.
* **Customer Insights**: Analyze customer behavior, popular menu items, and loyalty program effectiveness.

**Tables and Relationships**

1. **Customers**
   * **CustomerID** (Primary Key)
   * FirstName
   * LastName
   * Email
   * PhoneNumber
   * LoyaltyPoints (if a loyalty program is implemented)
2. **Employees**
   * **EmployeeID** (Primary Key)
   * FirstName
   * LastName
   * Position (e.g., Barista, Manager, Cashier)
   * Email
   * PhoneNumber
   * HireDate
   * Salary
3. **MenuItems**
   * **MenuItemID** (Primary Key)
   * Name
   * Description
   * Price
   * Category (e.g., Beverage, Food)
   * AvailabilityStatus
4. **Orders**
   * **OrderID** (Primary Key)
   * CustomerID (Foreign Key)
   * EmployeeID (Foreign Key)
   * OrderDate
   * TotalAmount
   * PaymentMethod
   * Status (e.g., Completed, Pending, Cancelled)
5. **OrderDetails**
   * **OrderDetailID** (Primary Key)
   * OrderID (Foreign Key)
   * MenuItemID (Foreign Key)
   * Quantity
   * ItemPrice
6. **Inventory**
   * **InventoryID** (Primary Key)
   * ItemName
   * Quantity
   * Unit (e.g., kg, liter)
   * ReorderLevel
   * SupplierID (Foreign Key)
7. **Suppliers**
   * **SupplierID** (Primary Key)
   * SupplierName
   * ContactPerson
   * PhoneNumber
   * Email
   * Address
8. **Payments**
   * **PaymentID** (Primary Key)
   * OrderID (Foreign Key)
   * Amount
   * PaymentDate
   * PaymentMethod (e.g., Cash, Credit Card, Mobile Payment)

**Relationships**

* **Customers** to **Orders**: One-to-Many (One customer can place many orders)
* **Employees** to **Orders**: One-to-Many (One employee can handle many orders)
* **Orders** to **OrderDetails**: One-to-Many (One order can have many order details)
* **MenuItems** to **OrderDetails**: One-to-Many (One menu item can appear in many order details)
* **Suppliers** to **Inventory**: One-to-Many (One supplier can provide many inventory items)
* **Orders** to **Payments**: One-to-One (Each order has one payment record)

**SQL Schema**

Here is a basic SQL schema for the above tables:

CREATE TABLE Customers (

CustomerID INT PRIMARY KEY AUTO\_INCREMENT,

FirstName VARCHAR(50),

LastName VARCHAR(50),

Email VARCHAR(100),

PhoneNumber VARCHAR(15),

LoyaltyPoints INT DEFAULT 0

);

CREATE TABLE Employees (

EmployeeID INT PRIMARY KEY AUTO\_INCREMENT,

FirstName VARCHAR(50),

LastName VARCHAR(50),

Position VARCHAR(50),

Email VARCHAR(100),

PhoneNumber VARCHAR(15),

HireDate DATE,

Salary DECIMAL(10, 2)

);

CREATE TABLE MenuItems (

MenuItemID INT PRIMARY KEY AUTO\_INCREMENT,

Name VARCHAR(100),

Description TEXT,

Price DECIMAL(10, 2),

Category VARCHAR(50),

AvailabilityStatus BOOLEAN DEFAULT TRUE

);

CREATE TABLE Orders (

OrderID INT PRIMARY KEY AUTO\_INCREMENT,

CustomerID INT,

EmployeeID INT,

OrderDate DATETIME,

TotalAmount DECIMAL(10, 2),

PaymentMethod VARCHAR(50),

Status VARCHAR(50),

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID),

FOREIGN KEY (EmployeeID) REFERENCES Employees(EmployeeID)

);

CREATE TABLE OrderDetails (

OrderDetailID INT PRIMARY KEY AUTO\_INCREMENT,

OrderID INT,

MenuItemID INT,

Quantity INT,

ItemPrice DECIMAL(10, 2),

FOREIGN KEY (OrderID) REFERENCES Orders(OrderID),

FOREIGN KEY (MenuItemID) REFERENCES MenuItems(MenuItemID)

);

CREATE TABLE Inventory (

InventoryID INT PRIMARY KEY AUTO\_INCREMENT,

ItemName VARCHAR(100),

Quantity DECIMAL(10, 2),

Unit VARCHAR(20),

ReorderLevel DECIMAL(10, 2),

SupplierID INT,

FOREIGN KEY (SupplierID) REFERENCES Suppliers(SupplierID)

);

CREATE TABLE Suppliers (

SupplierID INT PRIMARY KEY AUTO\_INCREMENT,

SupplierName VARCHAR(100),

ContactPerson VARCHAR(100),

PhoneNumber VARCHAR(15),

Email VARCHAR(100),

Address TEXT

);

CREATE TABLE Payments (

PaymentID INT PRIMARY KEY AUTO\_INCREMENT,

OrderID INT,

Amount DECIMAL(10, 2),

PaymentDate DATETIME,

PaymentMethod VARCHAR(50),

FOREIGN KEY (OrderID) REFERENCES Orders(OrderID)

);

ER Diagram