CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

UNIVERSITY INSTITUTE OF COMPUTING

Project On
Database Management System

Food Delivery System Project

Program Name: BCA
Subject Name/Code: Database Management System (23CAT-251)

Submitted by:

Name: Kamaljeet Kaur

UID: 23BCA10385

Section: BCA – 7

Group: B

Submitted to:

Name: Mrs. Sonia

1. Aim

To design and implement a database for an online food delivery system that manages restaurants, customers, orders, and delivery personnel efficiently.

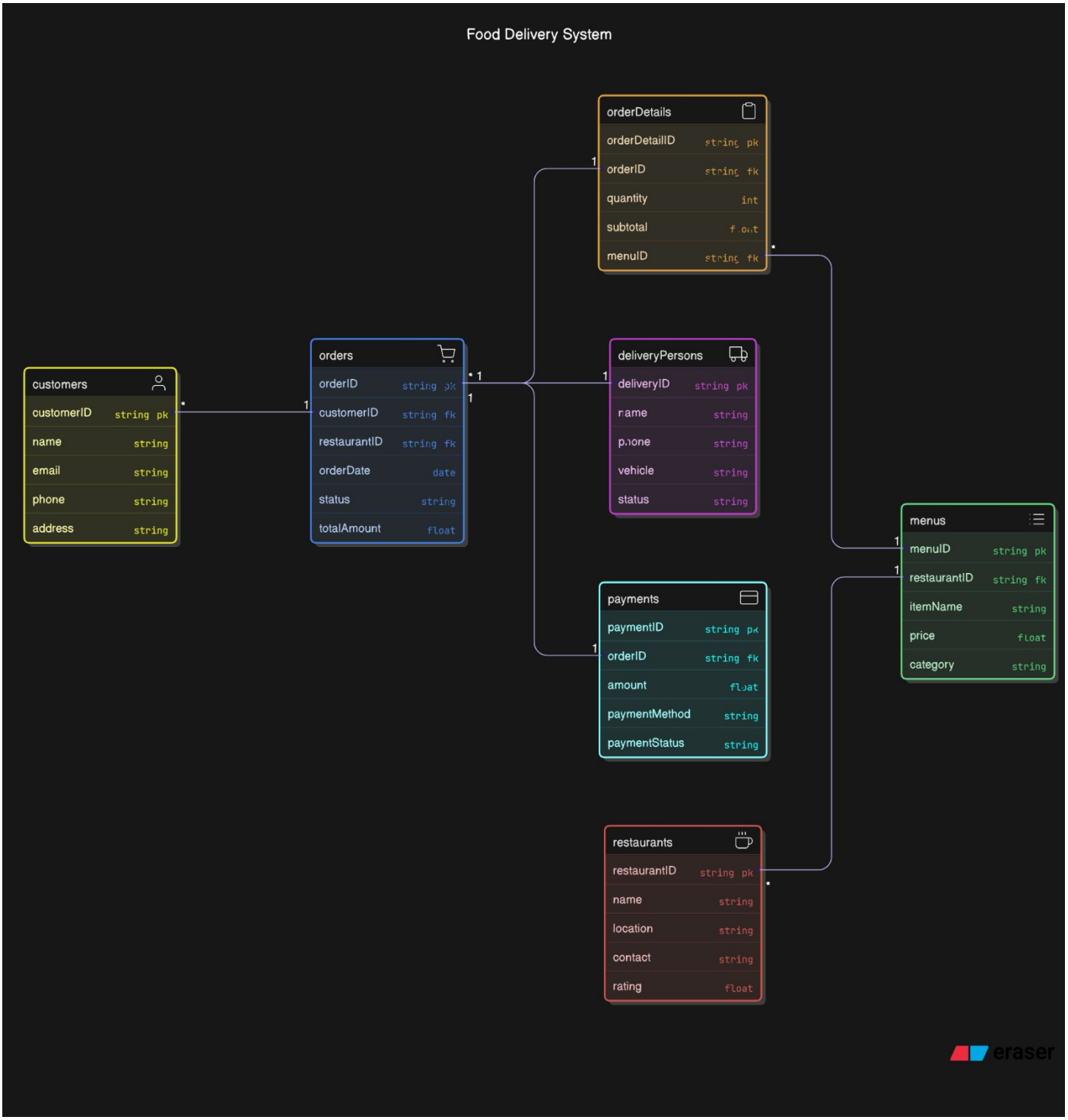
2. Overview

The system allows customers to browse restaurants, place orders, track deliveries, and make payments. Restaurants manage their menus and orders, while delivery personnel handle order dispatch and delivery.

3. ER Diagram

- **Entities:**
  - **Customer** (CustomerID, Name, Email, Phone, Address)
  - **Restaurant** (RestaurantID, Name, Location, Contact, Rating)
  - **Menu** (MenuID, RestaurantID, ItemName, Price, Category)
  - **Order** (OrderID, CustomerID, RestaurantID, OrderDate, Status, TotalAmount)
  - **OrderDetails** (OrderDetailID, OrderID, MenuID, Quantity, Subtotal)
  - **DeliveryPerson** (DeliveryID, Name, Phone, Vehicle, Status)
  - **Payment** (PaymentID, OrderID, Amount, PaymentMethod, PaymentStatus)
- **Relationships:**
  - A **Customer** places multiple **Orders**.
  - A **Restaurant** has multiple **Menu** items.

- An **Order** contains multiple **OrderDetails**.
- A **DeliveryPerson** delivers an **Order**.
- An **Order** has one **Payment**.



4. Table Schema

Customer Table

```
CREATE TABLE Customer (
    CustomerID INT PRIMARY KEY,
    Name VARCHAR(100),
    Email VARCHAR(100) UNIQUE,
    Phone VARCHAR(15),
    Address TEXT
);
```

Restaurant Table

```
CREATE TABLE Restaurant (
    RestaurantID INT PRIMARY KEY,
    Name VARCHAR(100),
    Location TEXT,
    Contact VARCHAR(15),
    Rating DECIMAL(2,1)
);
```

Menu Table

```
CREATE TABLE Menu (
    MenuID INT PRIMARY KEY,
    RestaurantID INT,
    ItemName VARCHAR(100),
    Price DECIMAL(10,2),
    Category VARCHAR(50),
    FOREIGN KEY (RestaurantID) REFERENCES Restaurant(RestaurantID)
);
```

Order Table

```
CREATE TABLE Orders (
    OrderID INT PRIMARY KEY,
    CustomerID INT,
    RestaurantID INT,
    OrderDate DATETIME,
    Status VARCHAR(20),
    TotalAmount DECIMAL(10,2),
    FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID),
    FOREIGN KEY (RestaurantID) REFERENCES Restaurant(RestaurantID)
);
```

OrderDetails Table

```
CREATE TABLE OrderDetails (
    OrderDetailID INT PRIMARY KEY,
    OrderID INT,
    MenuID INT,
    Quantity INT,
    Subtotal DECIMAL(10,2),
    FOREIGN KEY (OrderID) REFERENCES Orders(OrderID),
    FOREIGN KEY (MenuID) REFERENCES Menu(MenuID)
);
```

DeliveryPerson Table

```
CREATE TABLE DeliveryPerson (
    DeliveryID INT PRIMARY KEY,
    Name VARCHAR(100),
    Phone VARCHAR(15),
    Vehicle VARCHAR(50),
    Status VARCHAR(20)
);
```

Payment Table

```
CREATE TABLE Payment (
    PaymentID INT PRIMARY KEY,
    OrderID INT,
    Amount DECIMAL(10,2),
    PaymentMethod VARCHAR(50),
    PaymentStatus VARCHAR(20),
    FOREIGN KEY (OrderID) REFERENCES Orders(OrderID)
);
```

Entering Sample Data

```
INSERT INTO Customer VALUES (1, 'John Doe', 'john@example.com', '9876543210', '123 Main St');
INSERT INTO Restaurant VALUES (1, 'Pizza Palace', 'Downtown', '9876543211', 4.5);
INSERT INTO Menu VALUES (1, 1, 'Margherita Pizza', 8.99, 'Pizza');
INSERT INTO Orders VALUES (1, 1, 1, '2025-03-28 12:30:00', 'Pending', 8.99);
INSERT INTO OrderDetails VALUES (1, 1, 1, 1, 8.99);
INSERT INTO DeliveryPerson VALUES (1, 'Alex Rider', '9876543222', 'Bike', 'Available');
INSERT INTO Payment VALUES (1, 1, 8.99, 'Credit Card', 'Paid');
```

6. Summary

This database design efficiently handles food orders, restaurant menus, customer management, payments, and delivery personnel. The relationships ensure data integrity and efficient retrieval for a seamless food delivery experience.

7. Conclusion :

This **Food Delivery System DBMS** project design provides a **simplified structure** with **essential relationships** and **tables**. The **ER diagram** clearly illustrates how entities like **Customer**, **Restaurant**, **Order**, **Menu**, **Payment**, and **Delivery** are related to each other.

- **Tables** are designed to minimize complexity while ensuring core functionality.
- The **ER diagram** clearly shows the relationships, including **one-to-many** and **many-to-one** relationships between entities.
- **SQL queries** are provided for the most common actions, such as **placing orders**, **updating order statuses**, and **marking deliveries**.

This simplified design gives you a solid foundation for building a **Food Delivery System** while avoiding unnecessary complexity.