Hackthon 2st Online food ordering system

```
- - - Create Table

CREATE DATABASE FoodOrderingDB;
USE FoodOrderingDB;

- - - User Table

CREATE TABLE Users (
   user_id INT AUTO_INCREMENT PRIMARY KEY,
   name VARCHAR(100) NOT NULL,
   email VARCHAR(100) UNIQUE NOT NULL,
   password_hash VARCHAR(255) NOT NULL,
   phone VARCHAR(15),
   address TEXT,
   created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

- - - Restaurants Table

CREATE TABLE Restaurants (
   restaurant_id INT AUTO_INCREMENT PRIMARY KEY,
   name VARCHAR(100) NOT NULL,
   location VARCHAR(255),
   contact VARCHAR(20),
   opening_hours VARCHAR(50),
   created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

```
--- Menu Items Table

CREATE TABLE Menu_Items (
    item_id INT AUTO_INCREMENT PRIMARY KEY,
    restaurant_id INT,
    name VARCHAR(100) NOT NULL,
    description TEXT,
    price DECIMAL(10,2) NOT NULL,
    availability BOOLEAN DEFAULT TRUE,
    FOREIGN KEY (restaurant_id) REFERENCES Restaurants(restaurant_id) ON
DELETE CASCADE
);

--- Order Table

CREATE TABLE Orders (
    order_id INT AUTO_INCREMENT PRIMARY KEY,
    user_id INT,
    restaurant_id INT,
    order_status ENUM('Pending', 'Preparing', 'Out for Delivery', 'Delivered',
'Cancelled') DEFAULT 'Pending',
    total_amount DECIMAL(10,2),
    order_time TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (user_id) REFERENCES Users(user_id) ON DELETE CASCADE,
    FOREIGN KEY (restaurant_id) REFERENCES Restaurants(restaurant_id) ON
DELETE CASCADE
);
--- Order_Items Table

CREATE TABLE Order_Items (
    order_item_id INT AUTO_INCREMENT PRIMARY KEY,
    order_id INT,
    item_id INT,
    quantity INT NOT NULL,
    price DECIMAL(10,2) NOT NULL,
    FOREIGN KEY (order_id) REFERENCES Orders(order_id) ON DELETE CASCADE,
    FOREIGN KEY (item_id) REFERENCES Menu_Items(item_id) ON
DELETE CASCADE
);

--- Payments Table

CREATE TABLE Payments (
    payment_id INT AUTO_INCREMENT PRIMARY KEY,
    order_id INT,
    payment_method ENUM('Credit Card', 'Debit Card', 'UPI', 'Cash on Delivery',
'Wallet'),
    payment_status ENUM('Pending', 'Completed', 'Failed') DEFAULT 'Pending',
    transaction_id VARCHAR(255),
    amount DECIMAL(10,2),
    payment_time TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (order_id) REFERENCES Orders(order_id) ON DELETE CASCADE
);

--- Delivery Table

CREATE TABLE Delivery (
    delivery_id INT AUTO_INCREMENT PRIMARY KEY,
    order_id INT,
    delivery_status ENUM('Pending', 'On the Way', 'Delivered') DEFAULT 'Pending',
    delivery_agent VARCHAR(100),
    estimated_time TIME,
    FOREIGN KEY (order_id) REFERENCES Orders(order_id) ON DELETE CASCADE
);
```

```
- - - Insert Users

INSERT INTO Users (name, email, password_hash, phone, address)
VALUES ('John Doe', 'john@example.com', 'hashedpassword123', '9876543210',
'123 Street, City');

- - - Insert Restaurants

INSERT INTO Restaurants (name, location, contact, opening_hours)
VALUES ('Pizza Palace', 'Downtown', '9876543210', '10:00 AM - 11:00 PM');

- - - Insert Menu Items

INSERT INTO Menu_Items (restaurant_id, name, description, price, availability)
VALUES (1, 'Margherita Pizza', 'Classic cheese pizza', 8.99, TRUE);

- - - Insert Orders

INSERT INTO Orders (user_id, restaurant_id, order_status, total_amount)
VALUES (1, 1, 'Pending', 20.50);



- - - Insert Order items

INSERT INTO Order_Items (order_id, item_id, quantity, price)
VALUES (1, 1, 2, 17.98);

- - - Insert Payment Details

INSERT INTO Payments (order_id, payment_method, payment_status,
transaction_id, amount)
VALUES (1, 'UPI', 'Completed', 'TXN12345UPI', 20.50);

--- Insert Delivery Details

INSERT INTO Delivery (order_id, delivery_status, delivery_agent, estimated_time)
VALUES (1, 'On the Way', 'David', '00:30:00');

--- Retrieve All Users

SELECT * FROM Users;

--- Get All Menu Items For a Specific Restaurant

SELECT name, description, price FROM Menu_Items WHERE restaurant_id = 1;



--- Retrieve All Orders With User Details
SELECT Orders.order_id, Users.name, Users.email, Orders.order_status,
Orders.total_amount, Orders.order_time
FROM Orders
JOIN Users ON Orders.user_id = Users.user_id;



--- Check Order Stats and Payment Details

SELECT Orders.order_id, Orders.order_status, Payments.payment_status,
Payments.payment_method
FROM Orders
JOIN Payments ON Orders.order_id = Payments.order_id
WHERE Orders.order_id = 1;
```

```sql
- - - Find All Pending Deliveries

SELECT Delivery.order_id, Users.name AS Customer, Delivery.delivery_status, Delivery.delivery_agent

FROM Delivery

JOIN Orders ON Delivery.order_id = Orders.order_id

JOIN Users ON Orders.user_id = Users.user_id

WHERE Delivery.delivery_status = 'Pending';
```