**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;

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';