**MySQL Project: Zomato Database**

**Step 1: Database Schema Design**

Create a database named zomato:

CREATE DATABASE zomato;

USE zomato;

**Step 2: Creating Tables**

**1. Users Table**

* Stores information about the users of the app.

CREATE TABLE Users (

user\_id INT PRIMARY KEY AUTO\_INCREMENT,

username VARCHAR(50) NOT NULL,

email VARCHAR(100) NOT NULL UNIQUE,

phone VARCHAR(15),

address TEXT

);

**2. Restaurants Table**

* Stores details of the restaurants listed on the platform.

CREATE TABLE Restaurants (

restaurant\_id INT PRIMARY KEY AUTO\_INCREMENT,

name VARCHAR(100) NOT NULL,

location VARCHAR(100),

rating FLOAT CHECK (rating >= 0 AND rating <= 5),

cuisine VARCHAR(50),

contact VARCHAR(15)

);

**3. Menu\_Items Table**

* Stores information about the menu items offered by each restaurant.

CREATE TABLE Menu\_Items (

item\_id INT PRIMARY KEY AUTO\_INCREMENT,

restaurant\_id INT,

item\_name VARCHAR(100) NOT NULL,

price DECIMAL(8, 2) NOT NULL,

description TEXT,FOREIGN KEY (restaurant\_id) REFERENCES Restaurants(restaurant\_id) ON DELETE CASCADE

);

**4. Orders Table**

* Manages information about customer orders.

CREATE TABLE Orders (

order\_id INT PRIMARY KEY AUTO\_INCREMENT,

user\_id INT,

restaurant\_id INT,

order\_date DATETIME DEFAULT CURRENT\_TIMESTAMP,

status ENUM('Pending', 'Completed', 'Cancelled') DEFAULT 'Pending',

total\_price DECIMAL(8, 2),

FOREIGN KEY (user\_id) REFERENCES Users(user\_id) ON DELETE CASCADE,

FOREIGN KEY (restaurant\_id) REFERENCES Restaurants(restaurant\_id) ON DELETE CASCADE

);

**5. Order\_Items Table**

* Manages individual items in an order (many-to-many relationship between Orders and Menu\_Items).

CREATE TABLE Order\_Items (

order\_item\_id INT PRIMARY KEY AUTO\_INCREMENT,

order\_id INT,

item\_id INT,

quantity INT DEFAULT 1,

price DECIMAL(8, 2),

FOREIGN KEY (order\_id) REFERENCES Orders(order\_id) ON DELETE CASCADE,

FOREIGN KEY (item\_id) REFERENCES Menu\_Items(item\_id) ON DELETE CASCADE

);

**6. Reviews Table**

* Stores reviews from users for restaurants.

CREATE TABLE Reviews (

review\_id INT PRIMARY KEY AUTO\_INCREMENT,

user\_id INT,

restaurant\_id INT,

rating INT CHECK (rating >= 0 AND rating <= 5),

comments TEXT,

review\_date DATETIME 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

);

**Step 3: Inserting Sample Data**

Add sample data to the tables to test functionality.

**Users Table**

**Syntax:**

INSERT INTO Users (username, email, phone, address)

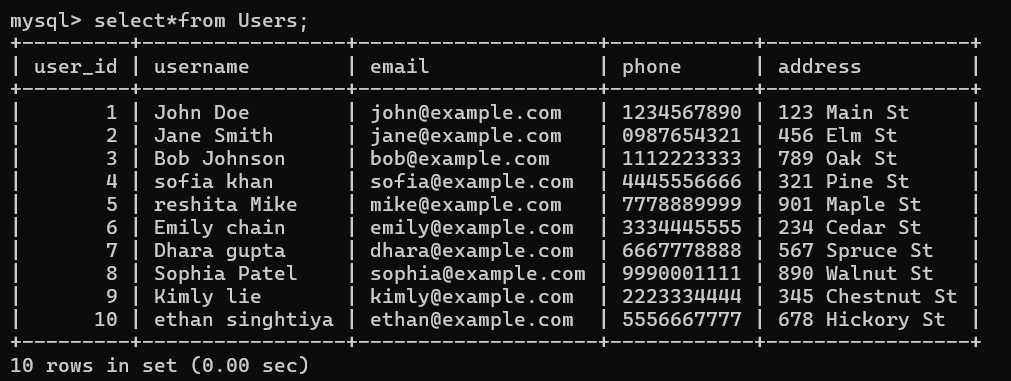
VALUES

('John Doe', 'john@example.com', '1234567890', '123 Main St'),

('Jane Smith', 'jane@example.com', '0987654321', '456 Elm St'),

(‘Bob johnson’,’bob@example.com’,’1112223333’,’789Oak St’);

Output:



**Restaurants Table**

**Syntax:**

INSERT INTO Restaurants (name, location, rating, cuisine, contact)

VALUES

(‘Burger King’,’New York, NY’,4.5,’American’,’8663942493’),

(‘Taco Bell’,’Los Angeles, CA’,4.8,’Mexican’,’8008226235)’

(‘McDonalds’,’Chicago IL’,4.2,’American’,’8002446227’);

Output:



**Menu\_Items Table**

**Syntax:**

INSERT INTO Menu\_Items (restaurant\_id, item\_name, price, description) VALUES

(1, 'Margherita Pizza', 9.99, 'Classic cheese and tomato pizza'),

(1, 'Pepperoni Pizza', 12.99, 'Pepperoni pizza with mozzarella'),

(2, 'Salmon Sushi', 15.99, 'Fresh salmon nigiri sushi'),

(2, 'Tuna Sushi', 14.99, 'Fresh tuna nigiri sushi');

(3,’Crunchy Taco’,2.99,’Crunchy taco with beef and lettuce’),

(3,’Nachos BellGrande’,8.99,’Nachose with beef,cheese,and salsa’);

Output:



**Orders Table**

Syntax:

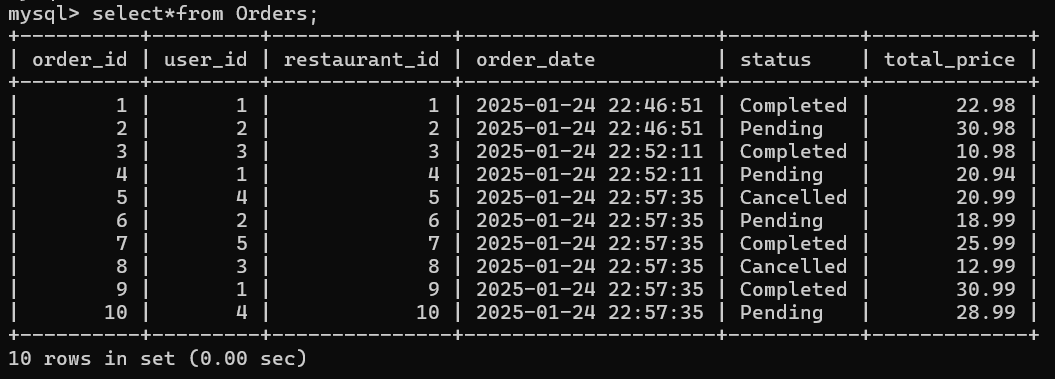
INSERT INTO Orders (user\_id, restaurant\_id, total\_price, status)

VALUES(3, 3, 10.98, 'Completed'),

(1, 4, 20.94, 'Pending'),

(4, 5, 20.99, ’Cancelled’);

**Output:**



**Order\_Items Table**

**Syntax:**

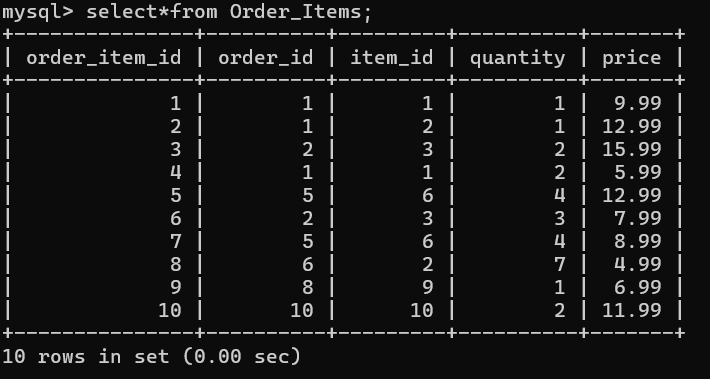
INSERT INTO Order\_Items (order\_id, item\_id, quantity, price)

VALUES(1, 1, 1, 9.99),

(1, 2, 1, 12.99),

(2, 3, 2, 15.99),

Output:



**Reviews Table**

Syntax:

INSERT INTO Reviews (user\_id, restaurant\_id, rating, comments)

VALUES (1, 1, 5, 'Amazing pizza! Highly recommend.'),

(2, 2, 4, 'Great sushi, but a bit pricey.');

(1, 1, 4, 'Great food and service');

(1, 4, 4, 'Love their coffe!');

**Output:**



**Step 4: Writing Queries**

1. **Get all restaurants and their average rating:**

**Syntax:**

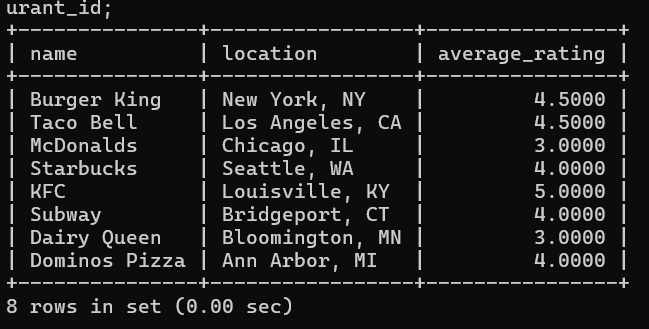
SELECT r.name, r.location, AVG(re.rating) AS average\_rating

FROM Restaurants r

JOIN Reviews re ON r.restaurant\_id = re.restaurant\_id

GROUP BY r.restaurant\_id;

**Output:**



**2. Find all menu items of a restaurant:**

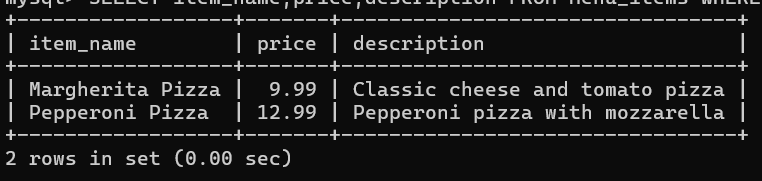
**Syntax:**

SELECT item\_name, price, description

FROM Menu\_Items

WHERE restaurant\_id = 1;

Output:



1. **Retrieve all orders by a user:**

**Syntax:**

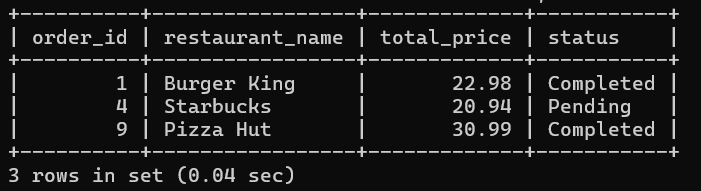
SELECT Orders.order\_id, Restaurants.name AS restaurant\_name, Orders.total\_price, Orders.status

FROM Orders

JOIN Restaurants ON Orders.restaurant\_id = Restaurants.restaurant\_id

WHERE Orders.user\_id = 1;

Output:



**4. Calculate total earnings for each restaurant:**

**Syntax:**

SELECT Restaurants.name, SUM(Orders.total\_price) AS total\_earnings

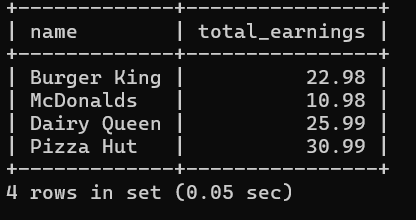
FROM Orders

JOIN Restaurants ON Orders.restaurant\_id = Restaurants.restaurant\_id

WHERE Orders.status = 'Completed'

GROUP BY Restaurants.restaurant\_id;

**Output:**



1. **Display all reviews for a restaurant:**

**Syntax:**

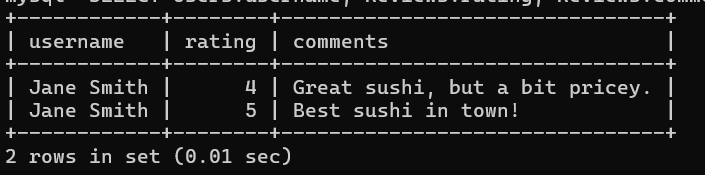
SELECT Users.username, Reviews.rating, Reviews.comments

FROM Reviews

JOIN Users ON Reviews.user\_id = Users.user\_id

WHERE restaurant\_id = 2;

**Output:**



1. **Find the most expensive menu item in each restaurant**

**Syntax:**

SELECT r.name, mi.item\_name, mi.price

FROM Restaurants r

JOIN Menu\_Items mi ON r.restaurant\_id = mi.restaurant\_id

WHERE (r.restaurant\_id, mi.price) IN (

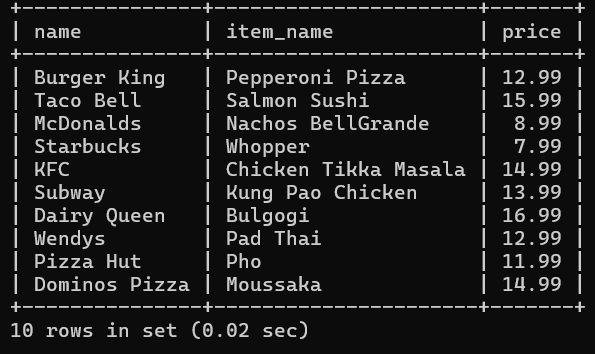
SELECT restaurant\_id, MAX(price)

FROM Menu\_Items

GROUP BY restaurant\_id

);

**Output:**



1. **Get all users who have ordered from a specific restaurant**

**Syntax:**

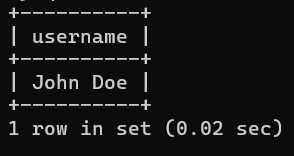
SELECT DISTINCT u.username

FROM Users u

JOIN Orders o ON u.user\_id = o.user\_id

WHERE o.restaurant\_id = 1;

**Output:**



1. **Calculate the total number of orders for each user**

**Syntax:**

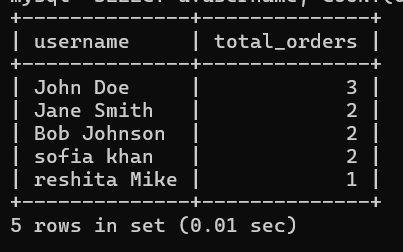
SELECT u.username, COUNT(o.order\_id) AS total\_orders

FROM Users u

JOIN Orders o ON u.user\_id = o.user\_id

GROUP BY u.user\_id;

**Output:**



**9.Display all menu items that have been ordered by a specific user**

**Syntax:**

SELECT DISTINCT mi.item\_name

FROM Menu\_Items mi

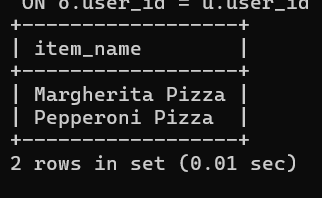
JOIN Order\_Items oi ON mi.item\_id = oi.item\_id

JOIN Orders o ON oi.order\_id = o.order\_id

JOIN Users u ON o.user\_id = u.user\_id

WHERE u.username = 'John Doe';

**Output:**



1. **Get the average rating for each cuisine**

**Syntax:**

SELECT r.cuisine, AVG(re.rating) AS average\_rating

FROM Restaurants r

JOIN Reviews re ON r.restaurant\_id = re.restaurant\_id

GROUP BY r.cuisine;

**Output:**

