Title: Food Ordering And Delivery Management System

Course: DATABASE MANAGEMENT SYSTEM (UE23CS351A)
Project Level: Experiential Learning: Level 2 (Mini Project)
Team Members:

- Student 1: SIRI S ARADHYA — PES1UG23CS906
- Student 2: YOGITHA A S — PES1UG23CS901

Institution: PES University

Submission Date: 1st NOV 20205

1. DESCRIPTION

Abstract: This project implements a Food Ordering System using MySQL as backend and Streamlit as the frontend. The system supports user registration/login (hashed passwords), restaurant browsing, menu management, cart functionality, order placement (stored procedure), automatic inventory updates (triggers), payment tracking, and user reviews. The application demonstrates normalized relational design, stored procedures, triggers, functions, and analytical queries—meeting the DBMS mini-project requirements.

2. User Requirement Specification

2.1 Purpose of the Project

The purpose of this project is to design and develop a Food Ordering System that enables users to browse restaurants, view menus, place orders, and provide feedback, while allowing administrators to manage restaurants, menu items, and customer orders efficiently.

This project aims to simulate the functionality of popular food delivery platforms like Swiggy or Zomato, providing an interactive web-based platform integrated with a relational database. It demonstrates how database concepts such as normalization, referential integrity, stored procedures, and triggers can be implemented effectively in a real-world business scenario.

The system provides an end-to-end food ordering experience — from restaurant browsing and cart management to order placement and payment tracking — all supported by robust backend database operations.

2.2 Scope of the Project

The scope of this project covers both user-facing and administrator-facing functionalities.

- **For Users:**
  The system allows registration and secure login using password hashing, browsing multiple restaurants and their menus, adding desired food items to a cart, placing orders, selecting payment methods, and submitting reviews with ratings.
- **For Administrators:**
  The system provides a dashboard to manage restaurants, update or delete menu items, track order progress, and update order statuses.

The project ensures data consistency and automation using triggers, procedures, and functions. It demonstrates complete CRUD (Create, Read, Update, Delete) operations across all entities and enforces business rules such as inventory control and order status history.

This application is scalable and can be extended further with delivery tracking, user analytics, and coupon-based discount systems, making it a comprehensive database-driven business solution.

2.3 Detailed Description of the Project

The Food Ordering System is implemented using MySQL as the backend database and Streamlit (Python) as the frontend interface.
It integrates multiple interrelated tables that represent real-world entities such as users, restaurants, menu items, orders, payments, and reviews.

- The database schema includes tables: Users, Restaurants, Menu, Orders, Order_Items, Cart, Payments, Coupons, Delivery_Partners, Reviews, and Order_Status_History.
- The database is normalized and uses foreign key constraints to maintain referential integrity among entities.
- Stored Procedures automate key operations such as order placement and review submission (PlaceOrderFromCart, AddReview).
- Functions such as GetOrderTotal and GetRestaurantAvgRating help compute derived data for business insights.
- Triggers handle automatic updates — adjusting stock levels after each order, recalculating totals, and maintaining a detailed log of order status changes.
- The Streamlit frontend provides a graphical user interface with interactive features for both customers and administrators. Users can browse restaurants with images, view menu categories, and interactively manage their cart, while admins can view and update data seamlessly.

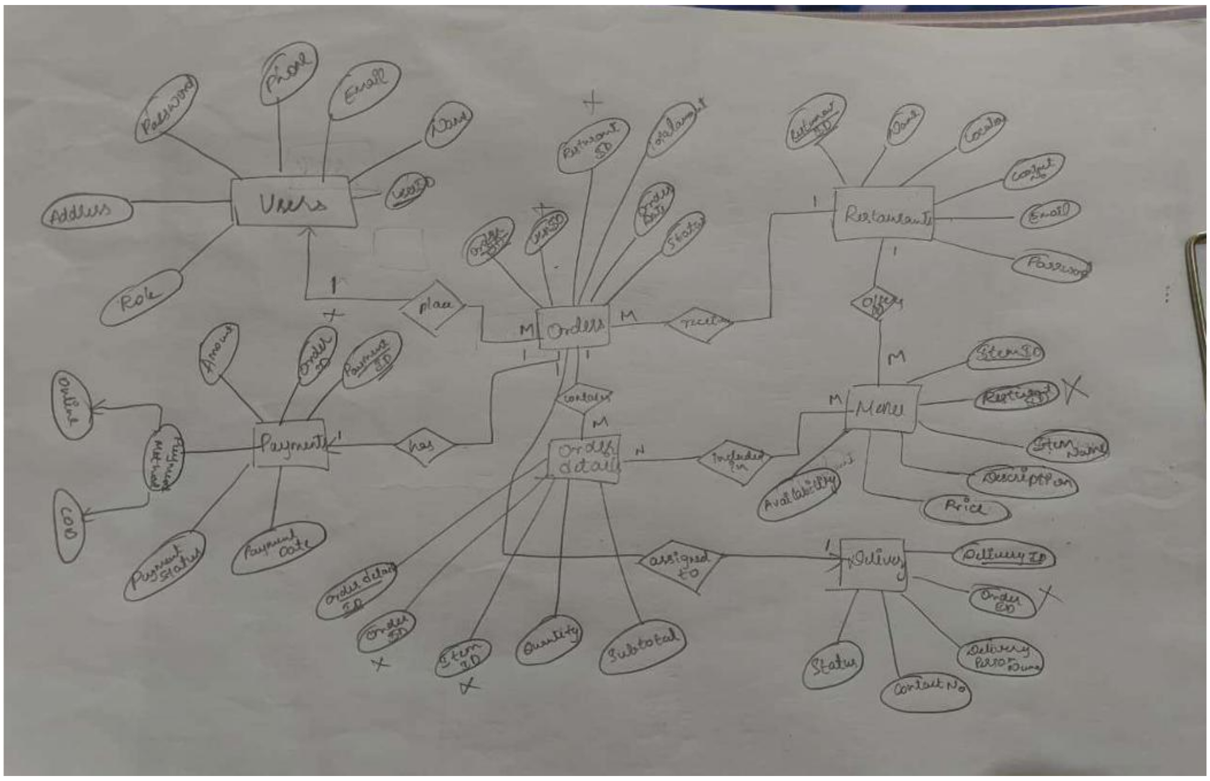
The system thus demonstrates a complete data flow from front-end operations to backend database transactions, ensuring accuracy, security, and ease of management.

2.4 Functional Requirements

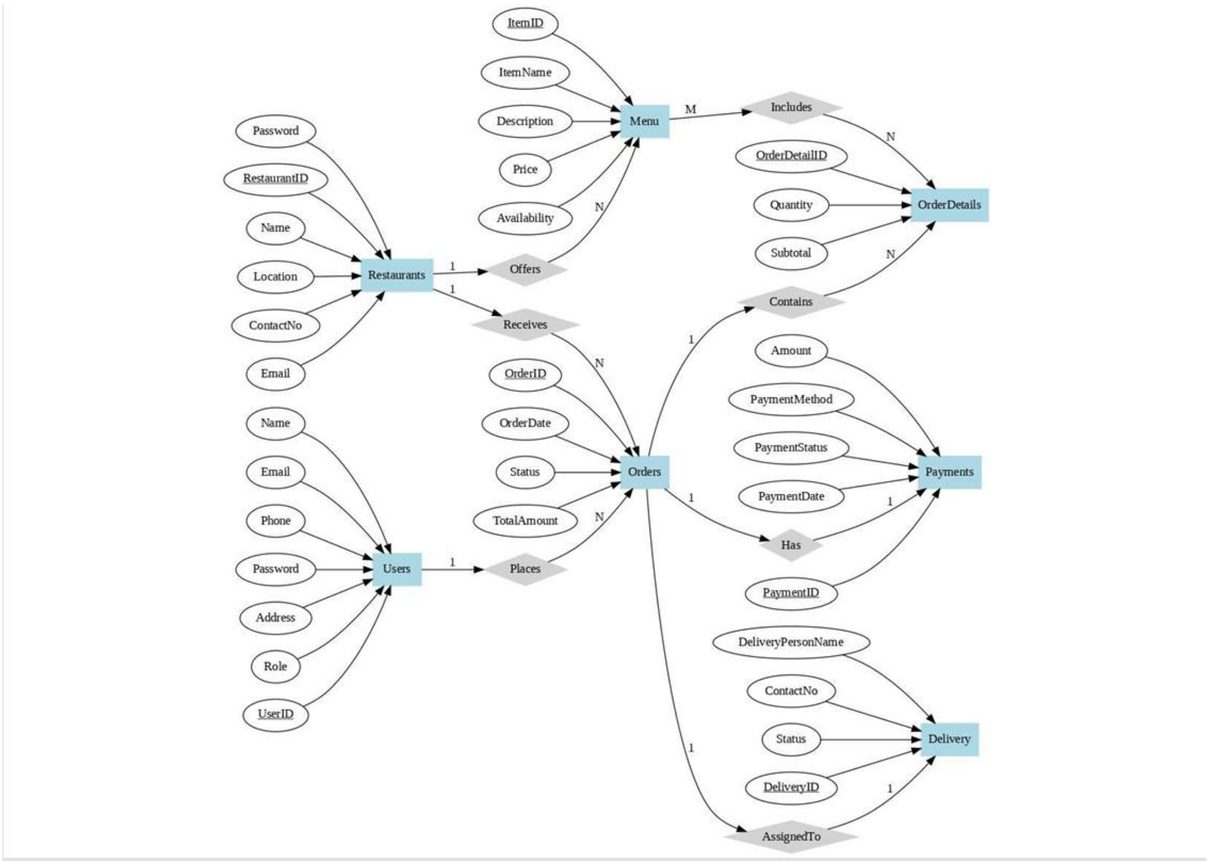
Sl. No.	System Functionality	Description
1	**User Registration & Login**	Allows users to sign up with name, email, and password. Passwords are hashed using SHA-256 for secure authentication.
2	**Restaurant Browsing**	Displays a list of restaurants with details and images. Users can select a restaurant to view its menu.
3	**Menu Display & Filtering**	Fetches menu items dynamically from the database and categorizes them (e.g., Pizza, Drinks, Desserts).
4	**Add to Cart**	Enables users to add menu items to their cart with a specified quantity. Duplicate entries are restricted by unique constraints.

5	**View / Modify Cart**	Displays all items in the user’s cart with quantities, prices, and totals. Allows removal of items.
6	**Place Order**	Executes stored procedure PlaceOrderFromCart to create an order, move items from the cart, and compute total amount.
7	**Payment Processing**	Updates payment details (method and amount) for each order. Supports Credit Card, Debit Card, UPI, Wallet, and Cash.
8	**Automatic Stock Update (Trigger)**	Triggers automatically adjust Menu.stock when items are ordered or deleted. Prevents negative stock levels.
9	**Order Status Tracking**	Order statuses are updated by users/admins. A trigger logs every change in Order_Status_History.
10	**Review Submission**	Users can submit ratings and comments through the AddReview procedure. Validation ensures ratings between 1 and 5.
11	**Admin Dashboard**	Admins can add/delete restaurants, manage menu items, and update order statuses from a single interface.
12	**Coupon Management**	Coupon codes provide percentage-based discounts with validation of expiry date and max discount amount.
13	**Data Analytics & Insights**	Functions and queries provide insights such as average restaurant ratings.
14	**Security & Integrity**	The system ensures data consistency through foreign keys, constraints, and controlled SQL operations in the application layer.

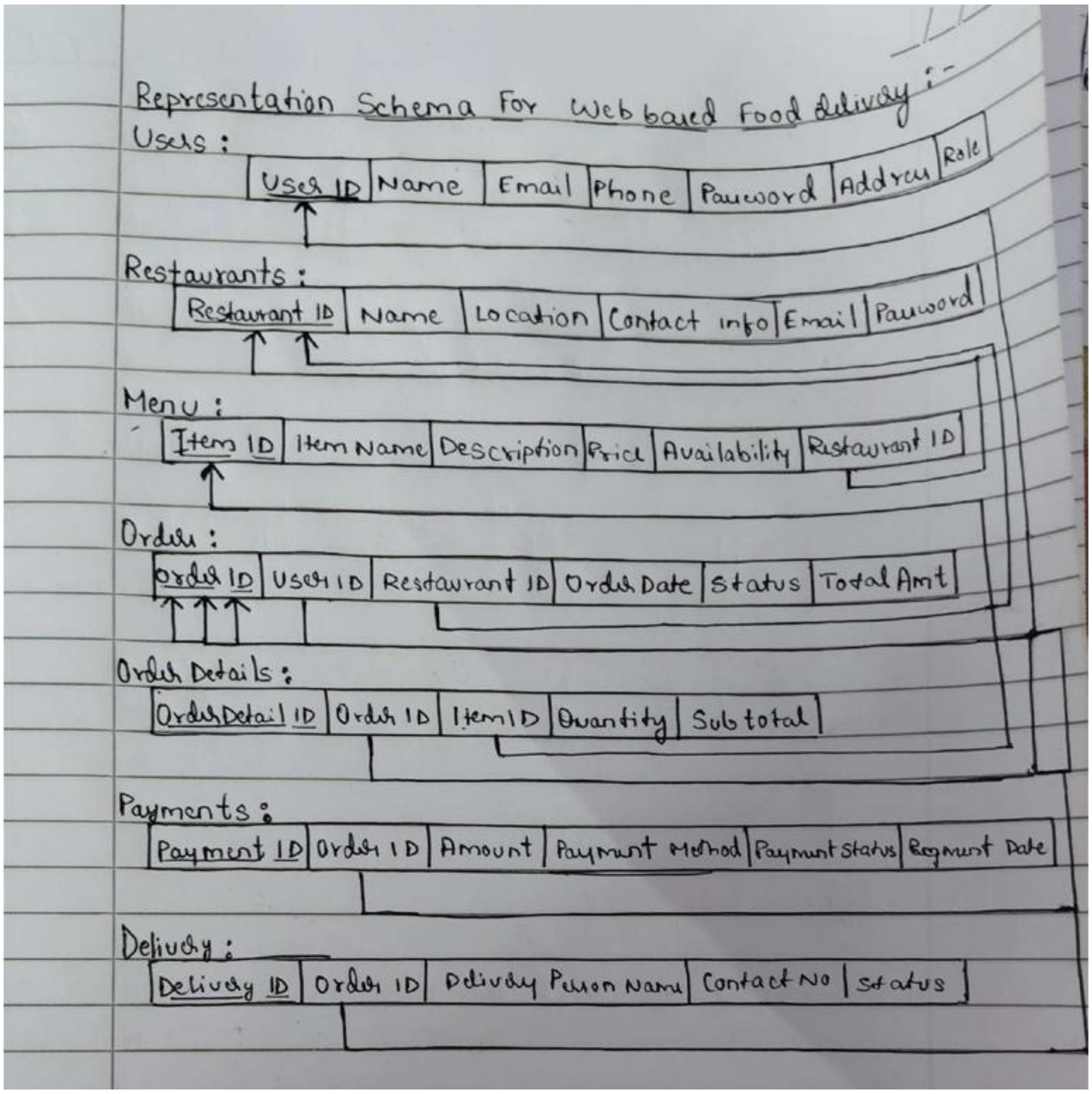
2.5 ER Diagram and Representation Schema



Users
Password
Phone
Email
Name
UserID
Address
Role
1
place
M
Orders
Order ID
UserID
Restaurant ID
Total Amount
Order Date
Status
M
receives
1
Restaurant
Restaurant ID
Name
Location
Contact No
Email
Password
1
Offers
M
M
Menu
Item ID
Restaurant ID
Item Name
Description
Price
Availability
1
1
contains
M
Order details
N
Included in
Payments
1
has
Amount
Order ID
Payment ID
Payment Method
Online
COD
Payment Status
Payment Date
assigned to
1
Delivery
Delivery ID
Order ID
Delivery Person Name
Status
Contact No
Order detail ID
Order ID
Item ID
Quantity
Subtotal



ItemID
ItemName
Description
Price
Availability
Menu
M
Includes
N
N
OrderDetailID
Quantity
Subtotal
OrderDetails
N
Password
RestaurantID
Name
Location
ContactNo
Email
Restaurants
1
Offers
1
Receives
N
Contains
1
OrderID
OrderDate
Status
TotalAmount
Orders
N
Amount
PaymentMethod
PaymentStatus
PaymentDate
Payments
1
1
Has
PaymentID
Name
Email
Phone
Password
Address
Role
UserID
Users
1
Places
DeliveryPersonName
ContactNo
Status
DeliveryID
Delivery
1
1
AssignedTo



Representation Schema For Webbased Food delivery :-

Users :

User ID	Name	Email	Phone	Password	Address	Role

Restaurants :

Restaurant ID	Name	Location	Contact info	Email	Password

Menu :

Item ID	Item Name	Description	Price	Availability	Restaurant ID

Order :

Order ID	User ID	Restaurant ID	Order Date	Status	Total Amt

Order Details :

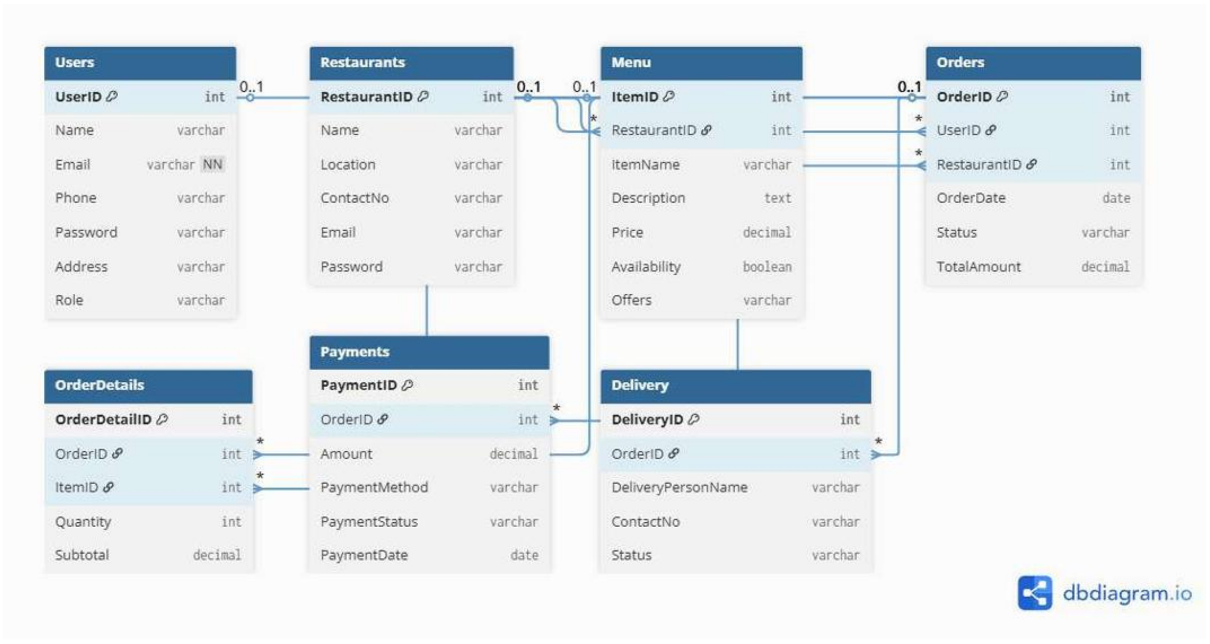
OrderDetail ID	Order ID	Item ID	Quantity	Subtotal

Payments :

Payment ID	Order ID	Amount	Payment Method	Payment Status	Payment Date

Delivery :

Delivery ID	Order ID	Delivery Person Name	Contact No	Status



3. List of Softwares/Tools/Programming languages used

Sl. No.	Software / Tool / Language	Purpose / Description
1	**MySQL 8.0**	Backend relational database used to create tables, relationships, triggers, procedures, and functions.
2	**MySQL Workbench**	Tool for database design, ER diagram creation, query execution, and schema visualization.
3	**Python 3.x**	Primary programming language used for backend logic and database integration.
4	**Streamlit**	Web-based frontend framework used to develop the user interface for both users and admins.
5	**mysql-connector-python**	Python library used to connect the Streamlit application with the MySQL database.
6	**pandas**	Python library used for handling and displaying database query results in tabular form.
7	**Pillow (PIL)**	Library used for image handling and display in the Streamlit UI.
8	**Git & GitHub**	Version control and repository hosting platform for maintaining project code and documentation.
9	**VS Code**	Integrated development environments (IDEs) used for writing and debugging Python and SQL code.
10	**Windows 10**	Operating system used for project development and testing environment.

4. DDL Commands

```
-- DATABASE
DROP DATABASE IF EXISTS FoodOrdering;
CREATE DATABASE FoodOrdering;
USE FoodOrdering;

-- USERS
CREATE TABLE Users (
   user_id INT AUTO_INCREMENT PRIMARY KEY,
   name VARCHAR(100) NOT NULL,
   email VARCHAR(100) UNIQUE NOT NULL,
   phone VARCHAR(20),
   address VARCHAR(255),
   password VARCHAR(255),
   created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
   updated_at DATETIME DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
);

-- RESTAURANTS
CREATE TABLE Restaurants (
   restaurant_id INT AUTO_INCREMENT PRIMARY KEY,
   name VARCHAR(100) NOT NULL,
   address VARCHAR(255),
   phone VARCHAR(20),
   created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
   updated_at DATETIME DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
);

-- DELIVERY PARTNERS
CREATE TABLE Delivery_Partners (
   delivery_partner_id INT AUTO_INCREMENT PRIMARY KEY,
   name VARCHAR(100),
   phone VARCHAR(20),
   created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
   updated_at DATETIME DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
);

-- MENU
CREATE TABLE Menu (
   menu_id INT AUTO_INCREMENT PRIMARY KEY,
   restaurant_id INT NOT NULL,
   name VARCHAR(100) NOT NULL,
   price DECIMAL(8,2) NOT NULL,
   category VARCHAR(50),
   stock INT DEFAULT 100,
   created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
   updated_at DATETIME DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
   FOREIGN KEY (restaurant_id) REFERENCES Restaurants(restaurant_id)
);

-- ORDERS
CREATE TABLE Orders (
   order_id INT AUTO_INCREMENT PRIMARY KEY,
   user_id INT NOT NULL,
   order_date DATETIME DEFAULT CURRENT_TIMESTAMP,
   total_amount DECIMAL(10,2) DEFAULT 0.00,
   status ENUM('Pending', 'Confirmed', 'Out for Delivery', 'Delivered', 'Cancelled') DEFAULT 'Pending',
   delivery_partner_id INT DEFAULT NULL,
   coupon_code VARCHAR(50),
   created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
   updated_at DATETIME DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
   FOREIGN KEY (user_id) REFERENCES Users(user_id),
   FOREIGN KEY (delivery_partner_id) REFERENCES Delivery_Partners(delivery_partner_id)
);
```

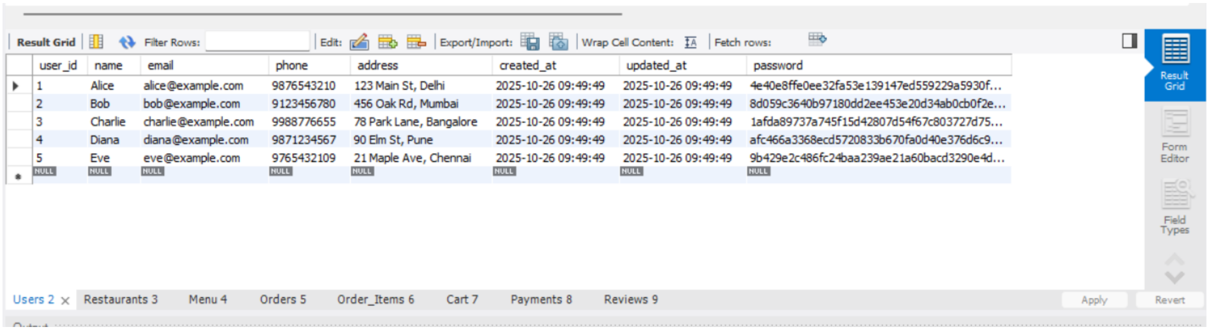
```
-- ORDER ITEMS
CREATE TABLE Order_Items (
    order_item_id INT AUTO_INCREMENT PRIMARY KEY,
    order_id INT NOT NULL,
    menu_id INT NOT NULL,
    quantity INT DEFAULT 1,
    created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
    updated_at DATETIME DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
    FOREIGN KEY (order_id) REFERENCES Orders(order_id),
    FOREIGN KEY (menu_id) REFERENCES Menu(menu_id),
    CHECK (quantity > 0)
);

-- CART
CREATE TABLE Cart (
    cart_id INT AUTO_INCREMENT PRIMARY KEY,
    user_id INT NOT NULL,
    menu_id INT NOT NULL,
    quantity INT DEFAULT 1,
    created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
    updated_at DATETIME DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
    FOREIGN KEY (user_id) REFERENCES Users(user_id),
    FOREIGN KEY (menu_id) REFERENCES Menu(menu_id),
    UNIQUE KEY uniq_user_menu (user_id, menu_id),
    CHECK (quantity > 0)
);

-- REVIEWS
CREATE TABLE Reviews (
    review_id INT AUTO_INCREMENT PRIMARY KEY,
    user_id INT NOT NULL,
    restaurant_id INT NOT NULL,
    rating INT NOT NULL CHECK (rating BETWEEN 1 AND 5),
    comment VARCHAR(500),
    review_date DATETIME DEFAULT CURRENT_TIMESTAMP,
    created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
    updated_at DATETIME DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
    FOREIGN KEY (user_id) REFERENCES Users(user_id),
    FOREIGN KEY (restaurant_id) REFERENCES Restaurants(restaurant_id)
);

-- PAYMENTS
CREATE TABLE Payments (
    payment_id INT AUTO_INCREMENT PRIMARY KEY,
    order_id INT NOT NULL,
    payment_date DATETIME DEFAULT CURRENT_TIMESTAMP,
    amount DECIMAL(10,2),
    method VARCHAR(50),
    status ENUM('Pending', 'Completed', 'Failed') DEFAULT 'Pending',
    created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
    updated_at DATETIME DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
    FOREIGN KEY (order_id) REFERENCES Orders(order_id)
);

-- ORDER STATUS HISTORY
CREATE TABLE Order_Status_History (
    history_id INT AUTO_INCREMENT PRIMARY KEY,
    order_id INT NOT NULL,
    old_status VARCHAR(50),
    new_status VARCHAR(50),
    changed_at DATETIME DEFAULT CURRENT_TIMESTAMP,
    changed_by VARCHAR(100) DEFAULT 'system',
    FOREIGN KEY (order_id) REFERENCES Orders(order_id)
);
```

```
-- COUPONS
CREATE TABLE Coupons (
    coupon_id INT AUTO_INCREMENT PRIMARY KEY,
    code VARCHAR(50) UNIQUE,
    discount_percent INT CHECK (discount_percent BETWEEN 0 AND 100),
    max_discount_amount DECIMAL(10,2),
    expiry_date DATE,
    active BOOLEAN DEFAULT TRUE,
    created_at DATETIME DEFAULT CURRENT_TIMESTAMP
);
```

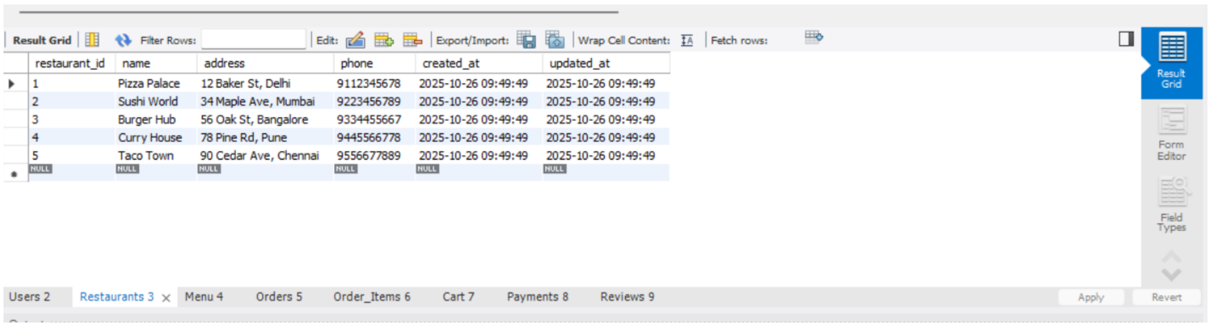
5. CRUD Operations

INSERT:

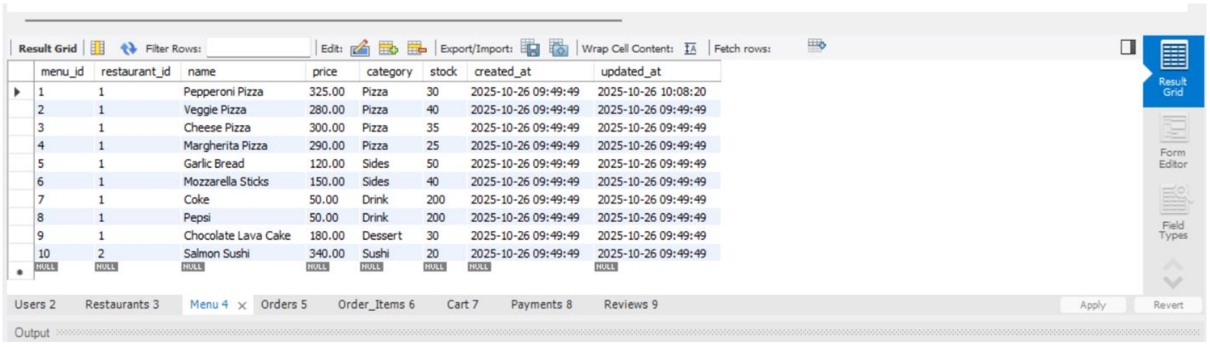
```
SHOW TABLES;
SELECT * FROM Users LIMIT 5;
SELECT * FROM Restaurants LIMIT 5;
SELECT * FROM Menu LIMIT 10;
SELECT * FROM Orders LIMIT 10;
SELECT * FROM Order_Items LIMIT 10;
SELECT * FROM Cart LIMIT 10;
SELECT * FROM Payments LIMIT 10;
SELECT * FROM Reviews LIMIT 10;
```



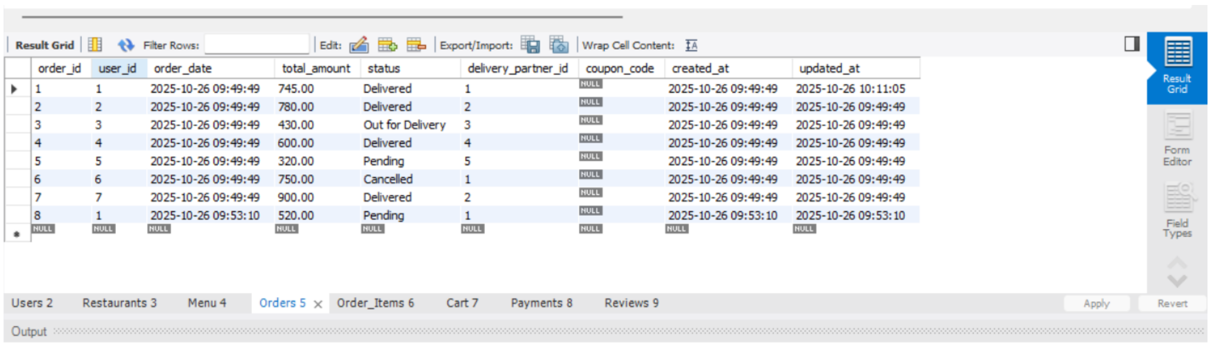
user_id	name	email	phone	address	created_at	updated_at	password
1	Alice	alice@example.com	9876543210	123 Main St, Delhi	2025-10-26 09:49:49	2025-10-26 09:49:49	4e40e8ffe0ee32fa53e139147ed559229a5930f...
2	Bob	bob@example.com	9123456780	456 Oak Rd, Mumbai	2025-10-26 09:49:49	2025-10-26 09:49:49	8d059c3640b97180dd2ee453e20d34ab0cb0f2e...
3	Charlie	charlie@example.com	9988776655	78 Park Lane, Bangalore	2025-10-26 09:49:49	2025-10-26 09:49:49	1afda89737a745f15d42807d54f67c803727d75...
4	Diana	diana@example.com	9871234567	90 Elm St, Pune	2025-10-26 09:49:49	2025-10-26 09:49:49	afc466a3368ecd5720833b670fa0d40e376d6c9...
5	Eve	eve@example.com	9765432109	21 Maple Ave, Chennai	2025-10-26 09:49:49	2025-10-26 09:49:49	9b429e2c486fc24baa239ae21a60bacd3290e4d...
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL



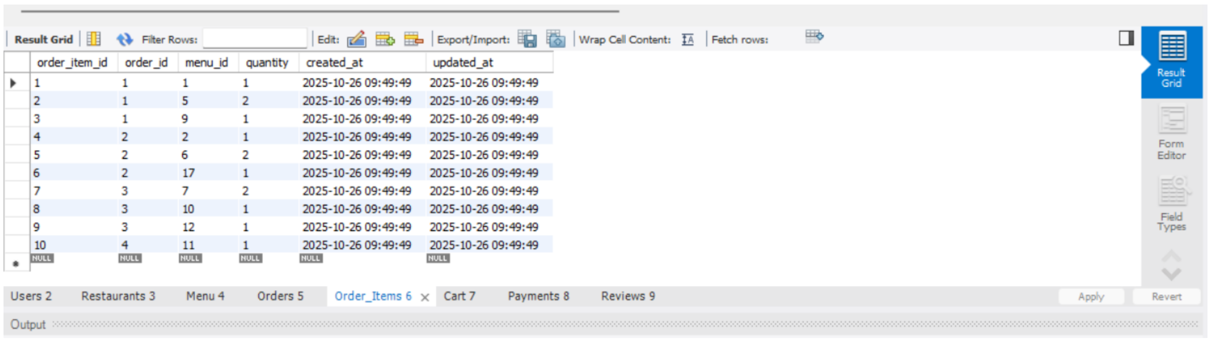
restaurant_id	name	address	phone	created_at	updated_at
1	Pizza Palace	12 Baker St, Delhi	9112345678	2025-10-26 09:49:49	2025-10-26 09:49:49
2	Sushi World	34 Maple Ave, Mumbai	9223456789	2025-10-26 09:49:49	2025-10-26 09:49:49
3	Burger Hub	56 Oak St, Bangalore	9334455667	2025-10-26 09:49:49	2025-10-26 09:49:49
4	Curry House	78 Pine Rd, Pune	9445566778	2025-10-26 09:49:49	2025-10-26 09:49:49
5	Taco Town	90 Cedar Ave, Chennai	9556677889	2025-10-26 09:49:49	2025-10-26 09:49:49
NULL	NULL	NULL	NULL	NULL	NULL



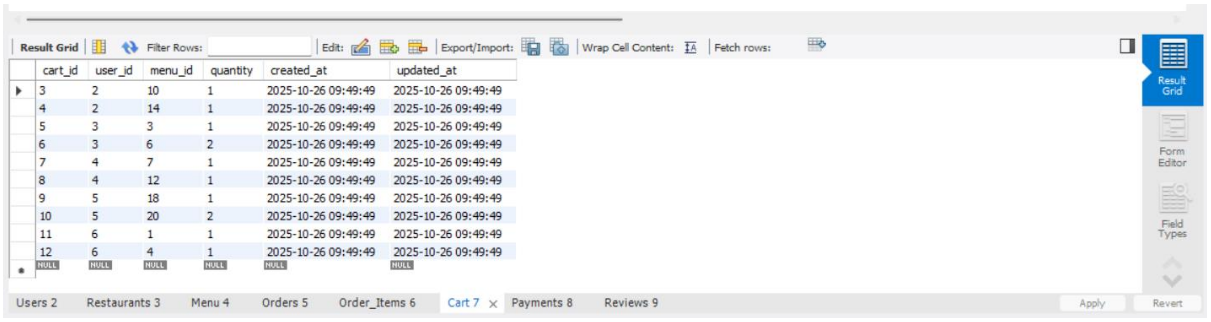
menu_id	restaurant_id	name	price	category	stock	created_at	updated_at
1	1	Pepperoni Pizza	325.00	Pizza	30	2025-10-26 09:49:49	2025-10-26 10:08:20
2	1	Veggie Pizza	280.00	Pizza	40	2025-10-26 09:49:49	2025-10-26 09:49:49
3	1	Cheese Pizza	300.00	Pizza	35	2025-10-26 09:49:49	2025-10-26 09:49:49
4	1	Margherita Pizza	290.00	Pizza	25	2025-10-26 09:49:49	2025-10-26 09:49:49
5	1	Garlic Bread	120.00	Sides	50	2025-10-26 09:49:49	2025-10-26 09:49:49
6	1	Mozzarella Sticks	150.00	Sides	40	2025-10-26 09:49:49	2025-10-26 09:49:49
7	1	Coke	50.00	Drink	200	2025-10-26 09:49:49	2025-10-26 09:49:49
8	1	Pepsi	50.00	Drink	200	2025-10-26 09:49:49	2025-10-26 09:49:49
9	1	Chocolate Lava Cake	180.00	Dessert	30	2025-10-26 09:49:49	2025-10-26 09:49:49
10	2	Salmon Sushi	340.00	Sushi	20	2025-10-26 09:49:49	2025-10-26 09:49:49
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL



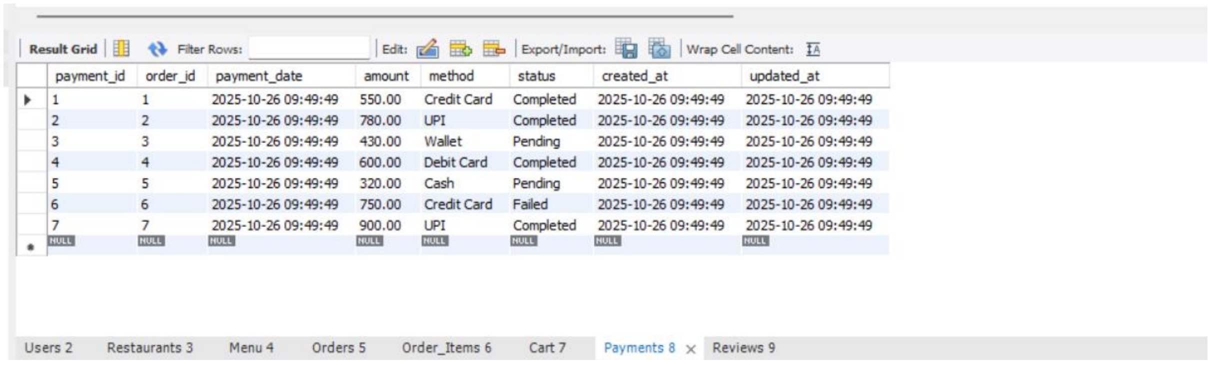
order_id	user_id	order_date	total_amount	status	delivery_partner_id	coupon_code	created_at	updated_at
1	1	2025-10-26 09:49:49	745.00	Delivered	1	NULL	2025-10-26 09:49:49	2025-10-26 10:11:05
2	2	2025-10-26 09:49:49	780.00	Delivered	2	NULL	2025-10-26 09:49:49	2025-10-26 09:49:49
3	3	2025-10-26 09:49:49	430.00	Out for Delivery	3	NULL	2025-10-26 09:49:49	2025-10-26 09:49:49
4	4	2025-10-26 09:49:49	600.00	Delivered	4	NULL	2025-10-26 09:49:49	2025-10-26 09:49:49
5	5	2025-10-26 09:49:49	320.00	Pending	5	NULL	2025-10-26 09:49:49	2025-10-26 09:49:49
6	6	2025-10-26 09:49:49	750.00	Cancelled	1	NULL	2025-10-26 09:49:49	2025-10-26 09:49:49
7	7	2025-10-26 09:49:49	900.00	Delivered	2	NULL	2025-10-26 09:49:49	2025-10-26 09:49:49
8	1	2025-10-26 09:53:10	520.00	Pending	1	NULL	2025-10-26 09:53:10	2025-10-26 09:53:10
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL



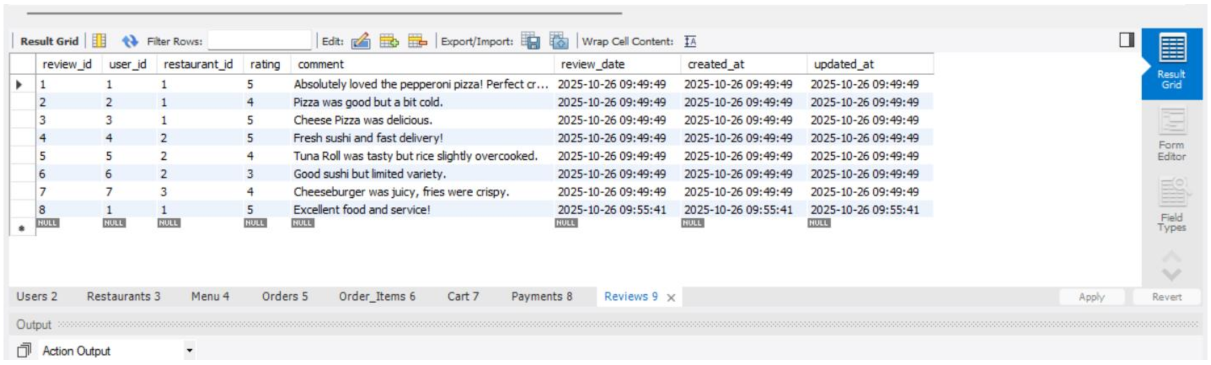
order_item_id	order_id	menu_id	quantity	created_at	updated_at
1	1	1	1	2025-10-26 09:49:49	2025-10-26 09:49:49
2	1	5	2	2025-10-26 09:49:49	2025-10-26 09:49:49
3	1	9	1	2025-10-26 09:49:49	2025-10-26 09:49:49
4	2	2	1	2025-10-26 09:49:49	2025-10-26 09:49:49
5	2	6	2	2025-10-26 09:49:49	2025-10-26 09:49:49
6	2	17	1	2025-10-26 09:49:49	2025-10-26 09:49:49
7	3	7	2	2025-10-26 09:49:49	2025-10-26 09:49:49
8	3	10	1	2025-10-26 09:49:49	2025-10-26 09:49:49
9	3	12	1	2025-10-26 09:49:49	2025-10-26 09:49:49
10	4	11	1	2025-10-26 09:49:49	2025-10-26 09:49:49
NULL	NULL	NULL	NULL	NULL	NULL



cart_id	user_id	menu_id	quantity	created_at	updated_at
3	2	10	1	2025-10-26 09:49:49	2025-10-26 09:49:49
4	2	14	1	2025-10-26 09:49:49	2025-10-26 09:49:49
5	3	3	1	2025-10-26 09:49:49	2025-10-26 09:49:49
6	3	6	2	2025-10-26 09:49:49	2025-10-26 09:49:49
7	4	7	1	2025-10-26 09:49:49	2025-10-26 09:49:49
8	4	12	1	2025-10-26 09:49:49	2025-10-26 09:49:49
9	5	18	1	2025-10-26 09:49:49	2025-10-26 09:49:49
10	5	20	2	2025-10-26 09:49:49	2025-10-26 09:49:49
11	6	1	1	2025-10-26 09:49:49	2025-10-26 09:49:49
12	6	4	1	2025-10-26 09:49:49	2025-10-26 09:49:49
NULL	NULL	NULL	NULL	NULL	NULL

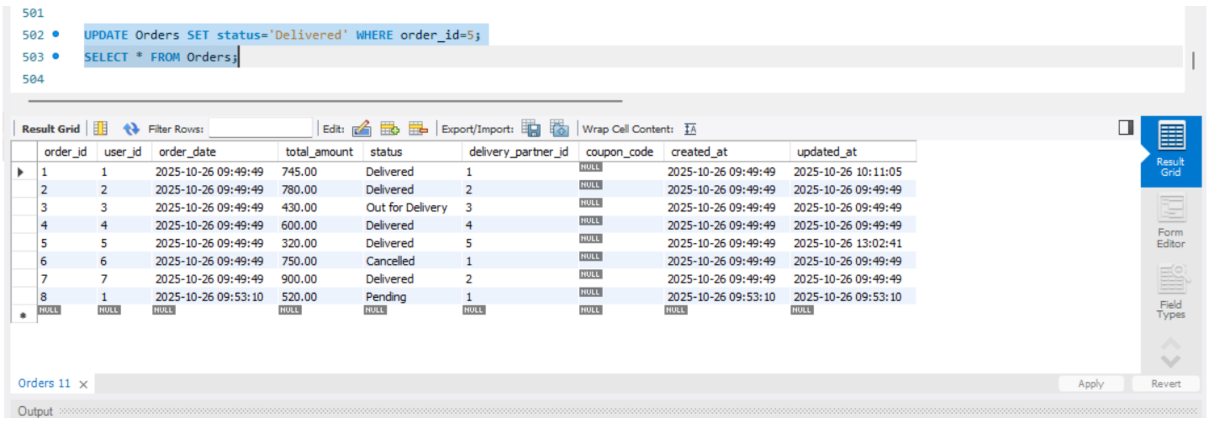


payment_id	order_id	payment_date	amount	method	status	created_at	updated_at
1	1	2025-10-26 09:49:49	550.00	Credit Card	Completed	2025-10-26 09:49:49	2025-10-26 09:49:49
2	2	2025-10-26 09:49:49	780.00	UPI	Completed	2025-10-26 09:49:49	2025-10-26 09:49:49
3	3	2025-10-26 09:49:49	430.00	Wallet	Pending	2025-10-26 09:49:49	2025-10-26 09:49:49
4	4	2025-10-26 09:49:49	600.00	Debit Card	Completed	2025-10-26 09:49:49	2025-10-26 09:49:49
5	5	2025-10-26 09:49:49	320.00	Cash	Pending	2025-10-26 09:49:49	2025-10-26 09:49:49
6	6	2025-10-26 09:49:49	750.00	Credit Card	Failed	2025-10-26 09:49:49	2025-10-26 09:49:49
7	7	2025-10-26 09:49:49	900.00	UPI	Completed	2025-10-26 09:49:49	2025-10-26 09:49:49
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL



review_id	user_id	restaurant_id	rating	comment	review_date	created_at	updated_at
1	1	1	5	Absolutely loved the pepperoni pizza! Perfect cr...	2025-10-26 09:49:49	2025-10-26 09:49:49	2025-10-26 09:49:49
2	2	1	4	Pizza was good but a bit cold.	2025-10-26 09:49:49	2025-10-26 09:49:49	2025-10-26 09:49:49
3	3	1	5	Cheese Pizza was delicious.	2025-10-26 09:49:49	2025-10-26 09:49:49	2025-10-26 09:49:49
4	4	2	5	Fresh sushi and fast delivery!	2025-10-26 09:49:49	2025-10-26 09:49:49	2025-10-26 09:49:49
5	5	2	4	Tuna Roll was tasty but rice slightly overcooked.	2025-10-26 09:49:49	2025-10-26 09:49:49	2025-10-26 09:49:49
6	6	2	3	Good sushi but limited variety.	2025-10-26 09:49:49	2025-10-26 09:49:49	2025-10-26 09:49:49
7	7	3	4	Cheeseburger was juicy, fries were crispy.	2025-10-26 09:49:49	2025-10-26 09:49:49	2025-10-26 09:49:49
8	1	1	5	Excellent food and service!	2025-10-26 09:55:41	2025-10-26 09:55:41	2025-10-26 09:55:41
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

UPDATE:



```
UPDATE Orders SET status='Delivered' WHERE order_id=5;
SELECT * FROM Orders;
```

order_id	user_id	order_date	total_amount	status	delivery_partner_id	coupon_code	created_at	updated_at
1	1	2025-10-26 09:49:49	745.00	Delivered	1	NULL	2025-10-26 09:49:49	2025-10-26 10:11:05
2	2	2025-10-26 09:49:49	780.00	Delivered	2	NULL	2025-10-26 09:49:49	2025-10-26 09:49:49
3	3	2025-10-26 09:49:49	430.00	Out for Delivery	3	NULL	2025-10-26 09:49:49	2025-10-26 09:49:49
4	4	2025-10-26 09:49:49	600.00	Delivered	4	NULL	2025-10-26 09:49:49	2025-10-26 09:49:49
5	5	2025-10-26 09:49:49	320.00	Delivered	5	NULL	2025-10-26 09:49:49	2025-10-26 13:02:41
6	6	2025-10-26 09:49:49	750.00	Cancelled	1	NULL	2025-10-26 09:49:49	2025-10-26 09:49:49
7	7	2025-10-26 09:49:49	900.00	Delivered	2	NULL	2025-10-26 09:49:49	2025-10-26 09:49:49
8	1	2025-10-26 09:53:10	520.00	Pending	1	NULL	2025-10-26 09:53:10	2025-10-26 09:53:10
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

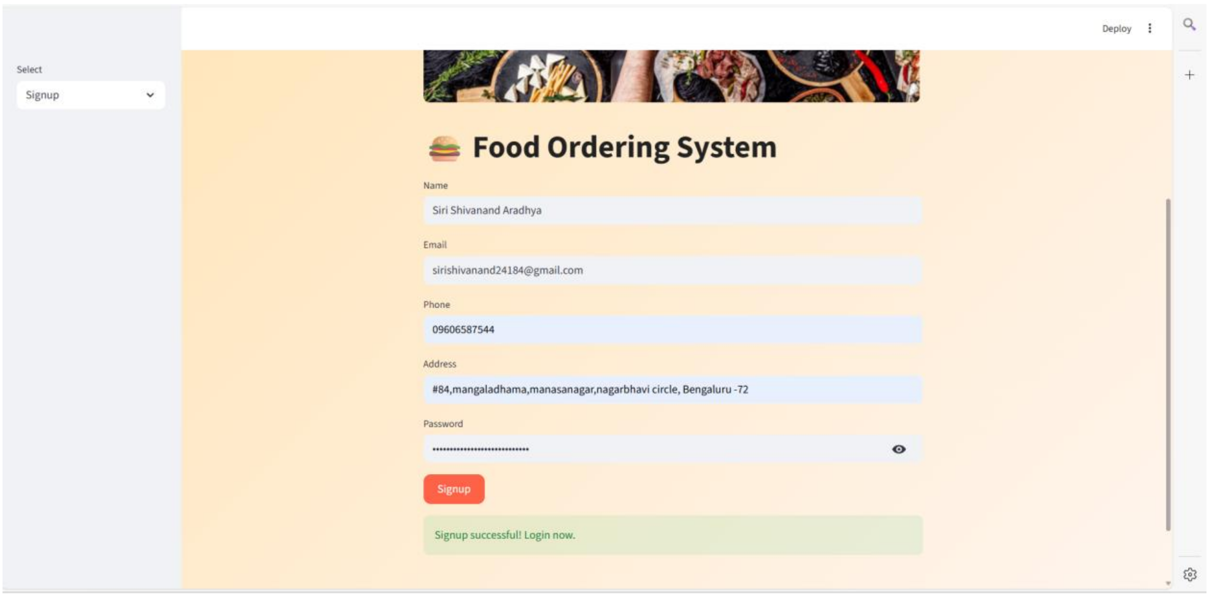
DELETE:

DELETE FROM Cart WHERE user_id=1 AND menu_id=2;

6. List of functionalities/features of the application and its associated screenshots using front end

USER PORTAL

SIGNUP



🍔 Food Ordering System

Select: Signup

Name: Siri Shivanand Aradhya

Email: sirishivanand24184@gmail.com

Phone: 09606587544

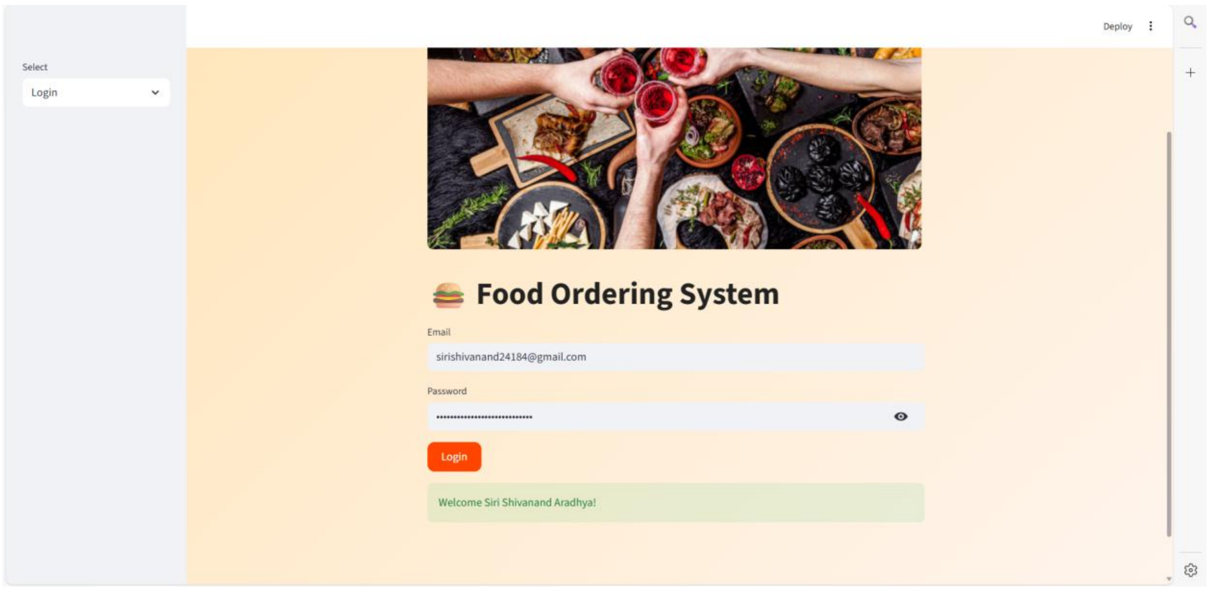
Address: #84,mangaladhama,manasanagar,nagarbhavi circle, Bengaluru -72

Password: ••••••••••••••••••••••••••

Signup

Signup successful! Login now.

LOGIN



🍔 Food Ordering System

Email

sirishivanand24184@gmail.com

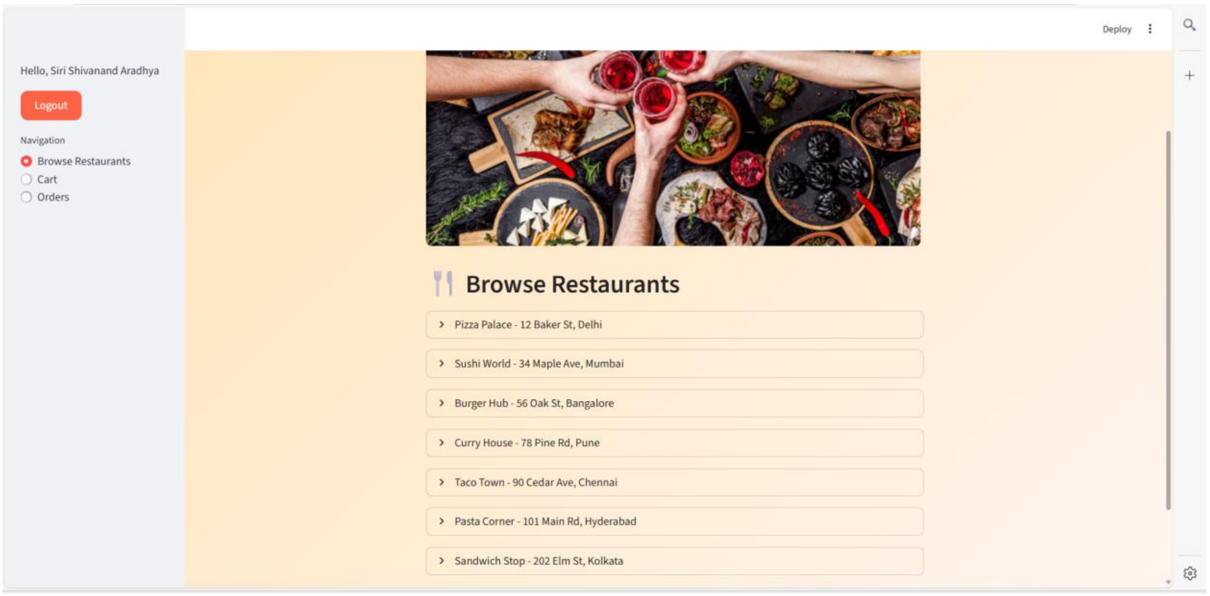
Password

••••••••••••••••••••••••••

Login

Welcome Siri Shivanand Aradhya!

BROWSE RESTAURANT

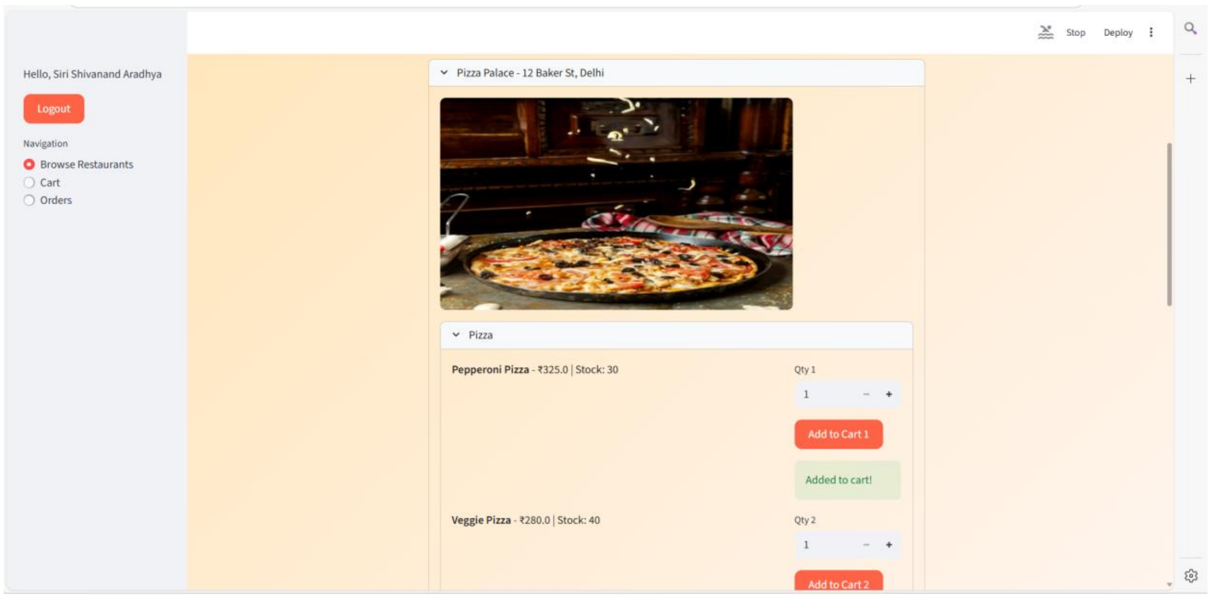


Hello, Siri Shivanand Aradhya

🍴 Browse Restaurants

- Pizza Palace - 12 Baker St, Delhi
- Sushi World - 34 Maple Ave, Mumbai
- Burger Hub - 56 Oak St, Bangalore
- Curry House - 78 Pine Rd, Pune
- Taco Town - 90 Cedar Ave, Chennai
- Pasta Corner - 101 Main Rd, Hyderabad
- Sandwich Stop - 202 Elm St, Kolkata

ADD TO CART



Hello, Siri Shivanand Aradhya

Navigation

[✓ Browse Restaurants]
Cart
Orders

Pizza Palace - 12 Baker St, Delhi

Pizza

Pepperoni Pizza - ₹325.0 | Stock: 30

Qty 1

1

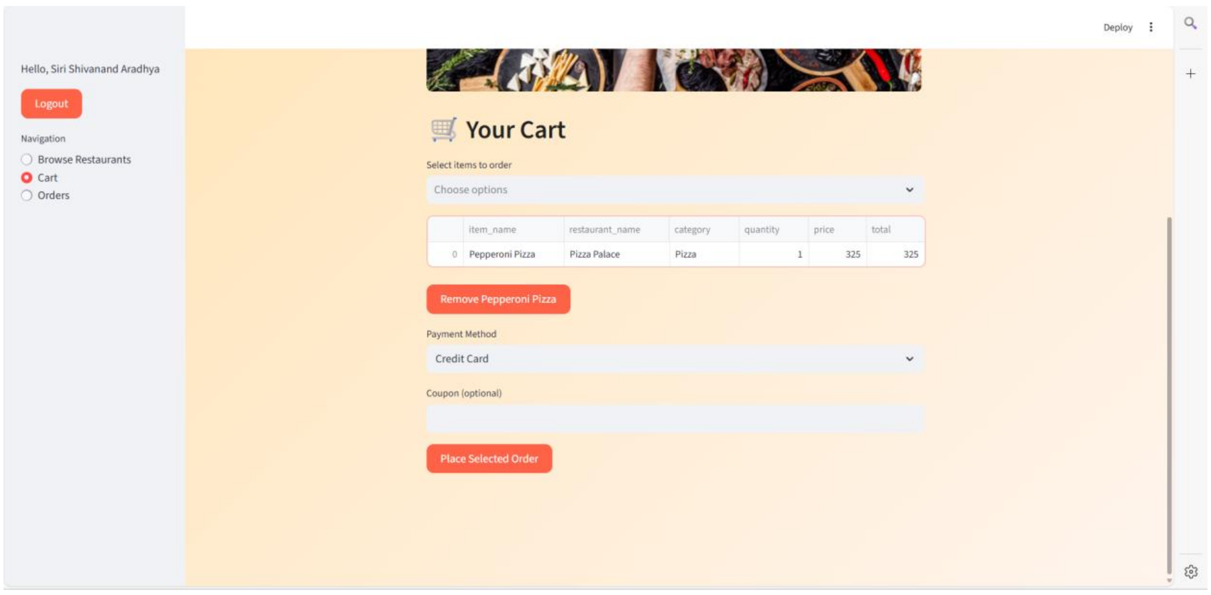
Added to cart!

Veggie Pizza - ₹280.0 | Stock: 40

Qty 2

1

REMOVE FROM CART OR PLACE ORDER



Hello, Siri Shivanand Aradhya

Navigation

Browse Restaurants
[✓ Cart]
Orders

Your Cart

Select items to order

Choose options

	item_name	restaurant_name	category	quantity	price	total
0	Pepperoni Pizza	Pizza Palace	Pizza	1	325	325

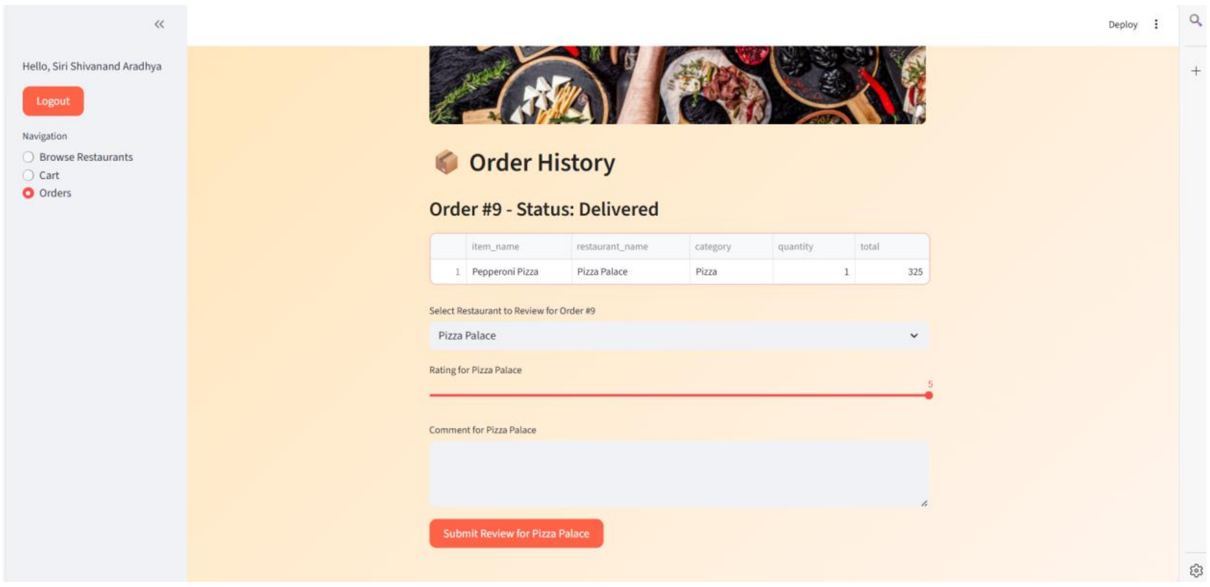
Payment Method

Credit Card

Coupon (optional)

VIEW ORDER HISTORY AND VIEW ORDER STATUS :

RATE AND REVIEW RESTAURANTS AND CANCEL ORDER:



Order History

Order #9 - Status: Delivered

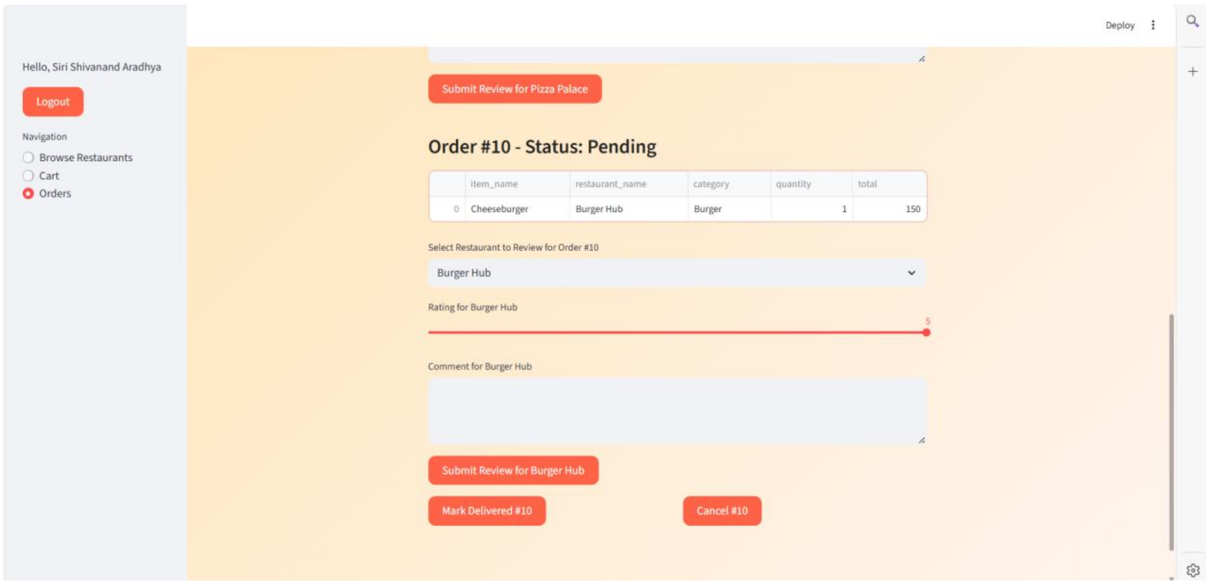
	item_name	restaurant_name	category	quantity	total
1	Pepperoni Pizza	Pizza Palace	Pizza	1	325

Select Restaurant to Review for Order #9: Pizza Palace

Rating for Pizza Palace: 5

Comment for Pizza Palace

Submit Review for Pizza Palace



Order #10 - Status: Pending

	item_name	restaurant_name	category	quantity	total
0	Cheeseburger	Burger Hub	Burger	1	150

Select Restaurant to Review for Order #10: Burger Hub

Rating for Burger Hub: 5

Comment for Burger Hub

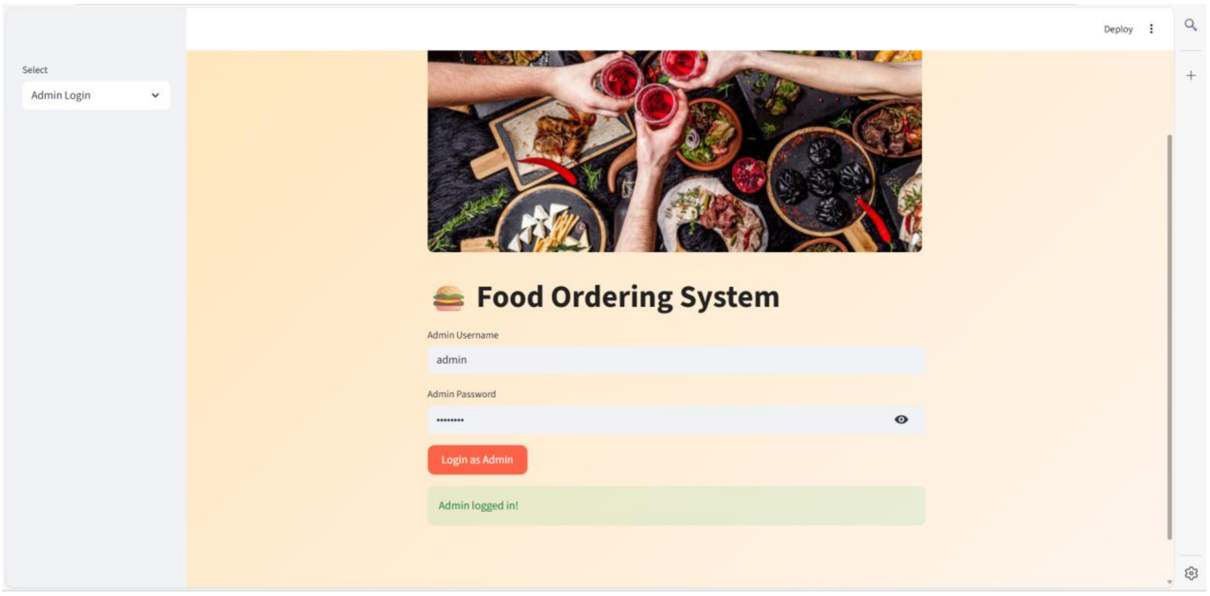
Submit Review for Burger Hub

Mark Delivered #10

Cancel #10

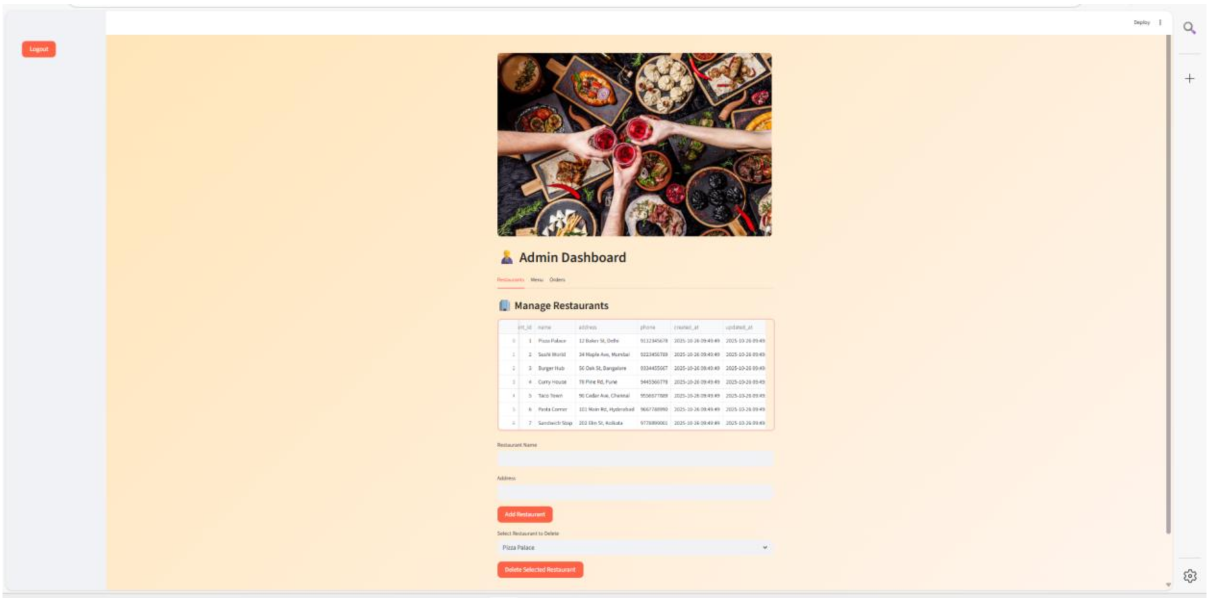
ADMIN PORTAL

LOGIN:



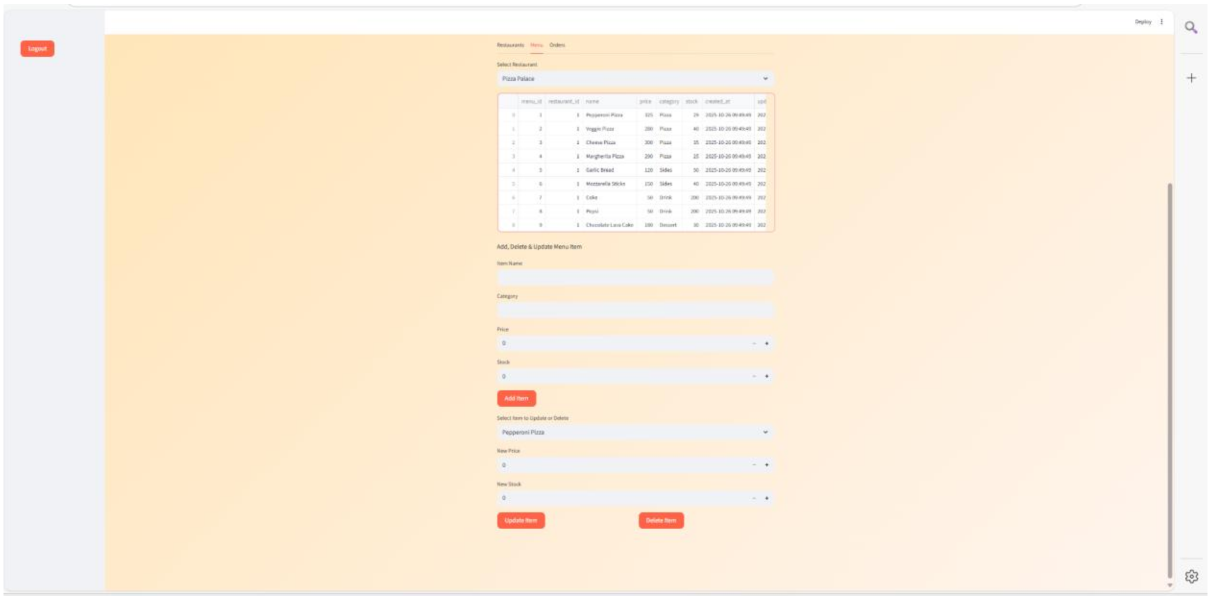
Admin logged in!

MANAGE RESTAURANTS – ADD/DELETE



	rest_id	name	address	phone	created_at	updated_at
0	1	Pizza Palace	12 Baker St, Delhi	9112345678	2025-10-26 09:49:49	2025-10-26 09:49
1	2	Sushi World	34 Maple Ave, Mumbai	9223456789	2025-10-26 09:49:49	2025-10-26 09:49
2	3	Burger Hub	56 Oak St, Bangalore	9334455667	2025-10-26 09:49:49	2025-10-26 09:49
3	4	Curry House	78 Pine Rd, Pune	9445566778	2025-10-26 09:49:49	2025-10-26 09:49
4	5	Taco Town	90 Cedar Ave, Chennai	9556677889	2025-10-26 09:49:49	2025-10-26 09:49
5	6	Pasta Corner	101 Main Rd, Hyderabad	9667788990	2025-10-26 09:49:49	2025-10-26 09:49
6	7	Sandwich Stop	202 Elm St, Kolkata	9778899001	2025-10-26 09:49:49	2025-10-26 09:49

ADD, DELETE, UPDATE MENU ITEMS:


Restaurants Menu Orders

Select Restaurant

Pizza Palace

	menu_id	restaurant_id	name	price	category	stock	created_at	upd
0	1	1	Pepperoni Pizza	325	Pizza	29	2025-10-26 09:49:49	202
1	2	1	Veggie Pizza	280	Pizza	40	2025-10-26 09:49:49	202
2	3	1	Cheese Pizza	300	Pizza	35	2025-10-26 09:49:49	202
3	4	1	Margherita Pizza	290	Pizza	25	2025-10-26 09:49:49	202
4	5	1	Garlic Bread	120	Sides	50	2025-10-26 09:49:49	202
5	6	1	Mozzarella Sticks	150	Sides	40	2025-10-26 09:49:49	202
6	7	1	Coke	50	Drink	200	2025-10-26 09:49:49	202
7	8	1	Pepsi	50	Drink	200	2025-10-26 09:49:49	202
8	9	1	Chocolate Lava Cake	180	Dessert	30	2025-10-26 09:49:49	202

Add, Delete & Update Menu Item

Item Name

Category

Price

0

Stock

0

Select Item to Update or Delete

Pepperoni Pizza

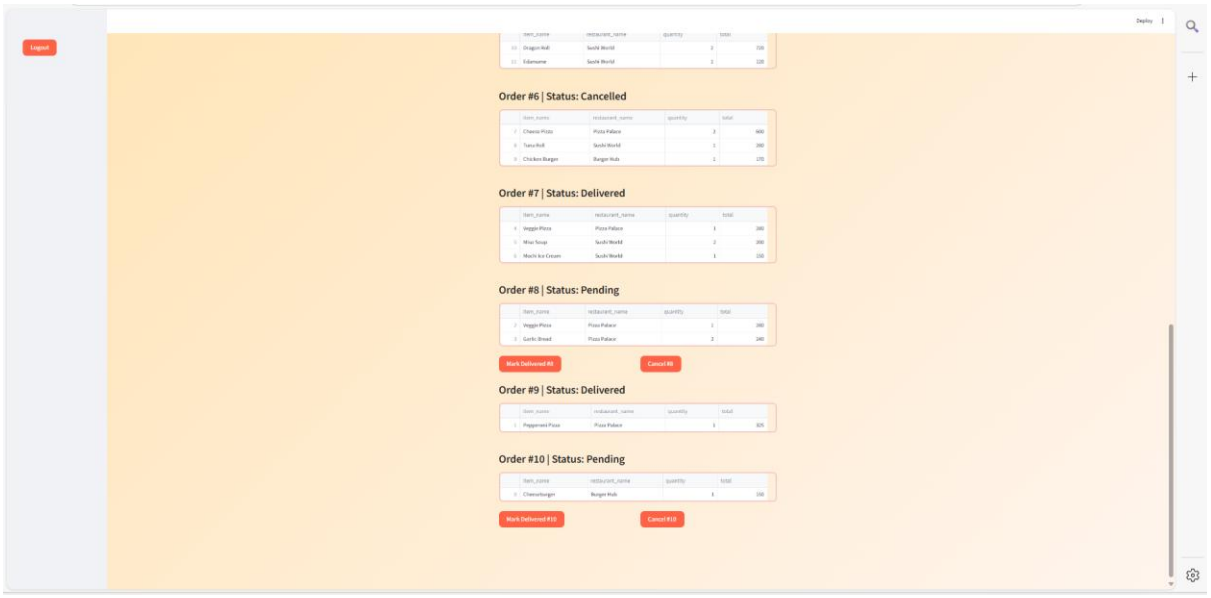
New Price

0

New Stock

0

ORDER MANAGEMENT


	item_name	restaurant_name	quantity	total
10	Dragon Roll	Sushi World	2	720
11	Edamame	Sushi World	1	120

Order #6 | Status: Cancelled

	item_name	restaurant_name	quantity	total
7	Cheese Pizza	Pizza Palace	2	600
8	Tuna Roll	Sushi World	1	280
9	Chicken Burger	Burger Hub	1	170

Order #7 | Status: Delivered

	item_name	restaurant_name	quantity	total
4	Veggie Pizza	Pizza Palace	1	280
5	Miso Soup	Sushi World	2	200
6	Mochi Ice Cream	Sushi World	1	150

Order #8 | Status: Pending

	item_name	restaurant_name	quantity	total
2	Veggie Pizza	Pizza Palace	1	280
3	Garlic Bread	Pizza Palace	2	240

Order #9 | Status: Delivered

	item_name	restaurant_name	quantity	total
1	Pepperoni Pizza	Pizza Palace	1	325

Order #10 | Status: Pending

	item_name	restaurant_name	quantity	total
0	Cheeseburger	Burger Hub	1	150

7. Triggers, Procedures/Functions, Nested query, Join, Aggregate queries

FUNCTIONS

```
DELIMITER //
CREATE FUNCTION GetOrderTotal(p_order_id INT)
RETURNS DECIMAL(10,2)
DETERMINISTIC
BEGIN
   DECLARE total DECIMAL(10,2) DEFAULT 0.00;
   SELECT IFNULL(SUM(m.price * oi.quantity), 0.00) INTO total
   FROM Order_Items oi
   JOIN Menu m ON oi.menu_id = m.menu_id
   WHERE oi.order_id = p_order_id;
   RETURN total;
END;
//
```

```
CREATE FUNCTION GetRestaurantAvgRating(p_restaurant_id INT)
RETURNS DECIMAL(3,2)
DETERMINISTIC
BEGIN
   DECLARE avg_rating DECIMAL(3,2);
   SELECT IFNULL(AVG(rating), 0.00) INTO avg_rating
   FROM Reviews
   WHERE restaurant_id = p_restaurant_id;
   RETURN avg_rating;
END;
//
DELIMITER ;

-- Invoking Functions
SELECT GetOrderTotal(1) AS OrderTotal;
SELECT GetRestaurantAvgRating(1) AS AverageRating;
```

PROCEDURE

```
DELIMITER //
CREATE PROCEDURE PlaceOrderFromCart(IN p_user_id INT, IN p_delivery_partner_id INT)
BEGIN
   DECLARE v_order_id INT;
   INSERT INTO Orders (user_id, total_amount, status, delivery_partner_id)
   VALUES (p_user_id, 0.00, 'Pending', p_delivery_partner_id);

   SET v_order_id = LAST_INSERT_ID();

   INSERT INTO Order_Items (order_id, menu_id, quantity)
   SELECT v_order_id, menu_id, quantity FROM Cart WHERE user_id = p_user_id;

   UPDATE Orders
   SET total_amount = GetOrderTotal(v_order_id)
   WHERE order_id = v_order_id;

   DELETE FROM Cart WHERE user_id = p_user_id;
END;
//

CREATE PROCEDURE AddReview(IN p_user_id INT, IN p_restaurant_id INT, IN p_rating INT, IN p_comment VARCHAR(500))
BEGIN
   IF p_rating < 1 OR p_rating > 5 THEN
      SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Rating must be between 1 and 5';
   END IF;

   INSERT INTO Reviews (user_id, restaurant_id, rating, comment)
   VALUES (p_user_id, p_restaurant_id, p_rating, p_comment);
END;
//
DELIMITER ;

-- Invoking Procedures
CALL PlaceOrderFromCart(1,1);
CALL AddReview(1,1,5,'Excellent food and service!');
```

TRIGGERS:

```
DELIMITER //
CREATE TRIGGER trg_after_insert_order_item
AFTER INSERT ON Order_Items
FOR EACH ROW
BEGIN
   UPDATE Menu SET stock = stock - NEW.quantity WHERE menu_id = NEW.menu_id;
   IF (SELECT stock FROM Menu WHERE menu_id = NEW.menu_id) < 0 THEN
      SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Insufficient stock for the menu item';
   END IF;
```

```
    UPDATE Orders SET total_amount = GetOrderTotal(NEW.order_id) WHERE order_id = NEW.order_id;
END;
//

CREATE TRIGGER trg_after_delete_order_item
AFTER DELETE ON Order_Items
FOR EACH ROW
BEGIN
    UPDATE Menu SET stock = stock + OLD.quantity WHERE menu_id = OLD.menu_id;
    UPDATE Orders SET total_amount = GetOrderTotal(OLD.order_id) WHERE order_id = OLD.order_id;
END;
//

CREATE TRIGGER trg_orders_update_status_after
AFTER UPDATE ON Orders
FOR EACH ROW
BEGIN
    IF NEW.status <> OLD.status THEN
        INSERT INTO Order_Status_History (order_id, old_status, new_status, changed_by)
        VALUES (NEW.order_id, OLD.status, NEW.status, 'system');
    END IF;
END;
//
DELIMITER ;

-- Trigger demonstration
INSERT INTO Order_Items (order_id, menu_id, quantity) VALUES (1,1,2);  -- trg_after_insert_order_item fires
DELETE FROM Order_Items WHERE order_item_id=(SELECT MAX(order_item_id) FROM Order_Items WHERE menu_id=1);  --
trg_after_delete_order_item fires
UPDATE Orders SET status='Delivered' WHERE order_id=1;  -- trg_orders_update_status_after fires
```

NESTED QUERIES:

```
DELETE FROM Order_Items
WHERE order_item_id = (
    SELECT t.order_item_id
    FROM (
        SELECT MAX(order_item_id) AS order_item_id
        FROM Order_Items
        WHERE menu_id = 1
    ) AS t
);
```

JOIN QUERIES:

```
-- Example Join (Participants & Events)
SELECT p.participant_name
FROM participant p
JOIN registration r ON p.participant_id = r.participant_id
JOIN event e ON r.event_id = e.event_id
WHERE e.price > (SELECT AVG(price) FROM event);

-- Orders with Users
SELECT o.order_id, u.name, o.total_amount
FROM Orders o
JOIN Users u ON o.user_id = u.user_id;
```

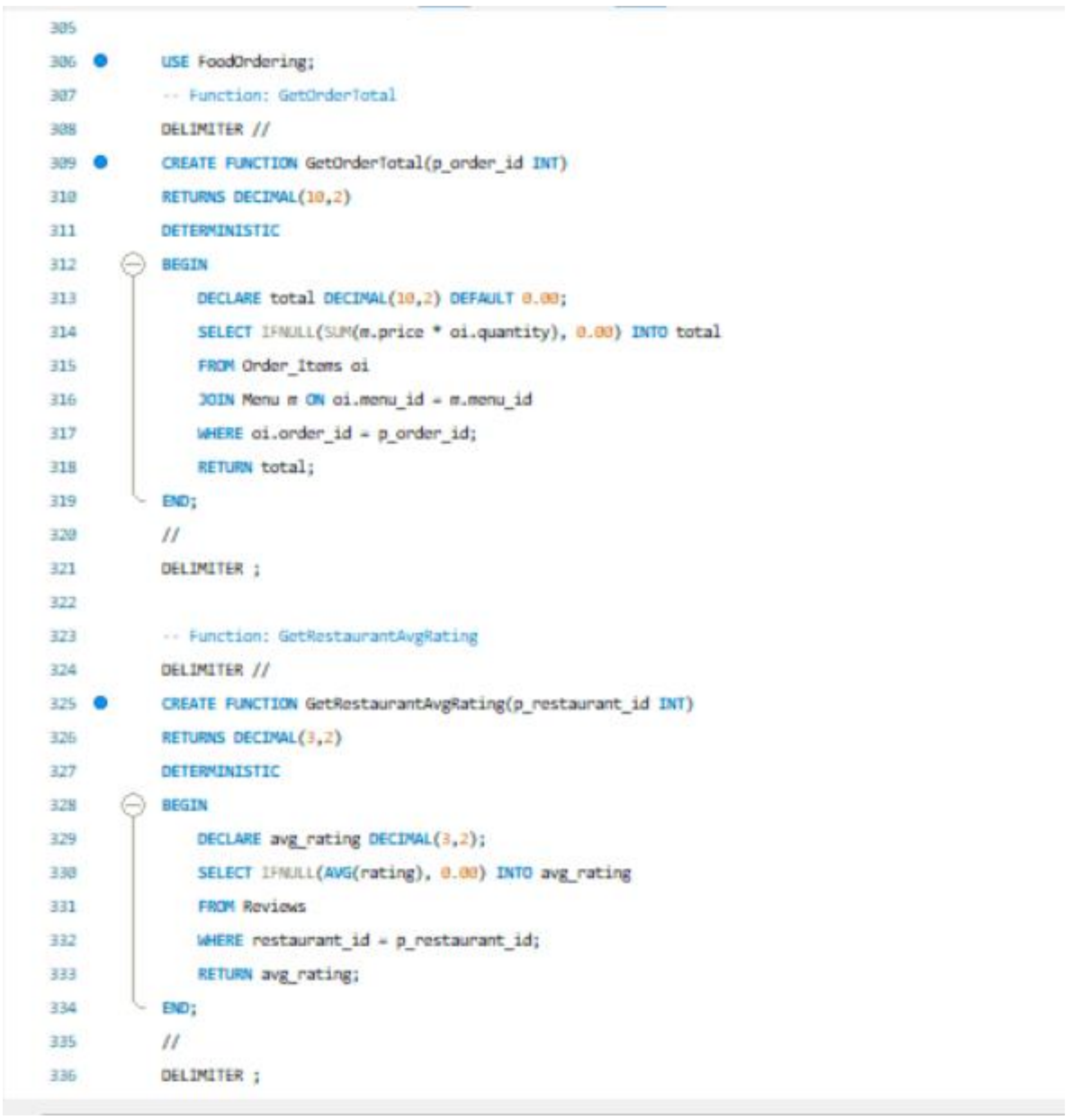
Aggregate Queries:

```
-- Average rating per restaurant
SELECT restaurant_id, AVG(rating) AS manual_avg
FROM Reviews
GROUP BY restaurant_id;
```

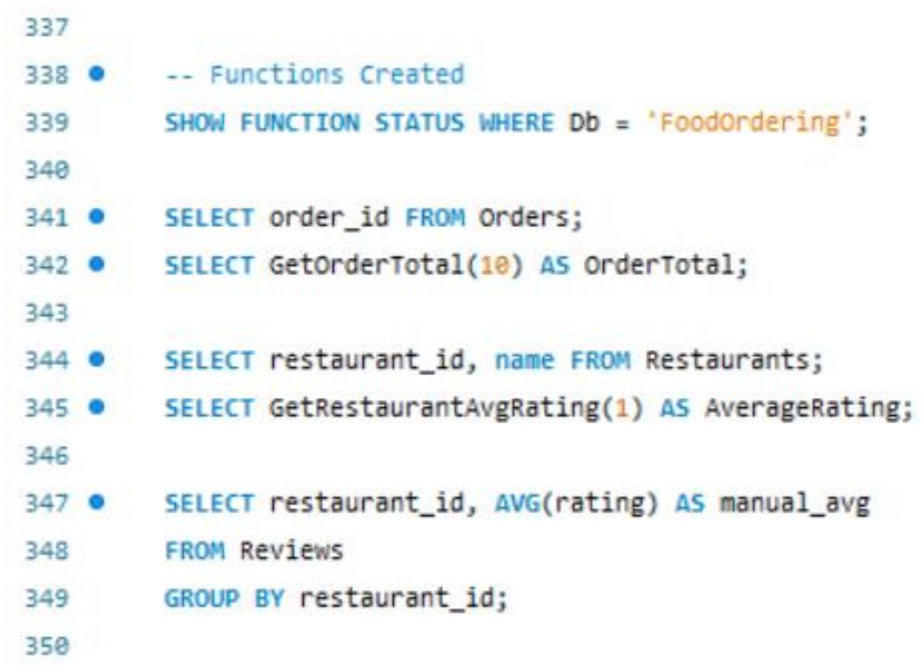
-- Total order amount using function
SELECT GetOrderTotal(1) AS OrderTotal;

8. Ss for triggers, procedure, functions output

FUNTIONS:

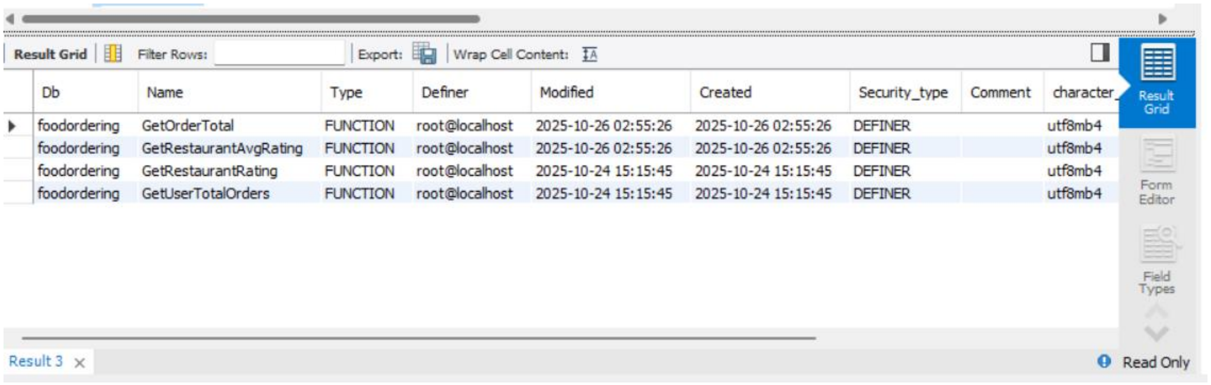


```
USE FoodOrdering;
-- Function: GetOrderTotal
DELIMITER //
CREATE FUNCTION GetOrderTotal(p_order_id INT)
RETURNS DECIMAL(10,2)
DETERMINISTIC
BEGIN
    DECLARE total DECIMAL(10,2) DEFAULT 0.00;
    SELECT IFNULL(SUM(m.price * oi.quantity), 0.00) INTO total
    FROM Order_Items oi
    JOIN Menu m ON oi.menu_id = m.menu_id
    WHERE oi.order_id = p_order_id;
    RETURN total;
END;
//
DELIMITER ;

-- Function: GetRestaurantAvgRating
DELIMITER //
CREATE FUNCTION GetRestaurantAvgRating(p_restaurant_id INT)
RETURNS DECIMAL(3,2)
DETERMINISTIC
BEGIN
    DECLARE avg_rating DECIMAL(3,2);
    SELECT IFNULL(AVG(rating), 0.00) INTO avg_rating
    FROM Reviews
    WHERE restaurant_id = p_restaurant_id;
    RETURN avg_rating;
END;
//
DELIMITER ;
```



```
-- Functions Created
SHOW FUNCTION STATUS WHERE Db = 'FoodOrdering';

SELECT order_id FROM Orders;
SELECT GetOrderTotal(10) AS OrderTotal;

SELECT restaurant_id, name FROM Restaurants;
SELECT GetRestaurantAvgRating(1) AS AverageRating;

SELECT restaurant_id, AVG(rating) AS manual_avg
FROM Reviews
GROUP BY restaurant_id;
```

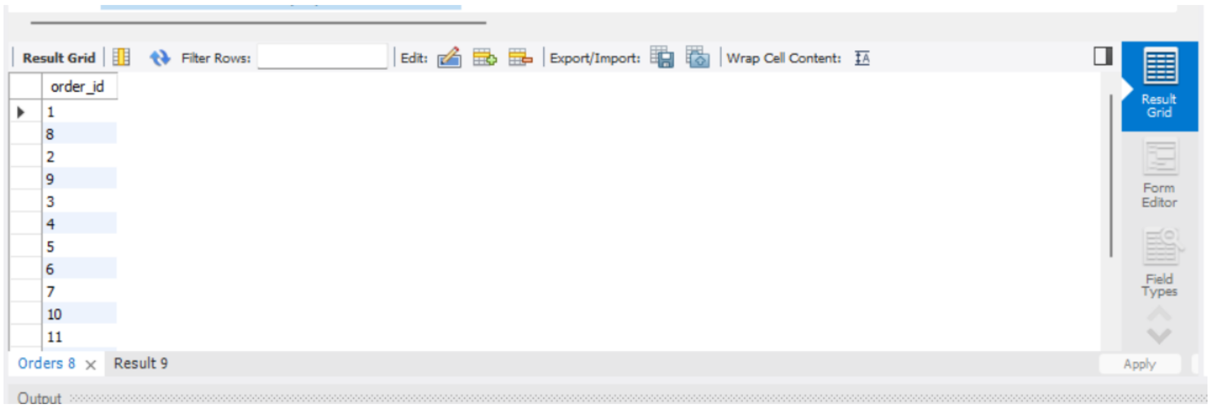
BEFORE:



Db	Name	Type	Definer	Modified	Created	Security_type	Comment	character_
foodordering	GetOrderTotal	FUNCTION	root@localhost	2025-10-26 02:55:26	2025-10-26 02:55:26	DEFINER		utf8mb4
foodordering	GetRestaurantAvgRating	FUNCTION	root@localhost	2025-10-26 02:55:26	2025-10-26 02:55:26	DEFINER		utf8mb4
foodordering	GetRestaurantRating	FUNCTION	root@localhost	2025-10-24 15:15:45	2025-10-24 15:15:45	DEFINER		utf8mb4
foodordering	GetUserTotalOrders	FUNCTION	root@localhost	2025-10-24 15:15:45	2025-10-24 15:15:45	DEFINER		utf8mb4

Result 3 | Read Only

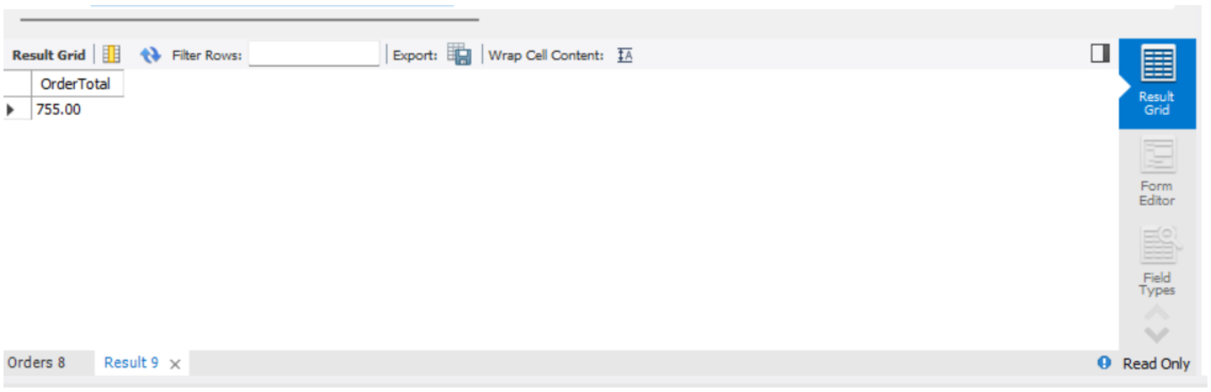
AFTER:



order_id
1
8
2
9
3
4
5
6
7
10
11

Orders 8 | Result 9

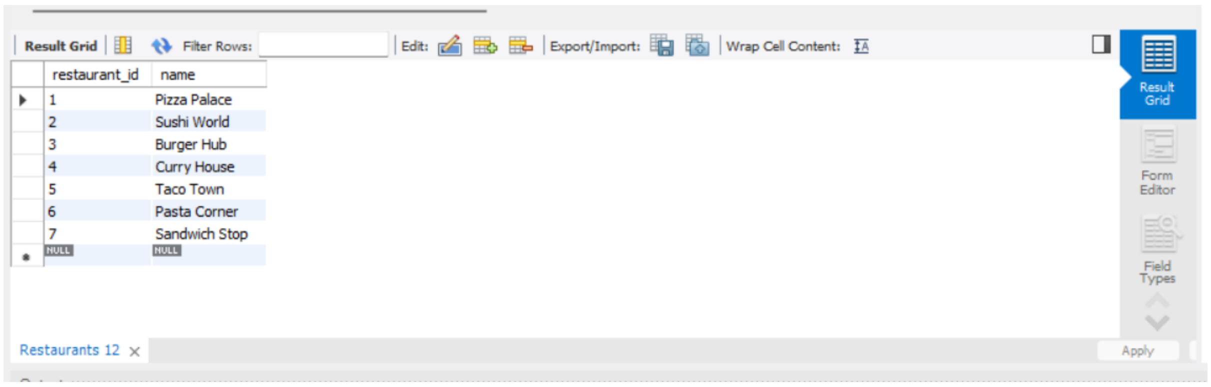
GET ORDER TOTAL


Result Grid | Filter Rows: | Export: | Wrap Cell Content:

	OrderTotal
▶	755.00

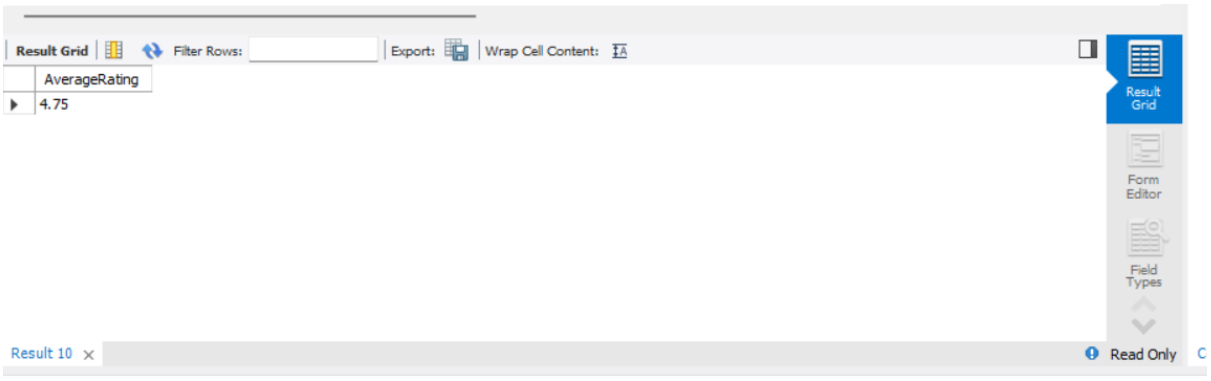
Orders 8 | Result 9 × | Read Only

GET RESTAURENT AVG RATING


Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

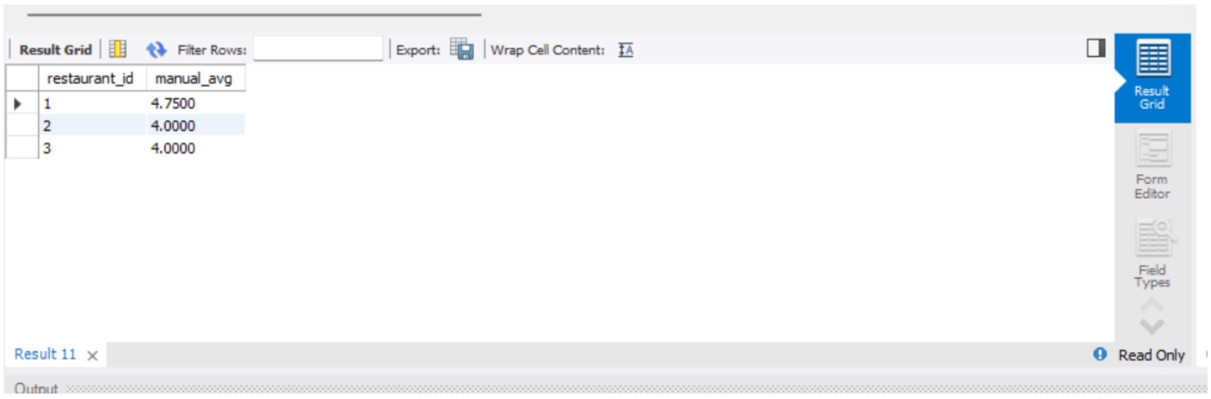
	restaurant_id	name
▶	1	Pizza Palace
	2	Sushi World
	3	Burger Hub
	4	Curry House
	5	Taco Town
	6	Pasta Corner
	7	Sandwich Stop
*	NULL	NULL

Restaurants 12 × | Apply


Result Grid | Filter Rows: | Export: | Wrap Cell Content:

	AverageRating
▶	4.75

Result 10 × | Read Only

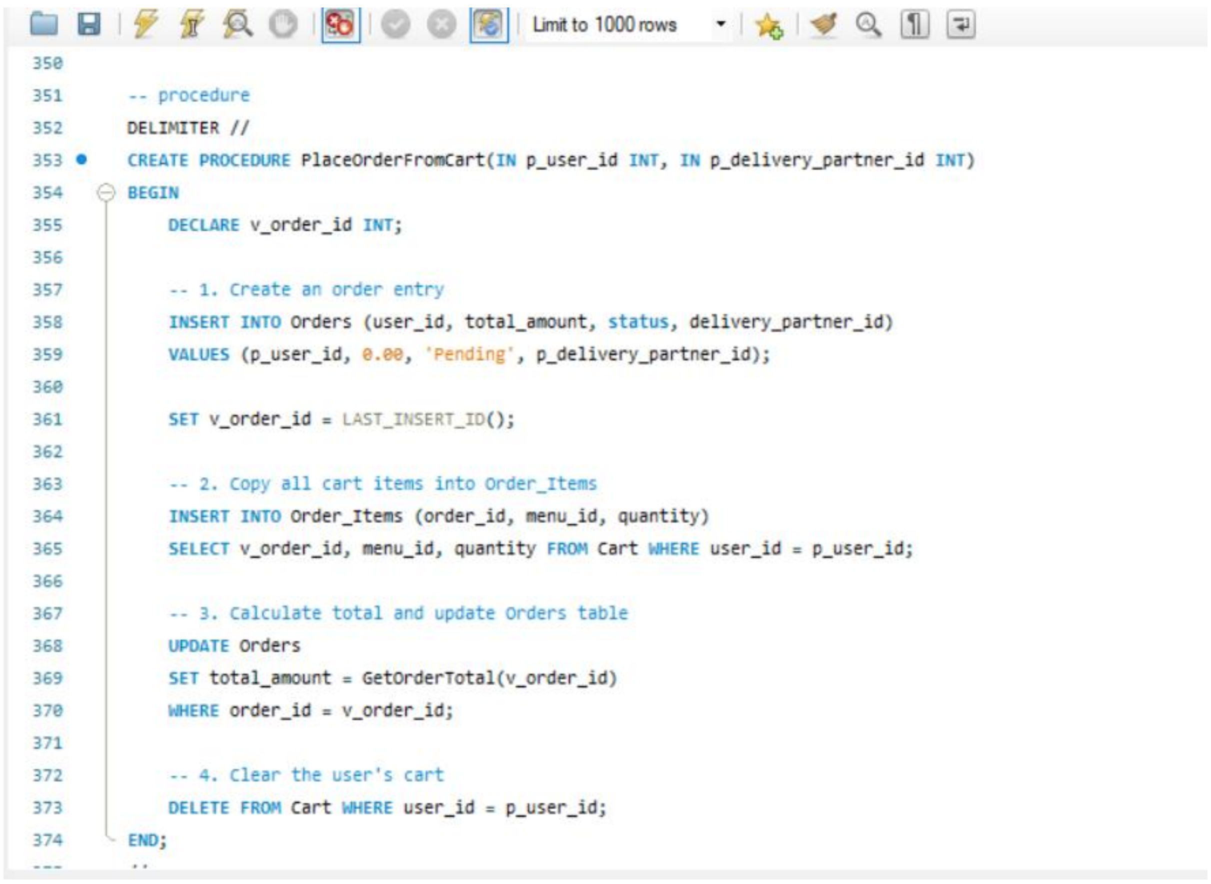

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

	restaurant_id	manual_avg
▶	1	4.7500
	2	4.0000
	3	4.0000

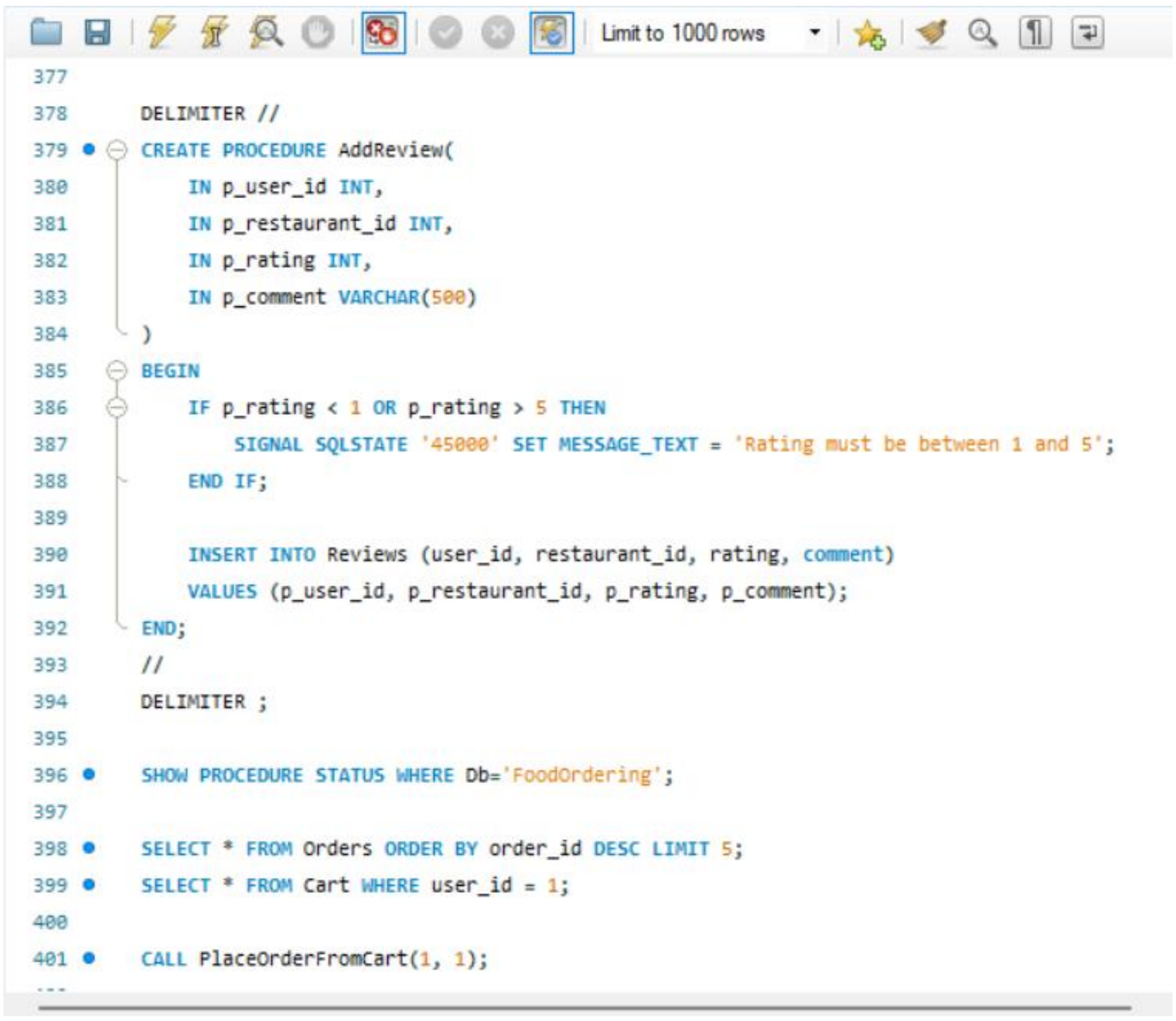
Result 11 × | Read Only

Output

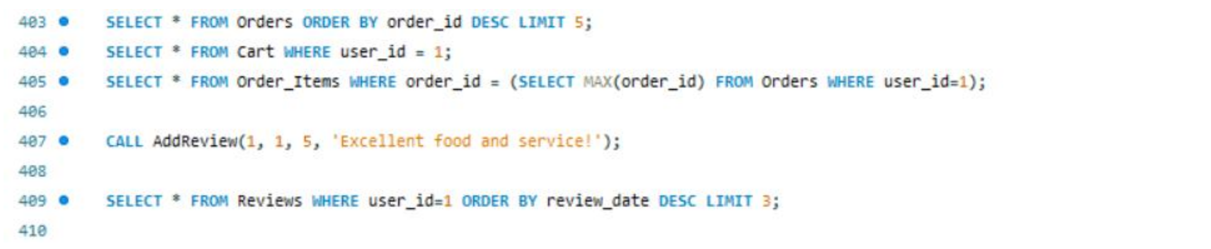
PROCEDURE



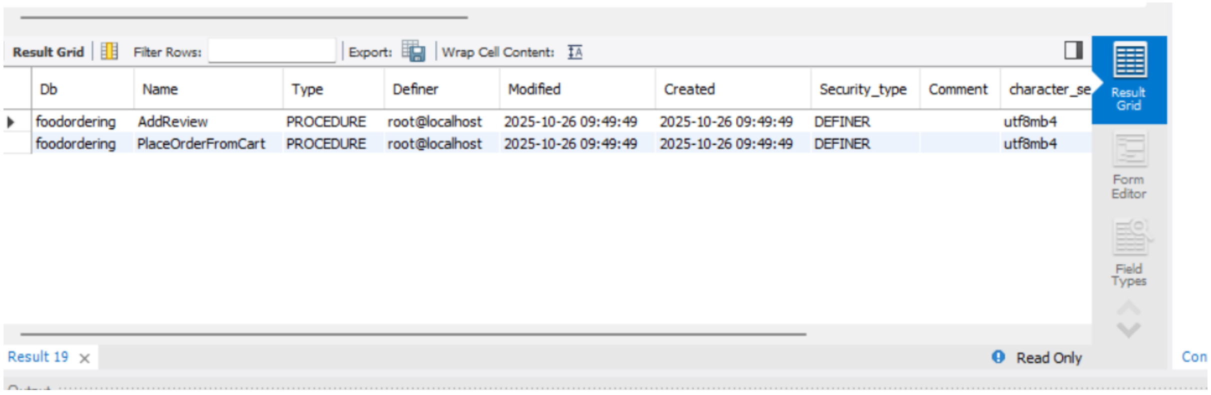
```
-- procedure
DELIMITER //
CREATE PROCEDURE PlaceOrderFromCart(IN p_user_id INT, IN p_delivery_partner_id INT)
BEGIN
    DECLARE v_order_id INT;

    -- 1. Create an order entry
    INSERT INTO Orders (user_id, total_amount, status, delivery_partner_id)
    VALUES (p_user_id, 0.00, 'Pending', p_delivery_partner_id);

    SET v_order_id = LAST_INSERT_ID();

    -- 2. Copy all cart items into Order_Items
    INSERT INTO Order_Items (order_id, menu_id, quantity)
    SELECT v_order_id, menu_id, quantity FROM Cart WHERE user_id = p_user_id;

    -- 3. Calculate total and update Orders table
    UPDATE Orders
    SET total_amount = GetOrderTotal(v_order_id)
    WHERE order_id = v_order_id;

    -- 4. Clear the user's cart
    DELETE FROM Cart WHERE user_id = p_user_id;
END;
```



```sql
DELIMITER //
CREATE PROCEDURE AddReview(
    IN p_user_id INT,
    IN p_restaurant_id INT,
    IN p_rating INT,
    IN p_comment VARCHAR(500)
)
BEGIN
    IF p_rating < 1 OR p_rating > 5 THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Rating must be between 1 and 5';
    END IF;

    INSERT INTO Reviews (user_id, restaurant_id, rating, comment)
    VALUES (p_user_id, p_restaurant_id, p_rating, p_comment);
END;
//
DELIMITER ;

SHOW PROCEDURE STATUS WHERE Db='FoodOrdering';

SELECT * FROM Orders ORDER BY order_id DESC LIMIT 5;
SELECT * FROM Cart WHERE user_id = 1;

CALL PlaceOrderFromCart(1, 1);
```

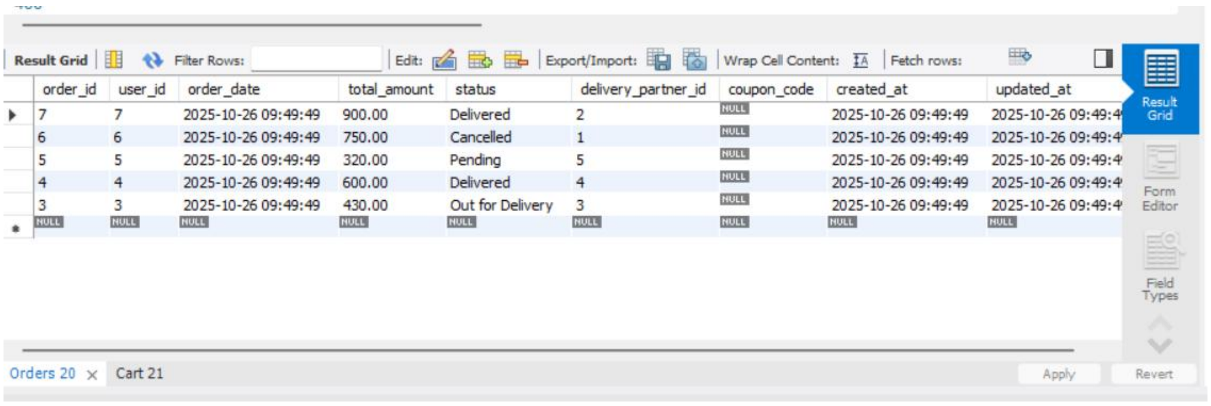


```sql
SELECT * FROM Orders ORDER BY order_id DESC LIMIT 5;
SELECT * FROM Cart WHERE user_id = 1;
SELECT * FROM Order_Items WHERE order_id = (SELECT MAX(order_id) FROM Orders WHERE user_id=1);

CALL AddReview(1, 1, 5, 'Excellent food and service!');

SELECT * FROM Reviews WHERE user_id=1 ORDER BY review_date DESC LIMIT 3;
```

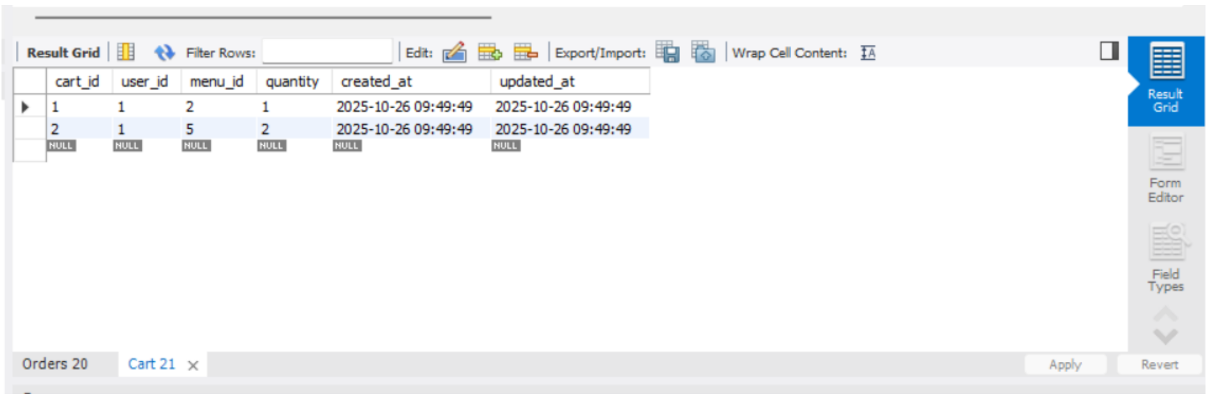


Db	Name	Type	Definer	Modified	Created	Security_type	Comment	character_se
foodordering	AddReview	PROCEDURE	root@localhost	2025-10-26 09:49:49	2025-10-26 09:49:49	DEFINER		utf8mb4
foodordering	PlaceOrderFromCart	PROCEDURE	root@localhost	2025-10-26 09:49:49	2025-10-26 09:49:49	DEFINER		utf8mb4

BEFORE

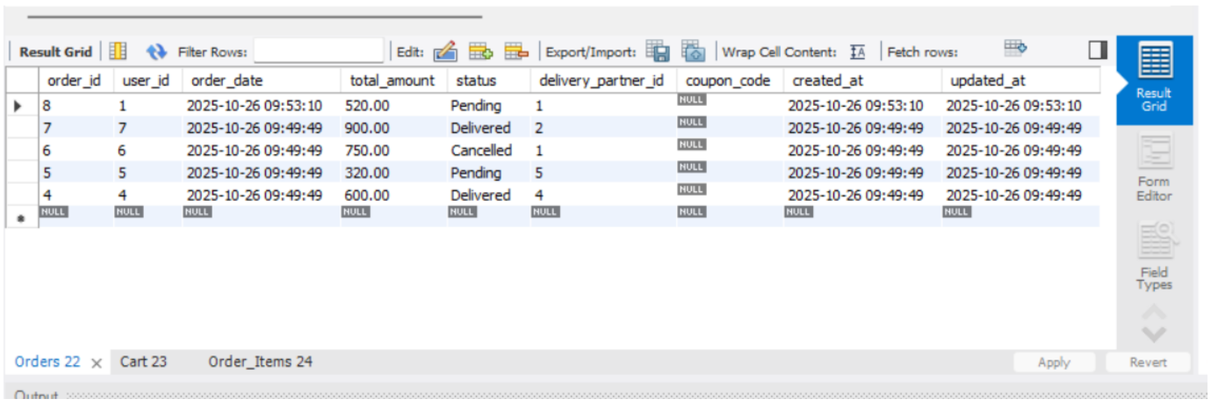


order_id	user_id	order_date	total_amount	status	delivery_partner_id	coupon_code	created_at	updated_at
7	7	2025-10-26 09:49:49	900.00	Delivered	2	NULL	2025-10-26 09:49:49	2025-10-26 09:49:4
6	6	2025-10-26 09:49:49	750.00	Cancelled	1	NULL	2025-10-26 09:49:49	2025-10-26 09:49:4
5	5	2025-10-26 09:49:49	320.00	Pending	5	NULL	2025-10-26 09:49:49	2025-10-26 09:49:4
4	4	2025-10-26 09:49:49	600.00	Delivered	4	NULL	2025-10-26 09:49:49	2025-10-26 09:49:4
3	3	2025-10-26 09:49:49	430.00	Out for Delivery	3	NULL	2025-10-26 09:49:49	2025-10-26 09:49:4
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

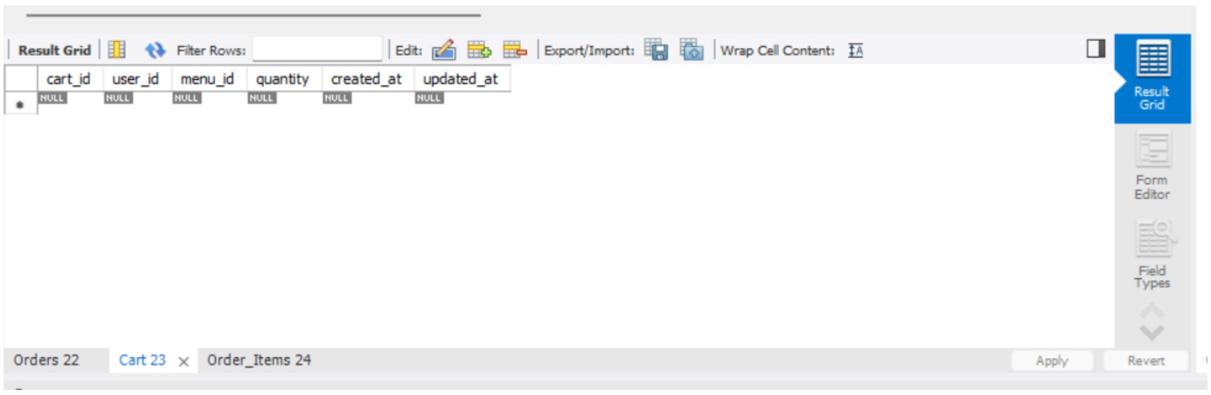


cart_id	user_id	menu_id	quantity	created_at	updated_at
1	1	2	1	2025-10-26 09:49:49	2025-10-26 09:49:49
2	1	5	2	2025-10-26 09:49:49	2025-10-26 09:49:49
NULL	NULL	NULL	NULL	NULL	NULL

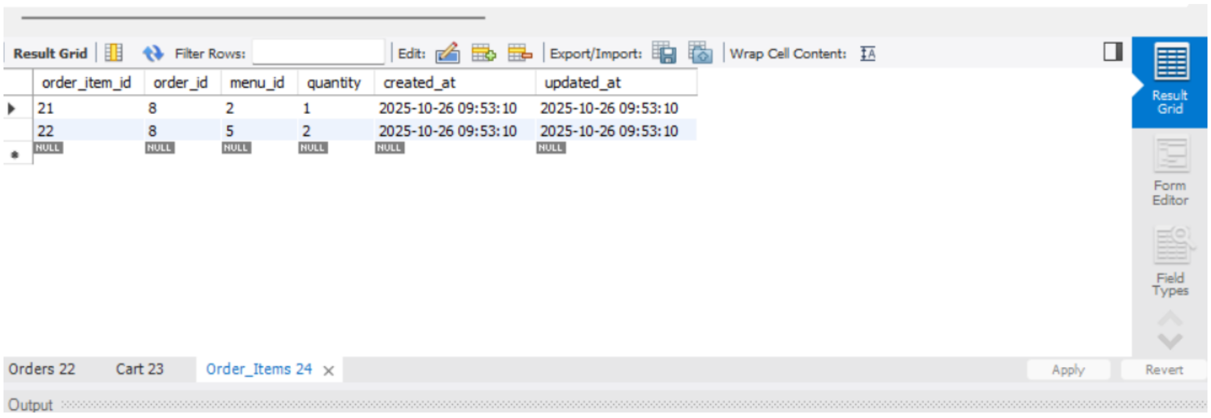
AFTER



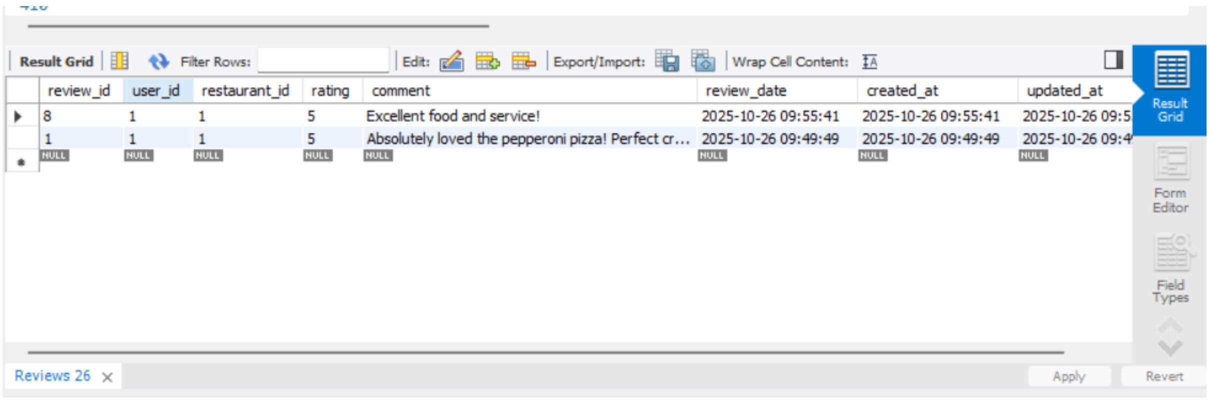
order_id	user_id	order_date	total_amount	status	delivery_partner_id	coupon_code	created_at	updated_at
8	1	2025-10-26 09:53:10	520.00	Pending	1	NULL	2025-10-26 09:53:10	2025-10-26 09:53:10
7	7	2025-10-26 09:49:49	900.00	Delivered	2	NULL	2025-10-26 09:49:49	2025-10-26 09:49:49
6	6	2025-10-26 09:49:49	750.00	Cancelled	1	NULL	2025-10-26 09:49:49	2025-10-26 09:49:49
5	5	2025-10-26 09:49:49	320.00	Pending	5	NULL	2025-10-26 09:49:49	2025-10-26 09:49:49
4	4	2025-10-26 09:49:49	600.00	Delivered	4	NULL	2025-10-26 09:49:49	2025-10-26 09:49:49
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL



cart_id	user_id	menu_id	quantity	created_at	updated_at
NULL	NULL	NULL	NULL	NULL	NULL

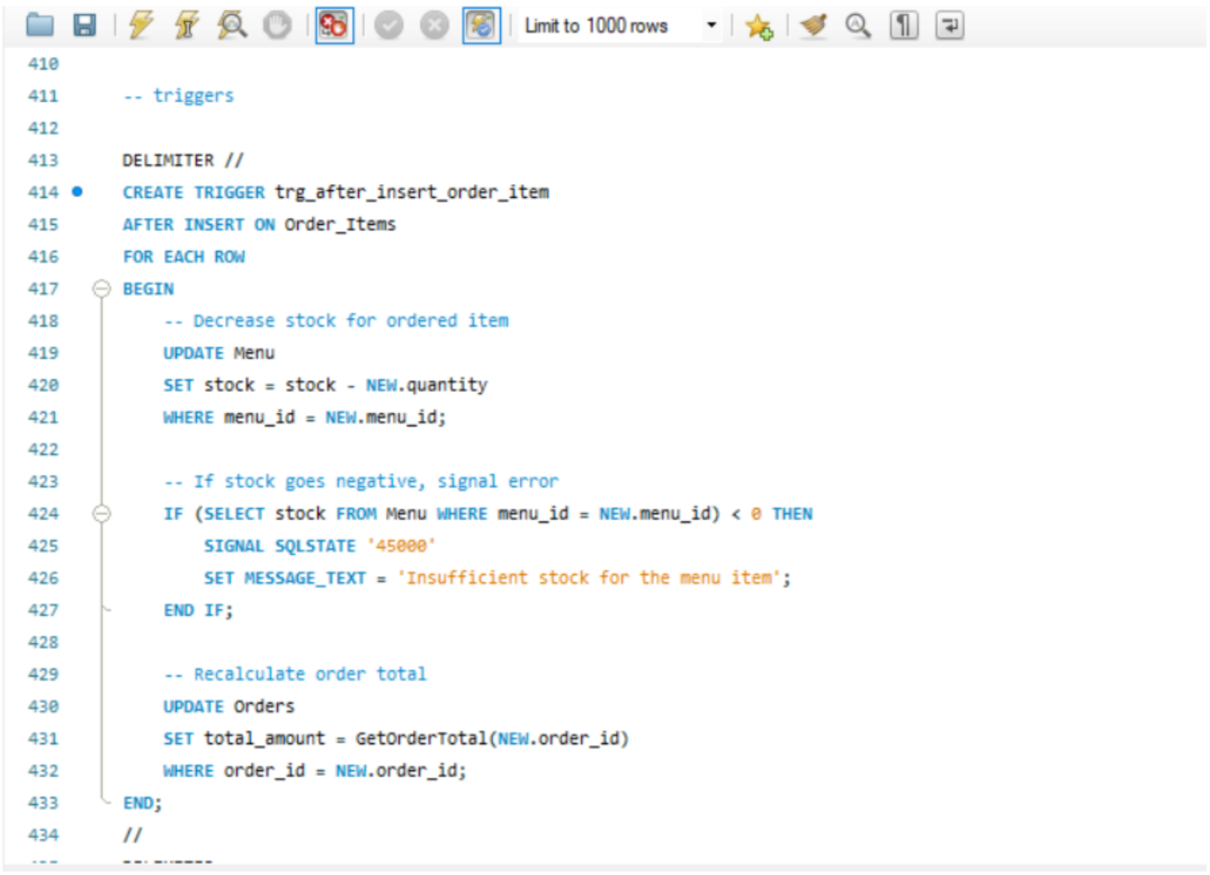


order_item_id	order_id	menu_id	quantity	created_at	updated_at
21	8	2	1	2025-10-26 09:53:10	2025-10-26 09:53:10
22	8	5	2	2025-10-26 09:53:10	2025-10-26 09:53:10
NULL	NULL	NULL	NULL	NULL	NULL

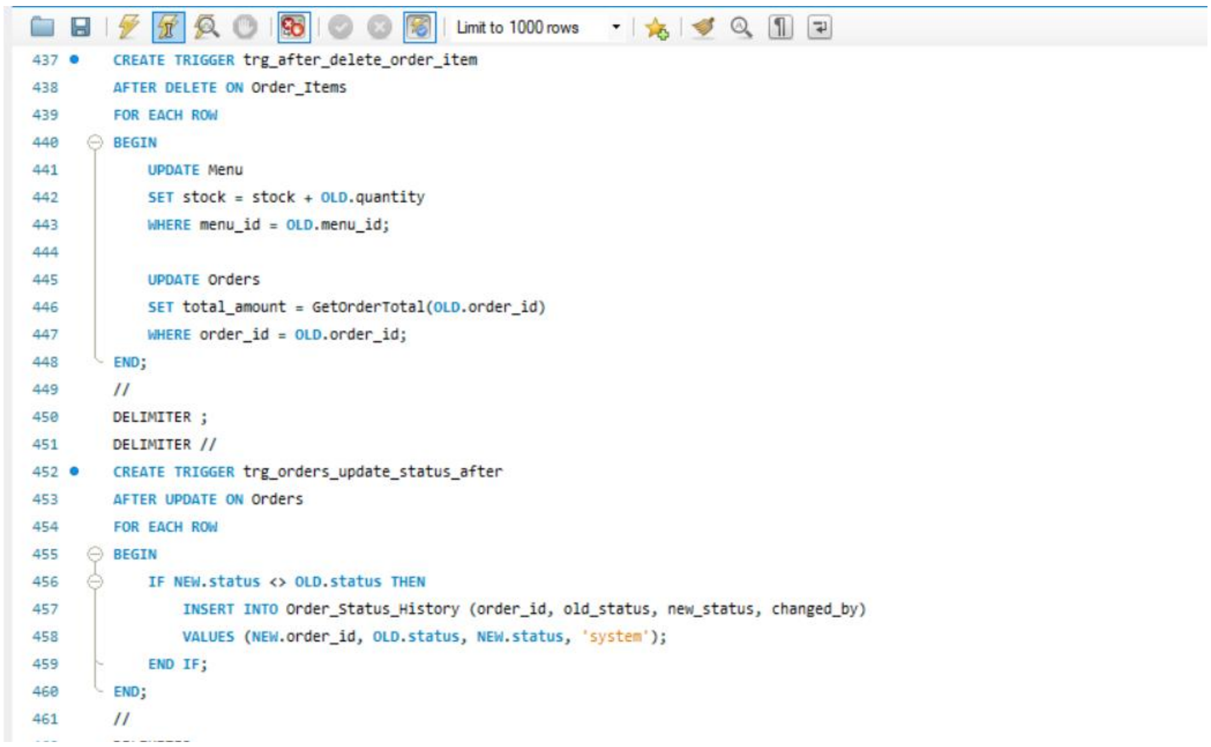


review_id	user_id	restaurant_id	rating	comment	review_date	created_at	updated_at
8	1	1	5	Excellent food and service!	2025-10-26 09:55:41	2025-10-26 09:55:41	2025-10-26 09:5
1	1	1	5	Absolutely loved the pepperoni pizza! Perfect cr...	2025-10-26 09:49:49	2025-10-26 09:49:49	2025-10-26 09:4
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

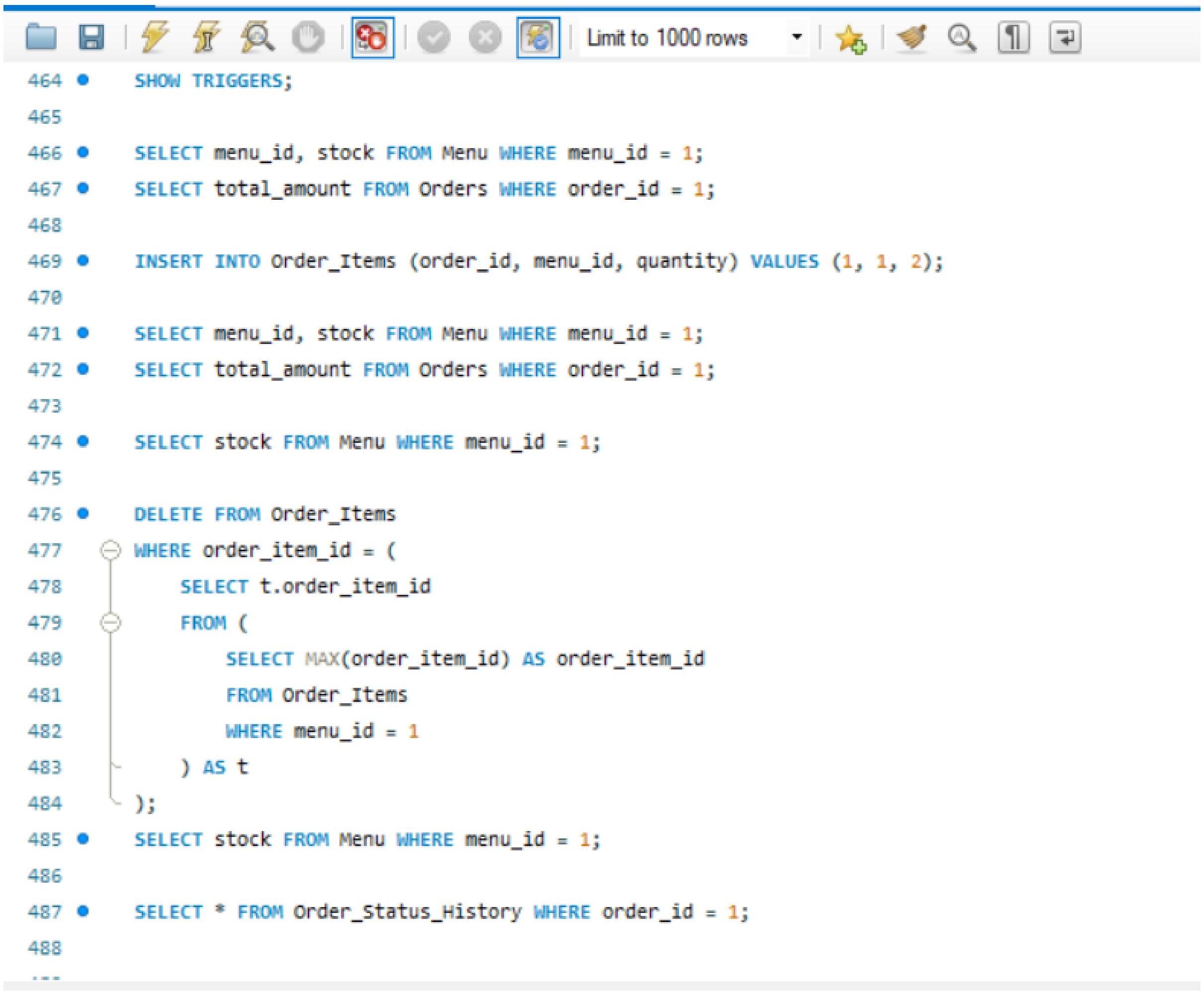
TRIGGERS:



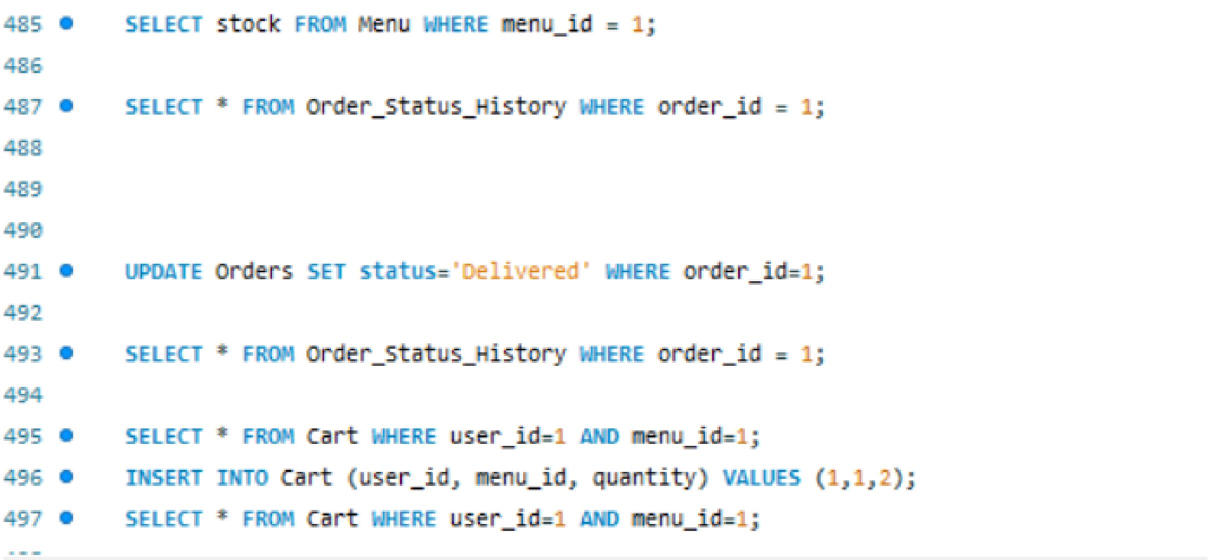
```

-- triggers

DELIMITER //
CREATE TRIGGER trg_after_insert_order_item
AFTER INSERT ON Order_Items
FOR EACH ROW
BEGIN
    -- Decrease stock for ordered item
    UPDATE Menu
    SET stock = stock - NEW.quantity
    WHERE menu_id = NEW.menu_id;

    -- If stock goes negative, signal error
    IF (SELECT stock FROM Menu WHERE menu_id = NEW.menu_id) < 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Insufficient stock for the menu item';
    END IF;

    -- Recalculate order total
    UPDATE Orders
    SET total_amount = GetOrderTotal(NEW.order_id)
    WHERE order_id = NEW.order_id;
END;
//
```



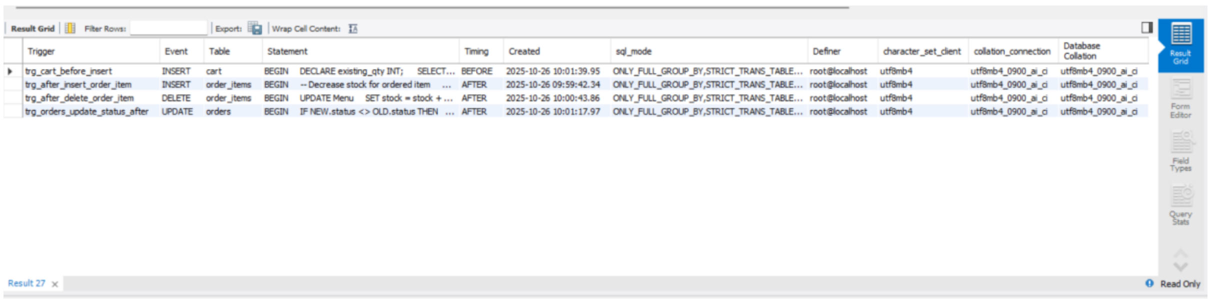
```sql
CREATE TRIGGER trg_after_delete_order_item
AFTER DELETE ON Order_Items
FOR EACH ROW
BEGIN
    UPDATE Menu
    SET stock = stock + OLD.quantity
    WHERE menu_id = OLD.menu_id;

    UPDATE Orders
    SET total_amount = GetOrderTotal(OLD.order_id)
    WHERE order_id = OLD.order_id;
END;
//
DELIMITER ;
DELIMITER //
CREATE TRIGGER trg_orders_update_status_after
AFTER UPDATE ON Orders
FOR EACH ROW
BEGIN
    IF NEW.status <> OLD.status THEN
        INSERT INTO Order_Status_History (order_id, old_status, new_status, changed_by)
        VALUES (NEW.order_id, OLD.status, NEW.status, 'system');
    END IF;
END;
//
```



```sql
SHOW TRIGGERS;

SELECT menu_id, stock FROM Menu WHERE menu_id = 1;
SELECT total_amount FROM Orders WHERE order_id = 1;

INSERT INTO Order_Items (order_id, menu_id, quantity) VALUES (1, 1, 2);

SELECT menu_id, stock FROM Menu WHERE menu_id = 1;
SELECT total_amount FROM Orders WHERE order_id = 1;

SELECT stock FROM Menu WHERE menu_id = 1;

DELETE FROM Order_Items
WHERE order_item_id = (
    SELECT t.order_item_id
    FROM (
        SELECT MAX(order_item_id) AS order_item_id
        FROM Order_Items
        WHERE menu_id = 1
    ) AS t
);
SELECT stock FROM Menu WHERE menu_id = 1;

SELECT * FROM Order_Status_History WHERE order_id = 1;

```



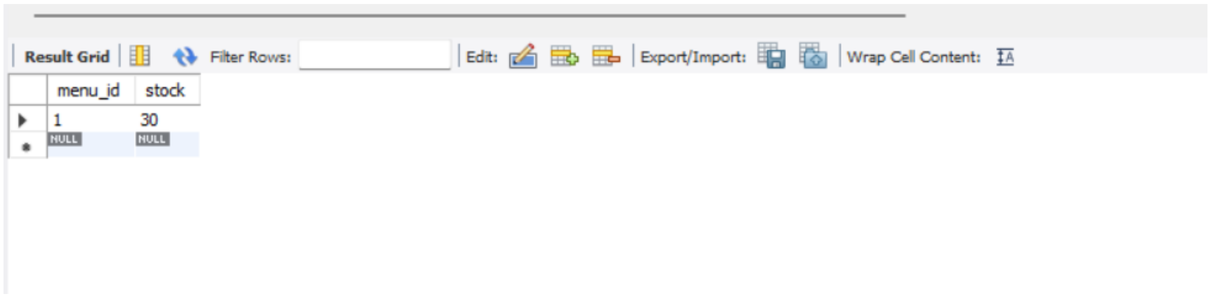
```sql
SELECT stock FROM Menu WHERE menu_id = 1;

SELECT * FROM Order_Status_History WHERE order_id = 1;



UPDATE Orders SET status='Delivered' WHERE order_id=1;

SELECT * FROM Order_Status_History WHERE order_id = 1;

SELECT * FROM Cart WHERE user_id=1 AND menu_id=1;
INSERT INTO Cart (user_id, menu_id, quantity) VALUES (1,1,2);
SELECT * FROM Cart WHERE user_id=1 AND menu_id=1;
```



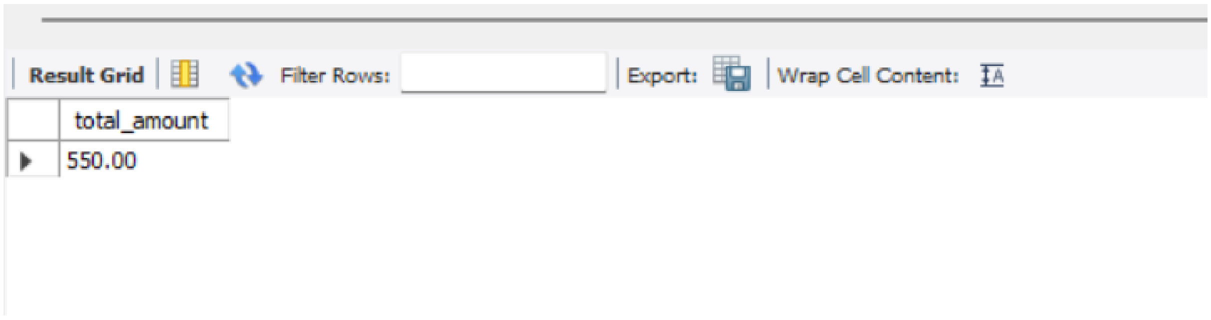
Trigger	Event	Table	Statement	Timing	Created	sql_mode	Definer	character_set_client	collation_connection	Database Collation
trg_cart_before_insert	INSERT	cart	BEGIN DECLARE existing_qty INT; SELECT...	BEFORE	2025-10-26 10:01:39.95	ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLE...	root@localhost	utf8mb4	utf8mb4_0900_ai_ci	utf8mb4_0900_ai_ci
trg_after_insert_order_item	INSERT	order_items	BEGIN -- Decrease stock for ordered item ...	AFTER	2025-10-26 09:59:42.34	ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLE...	root@localhost	utf8mb4	utf8mb4_0900_ai_ci	utf8mb4_0900_ai_ci
trg_after_delete_order_item	DELETE	order_items	BEGIN UPDATE Menu SET stock = stock + ...	AFTER	2025-10-26 10:00:43.86	ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLE...	root@localhost	utf8mb4	utf8mb4_0900_ai_ci	utf8mb4_0900_ai_ci
trg_orders_update_status_after	UPDATE	orders	BEGIN IF NEW.status <> OLD.status THEN ...	AFTER	2025-10-26 10:01:17.97	ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLE...	root@localhost	utf8mb4	utf8mb4_0900_ai_ci	utf8mb4_0900_ai_ci

BEFORE

Stock & Total Update Trigger:

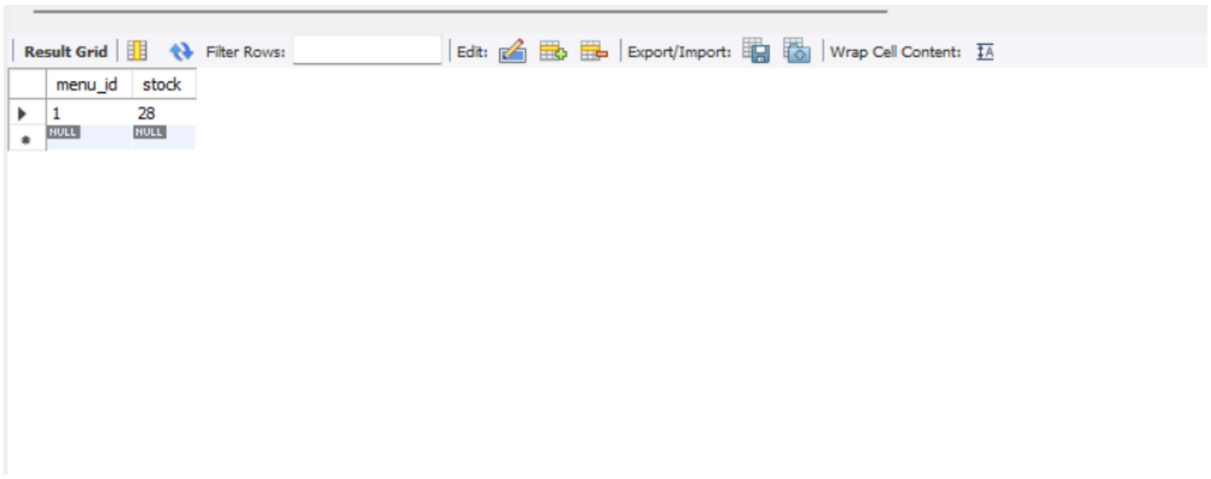


menu_id	stock
1	30
NULL	NULL

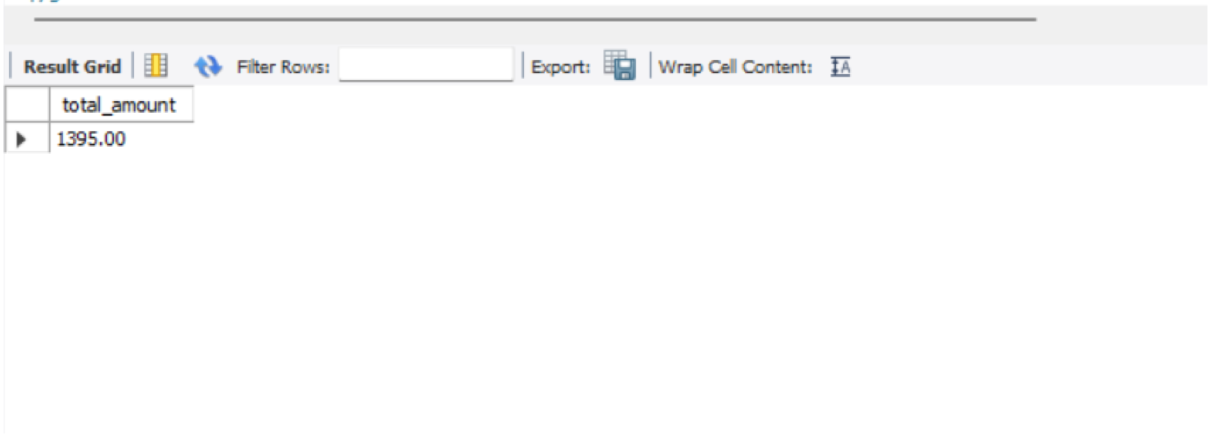


total_amount
550.00

AFTER


Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

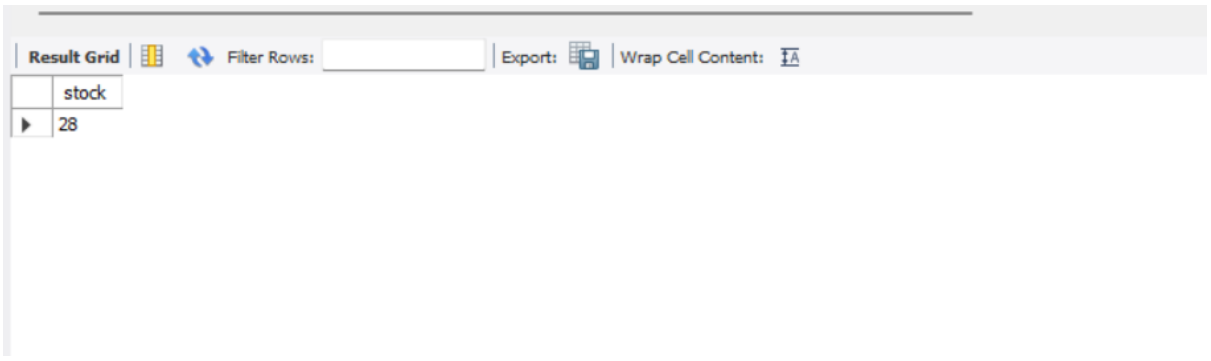
	menu_id	stock
▶	1	28
●	NULL	NULL


Result Grid | Filter Rows: | Export: | Wrap Cell Content:

	total_amount
▶	1395.00

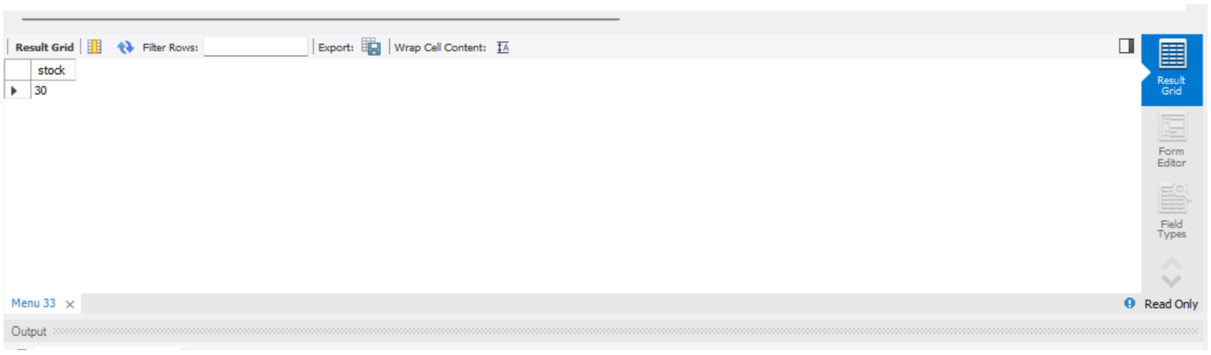
Delete Trigger

BEFORE


Result Grid | Filter Rows: | Export: | Wrap Cell Content:

	stock
▶	28

AFTER


Result Grid | Filter Rows: | Export: | Wrap Cell Content:

	stock
▶	30

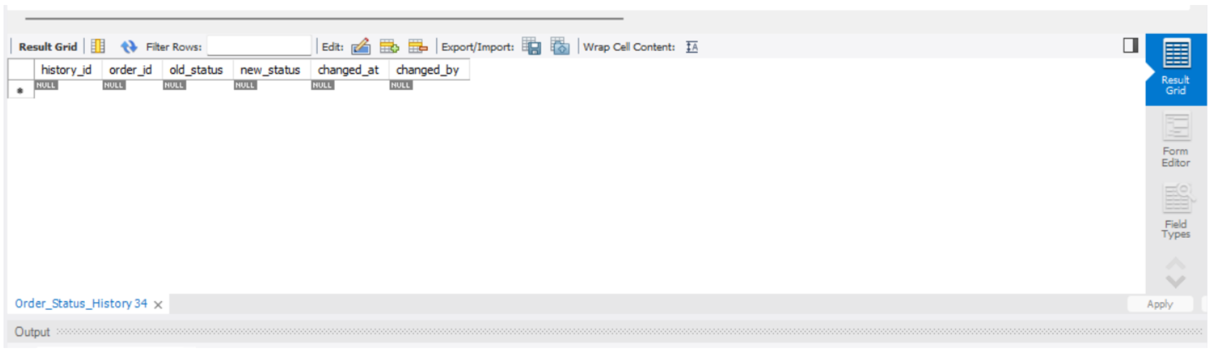
Menu 33 ×

Read Only

Output

Order Status History Trigger

BEFORE:

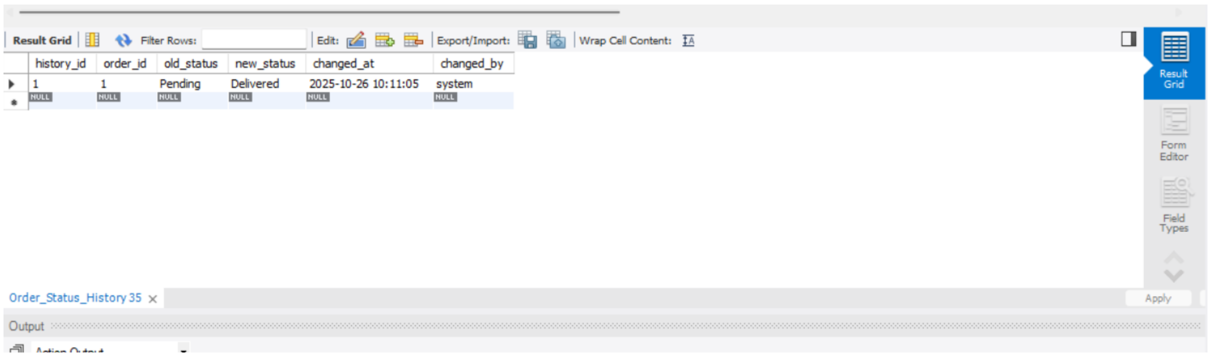

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

	history_id	order_id	old_status	new_status	changed_at	changed_by
●	NULL	NULL	NULL	NULL	NULL	NULL

Order_Status_History 34 ×

Output

AFTER:


Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

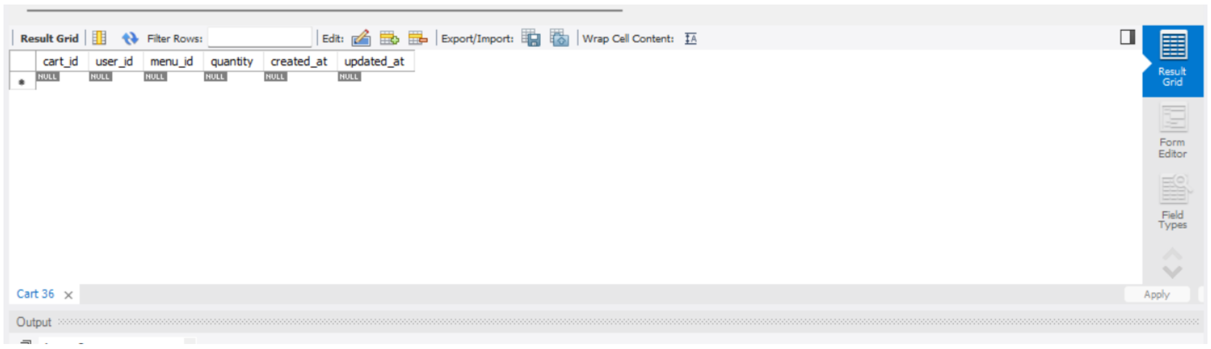
	history_id	order_id	old_status	new_status	changed_at	changed_by
▶	1	1	Pending	Delivered	2025-10-26 10:11:05	system
●	NULL	NULL	NULL	NULL	NULL	NULL

Order_Status_History 35 ×

Output

Cart Quantity Increment Trigger

BEFORE:

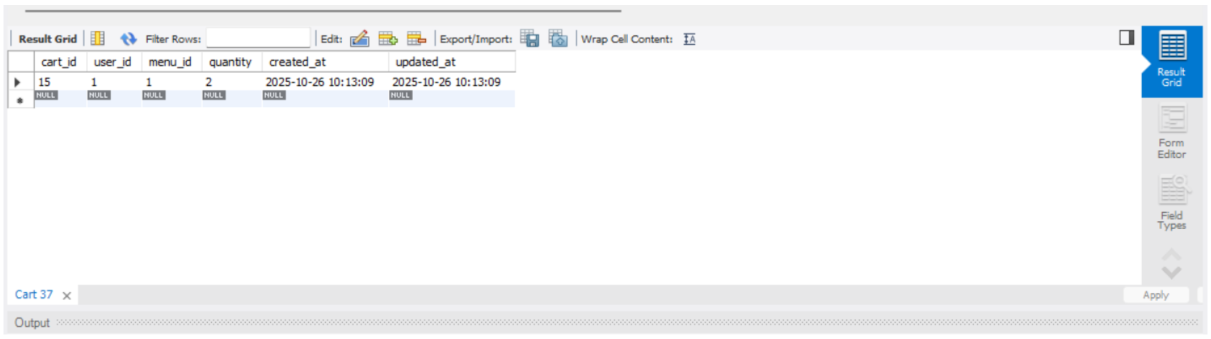

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

	cart_id	user_id	menu_id	quantity	created_at	updated_at
●	NULL	NULL	NULL	NULL	NULL	NULL

Cart 36 ×

Output

AFTER:



	cart_id	user_id	menu_id	quantity	created_at	updated_at
▶	15	1	1	2	2025-10-26 10:13:09	2025-10-26 10:13:09
*	NULL	NULL	NULL	NULL	NULL	NULL

9. GitHub repo link:

https://github.com/sirishivanand24184-png/FOOD-ORDERING-SYSTEM-.git