Group 11: Team Schema Shapers

Food Delivery Management System

Srisai Kumar Resu (Coordinator) (11744462)

Sri Santhoshi Sowmya Atreyapurapu (11803582)

Sri Anjani Mani Mounika Atreyapurapu (11803585)

Swetha Darasani (11637747)

Sri Vishnu Ravipati(11593793)

INFO 5707 - Data Modelling for Information Professionals

Dr. Tozammel Hossain

Contents

Project Description:

The **Food Delivery Management System** aims to provide a seamless experience for users to explore restaurants, make reservations, and place food orders. The system connects customers with restaurants and delivery services, allowing them to order food, book tables, browse menus, make payments, and track food deliveries in real-time. The main objective of the system is to manage business data efficiently while enhancing user experience through a smooth reservation and ordering process. The scope includes user registration, restaurant listings, menu exploration, order placement, order tracking, secure payment processing, and review functionalities. Additionally, the system provides an admin panel for restaurant owners to manage their operations, track sales, and improve their services.

Objectives:

1. **Restaurant Listings and Search**: Users can explore nearby restaurants based on location, cuisine, or ratings, with detailed information about each restaurant and its menu.
2. **Menu Exploration and Ordering**: Users will have access to detailed restaurant menus with descriptions and images of dishes, allowing them to place orders for pick-up or delivery directly from their mobile devices.
3. **Customer Registration and Authentication**: Customers will securely register and create accounts to access restaurant listings, place orders, and manage their profiles.
4. **Reservation Management**: Users can book tables at their preferred restaurants, with details including reservation date, time, and the number of people in their party, stored in the Reservation table. Each reservation is linked to a customer and a restaurant, ensuring accurate tracking and verification of reservations.
5. **Secure Payment Processing**: The system will support multiple payment methods, including credit/debit cards, digital wallets, and cash on delivery, ensuring secure transactions. The Payment table will store transaction details, such as payment date, amount, method, and status, ensuring secure and organized payment records linked to each Order.
6. **Account and Order History Management**: Users will be able to manage their profile information, view their order history, and save favourite restaurants, menu items, and delivery addresses for future convenience.
7. **Review and Rating System**: Customers can provide feedback on restaurants and dishes, enabling others to make informed decisions based on ratings and reviews.
8. **Order Tracking**: Real-time tracking of food orders will be provided, allowing customers to monitor delivery times and receive updates about the status of their orders.

Scope of the Database:

The scope of the database includes the following key areas:

1. **User Management**:
    - Store user details such as names, contact information, addresses, and account type.
2. **Restaurant Management**:
    - Store information about restaurants, including names, locations, contact details, type and cuisine type.
3. **Menu Management**:
    - Manage restaurant menus, including dish names, descriptions, prices, and availability.
    - Maintain detailed information about cuisines and menu categories (e.g., appetizers, main courses, desserts).
4. **Reservation Management**:
    - Record table reservations made by users, including restaurant details, reservation date, time, number of guests.
5. **Order and Delivery Management**:
    - Track food orders placed by customers, including order details, selected menu items, quantities, and prices.
    - Monitor delivery statuses (e.g., placed, prepared, out for delivery, delivered).
6. **Delivery Management**:
    - Manage delivery information such as delivery status, and date. This table links back to the Order table to track the real-time status of each order until it is delivered.
7. **Payment Processing**:
    - Store payment details for orders, including payment methods (credit/debit card, digital wallet, cash on delivery).
    - Maintain records of transactions and completed payments for financial tracking.

User Requirements:

1. The system will generate a unique ID for each customer and store their information, including contact details and preferences.
2. Customers can update their profiles with the latest information, such as delivery addresses and payment details.
3. A unique order ID will be generated for each reservation or food order, tracking items, quantities, order status, and delivery details.
4. Customers should be able to view detailed restaurant information, including contact details, ratings, and available cuisines.
5. Customers should be able to view the menu of a selected restaurant, with details of each dish including name, description, price, and type (e.g., appetizer, main course, dessert).
6. Customers will have access to their order history and can view details of reservations, orders, and payments.
7. Real-time order tracking will be available, allowing customers to follow the status of their orders until delivery.
8. Restaurant owners can manage their menus, reservations, and order statuses through an admin panel.
9. Restaurant staff will update order statuses after food has been delivered.
10. Restaurant staff will accept the reservation bookings based on availability.
11. Payment processing will be secure, supporting multiple payment options, including credit/debit cards and digital wallets and cash on delivery.
12. Multiple users, including restaurant staff and customers, can access the system simultaneously, with access based on their roles.

Business Rules:

1. Every customer has a unique ID.
2. Customer information must be complete before an order can be accepted.
3. The customer’s email address and phone number must be unique.
4. Accounts can only be associated with one customer.
5. Customers can create an account without placing an order.
6. Every order should have a unique order number.
7. Multiple items can be included in one order.
8. Every order must be associated with one and only one customer.
9. An order can only be associated with a restaurant that is in the system.
10. Each order must have a payment record, and the payment status should be marked as "Paid" for the order to be processed for delivery.
11. Each order can have only one delivery entry.
12. An order marked as "Delivered" in delivery status cannot be updated to any other status.
13. The delivery date must be on or after the order date.
14. A reservation must include a valid customer and restaurant.
15. Each reservation must specify a reservation date and the number of people.
16. Customers can only make one reservation per restaurant per day.
17. Reservations should be made at least 24 hours in advance of the reservation date.
18. Reservation is limited to a maximum of 20 people per customer.

19. There is no limitation on the number of items in an order.
20. Every item has a unique ID.
21. Item type should be from these Appetizers, Main Courses, Desserts, Beverages, Salads, Sides, Soups and Specials.
22. Item description must contain ingredients, calories, and serving size.
23. Menu items must have an item price greater than zero.
24. Each restaurant has a unique ID, email and contact.
25. The restaurant rating is between 1-5 stars.
26. Cuisine type should tell about type of cuisine the restaurant serves like Indian, Italian, Chinese, Mexican, French, Japanese, Mediterranean, Thai, American, Middle Eastern, Korean, and Vietnamese.

Project Requirements:

Operating System: Windows/MacOS

Database: PostgreSQL

Applications: PGAdmin 4, Lucid Chart, Microsoft Excel, Microsoft Word.

Database Tables:

Here's a list of tables with a brief description for each, based on the provided scope and requirements:

1. **Customer** - Stores user information including contact details and addresses.
2. **Restaurant** - Contains information about restaurants, such as name, location, cuisine type, and contact info.
3. **Menu** - Holds details of restaurant menus, including dish names, descriptions, prices, and types.
4. **Order** - Tracks customer orders with details about items ordered, quantities, and prices.
5. **Delivery** - Manages delivery information, such as order delivery status, delivery dates, and payment type.
6. **Payment** - Stores payment transaction details, including method, amount, and payment status.
7. **Reservation** - Records table reservations with reservation dates, times, and guest counts.

Entity Relationship Diagram:

Relationship:

Entity	Related Entity	Relationship Type
Customer	Order	One-to-Many (1-M)
Customer	Reservation	One-to-Many (1-M)
Restaurant	Menu	One-to-Many (1-M)
Restaurant	Order	One-to-Many (1-M)
Restaurant	Reservation	One-to-Many (1-M)
Order	Payment	One-to-One (1:1)
Order	Delivery	One-to-One (1:1)
Reservation	Customer	Many-to-One (M:1)
Reservation	Restaurant	Many-to-One (M:1)

Data Dictionary:

Tabl+E15:L53e Name	Attribute Name	Description	Data Type	Data Format	Required	PK or FK	Example
Customer	customerId	Unique identifier for each customer	INT	999999	Yes	PK	1001
	name	Customer's full name	VARCHAR(100)	XXXXXXXXX	Yes		Tony Stark
	address	Customer's address	VARCHAR(255)	XXXXXXXXXXXXX	Yes		123 Elm Street
	contactNumber	Customer's contact phone number	VARCHAR(15)	XXX-XXX-XXXX	Yes		123-456-7890
	email	Customer's email address	VARCHAR(100)	XXXXX@XXXX.com	Yes		tony@example.com
Restaurant	restaurantId	Unique identifier for each restaurant	INT	999999	Yes	PK	3001
	restaurantName	Name of the restaurant	VARCHAR(100)	XXXXXXXXX	Yes		Food Place
	restaurantAddress	Address of the restaurant	VARCHAR(255)	XXXXXXXXXXXXX	Yes		456 Food Lane
	restaurantRating	Rating of the restaurant	DECIMAL(2,1)	X.X	No		4.5
	restaurantContact	Contact number of the restaurant	VARCHAR(15)	XXX-XXX-XXXX	Yes		987-654-3210
	restaurantEmail	Email address of the restaurant	VARCHAR(100)	XXXXX@XXXX.com	Yes		contact@food.com
	cuisineType	Cuisine type offered by the restaurant	VARCHAR(50)	XXXXXXXXX	Yes		Italian
Menu	itemId	Unique identifier for each menu item	INT	999999	Yes	PK	6001
	restaurantId	ID of the restaurant offering the item	INT	999999	Yes	FK	3001
	itemName	Name of the menu item	VARCHAR(100)	XXXXXXXXX	Yes		Pasta
	itemDescription	Description of the menu item	TEXT	XXXXXXXXXXXXX	No		Creamy Alfredo
	itemPrice	Price of the menu item	DECIMAL(10,2)	XXXXXX.XX	Yes		12.99
	itemType	Type of item (e.g., Appetizer)	VARCHAR(50)	XXXXXXXXX	Yes		Main Course
Order	orderId	Unique identifier for each order	INT	999999	Yes	PK	2001
	customerId	ID of the customer who placed the order	INT	999999	Yes	FK	1001
	restaurantId	ID of the restaurant for the order	INT	999999	Yes	FK	3001
	orderDate	Date when the order was placed	DATE	YYYY-MM-DD	Yes		28/10/24
Reservation	reservationId	Unique identifier for each reservation	INT	999999	Yes	PK	7001
	customerId	ID of the customer making the reservatio	INT	999999	Yes	FK	1001
	restaurantId	ID of the restaurant for reservation	INT	999999	Yes	FK	3001
	reservationDate	Date of the reservation	DATE	YYYY-MM-DD	Yes		05/11/24
	numberOfPeople	Number of people for the reservation	INT	XX	Yes		4
Payment	paymentId	Unique identifier for each payment	INT	999999	Yes	PK	4001
	orderId	ID of the order associated with payment	INT	999999	Yes	FK	2001
	transactionId	Unique identifier for the transaction	VARCHAR(50)	XXXXXXXXXXXXX	Yes		TXN12345
	paymentDate	Date when the payment was made	DATE	YYYY-MM-DD	Yes		28/10/24
	totalAmount	Total amount paid	DECIMAL(10,2)	XXXXXX.XX	Yes		150
	paymentMethod	Method of payment (e.g., Card, Cash)	VARCHAR(20)	XXXXXXXXX	Yes		Card
	paymentStatus	Status of payment (e.g., Paid, Pending)	VARCHAR(20)	XXXXXXXXX	Yes		Paid
Delivery	deliveryId	Unique identifier for each delivery	INT	999999	Yes	PK	5001
	orderId	ID of the associated order	INT	999999	Yes	FK	2001
	deliveryStatus	Status of the delivery	VARCHAR(20)	XXXXXXXXX	Yes		Delivered
	deliveryDate	Date of delivery	DATE	YYYY-MM-DD	Yes		29/10/24

Queries and Operations

Section 1: Entity Generation and Data Entry: Create Database, Create Tables, and Insert Values to the Tables

Overview:

Our Restaurant Management data model consists of a single database. The table below lists the database schema for all the tables. Foreign key constraints are applied when the table is created to avoid inserting invalid data. The value of the foreign key in the child table must be one of the values from the primary key in the parent table. Therefore, the order for inserting values into the tables is vital to mitigate any errors when running the SQL queries.

Database Schema:

Insert Order	Table	Dependency	Reason for Order	Records
1	Customer	None	Independent table; no dependencies.	30
2	Restaurant	None	Independent table; no dependencies.	30
3	Menu	Restaurant	Depends on restaurantId from the Restaurant table to associate menu items with restaurants.	60
4	Order	Customer, Restaurant	Depends on customerId from the Customer table and restaurantId from the Restaurant table for placing orders.	30
5	Reservation	Customer, Restaurant	Depends on customerId and restaurantId for reservations.	30
6	Payment	Order	Depends on orderId from the Order table to record payments.	30
7	Delivery	Order	Depends on orderId from the Order table to track the delivery status of orders.	30

Database and Tables Creation:

```
-- Create the database
CREATE DATABASE FoodDeliveryManagementSystem;

-- Use the database
\c FoodDeliveryManagementSystem;

-- Create Customer table
CREATE TABLE Customer (
    customerId INT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    address VARCHAR(255) NOT NULL,
    contactNumber VARCHAR(15) NOT NULL,
    email VARCHAR(100) NOT NULL
);

-- Create Restaurant table
CREATE TABLE Restaurant (
    restaurantId INT PRIMARY KEY,
    restaurantName VARCHAR(100) NOT NULL,
    restaurantAddress VARCHAR(255) NOT NULL,
    restaurantRating DECIMAL(2,1),
    restaurantContact VARCHAR(15) NOT NULL,
    restaurantEmail VARCHAR(100) NOT NULL,
    cuisineType VARCHAR(50) NOT NULL
```

```
);

-- Create Menu table
CREATE TABLE Menu (
    itemId INT PRIMARY KEY,
    restaurantId INT REFERENCES Restaurant(restaurantId),
    itemName VARCHAR(100) NOT NULL,
    itemDescription TEXT,
    itemPrice DECIMAL(10,2) NOT NULL,
    itemType VARCHAR(50) NOT NULL
);

-- Create Order table
CREATE TABLE OrderTable (
    orderId INT PRIMARY KEY,
    customerId INT REFERENCES Customer(customerId),
    restaurantId INT REFERENCES Restaurant(restaurantId),
    orderDate DATE NOT NULL
);

-- Create Reservation table
CREATE TABLE Reservation (
    reservationId INT PRIMARY KEY,
    customerId INT REFERENCES Customer(customerId),
    restaurantId INT REFERENCES Restaurant(restaurantId),
```

```
    reservationDate DATE NOT NULL,
    numberOfPeople INT NOT NULL
);

-- Create Payment table
CREATE TABLE Payment (
    paymentId INT PRIMARY KEY,
    orderId INT REFERENCES OrderTable(orderId),
    transactionId VARCHAR(50) NOT NULL,
    paymentDate DATE NOT NULL,
    totalAmount DECIMAL(10,2) NOT NULL,
    paymentMethod VARCHAR(20) NOT NULL,
    paymentStatus VARCHAR(20) NOT NULL
);

-- Create Delivery table
CREATE TABLE Delivery (
    deliveryId INT PRIMARY KEY,
    orderId INT REFERENCES OrderTable(orderId),
    deliveryStatus VARCHAR(20) NOT NULL,
    deliveryDate DATE NOT NULL
);
```

Entity Generation and Data Entry for Table Customer:

Statements Explanation:

- The database FoodDeliveryManagementSystem is already created. Command Use FoodDeliveryManagementSystem is used to call the database.
- Then, the table Customer is created in the FoodDeliveryManagementSystem database. Command CREATE TABLE CUSTOMER is used to create the table.
- Next, table Customer is filled with the relevant data using the command INSERT INTO.
- Finally, the result will be displayed by using the command SELECT * FROM Customer to query all the inserted values into the table.

Query:

INSERT INTO Customer (customerId, name, address, contactNumber, email) VALUES

(1001, 'Tony Stark', '123 Elm Street', '123-456-7890', 'tony@example.com'),

(1002, 'Steve Rogers', '456 Shield Ave', '987-654-3210', 'steve@example.com'),

(1003, 'Bruce Banner', '789 Gamma Rd', '654-987-1230', 'bruce@example.com'),

(1004, 'Natasha Romanoff', '321 Spy Lane', '543-210-9876', 'natasha@example.com'),

(1005, 'Clint Barton', '101 Archer St', '555-555-5555', 'clint@example.com'),

(1006, 'Thor Odinson', '500 Thunder Rd', '999-888-7777', 'thor@example.com'),

(1007, 'Peter Parker', '20 Queens Blvd', '888-777-6666', 'peter@example.com'),

(1008, 'Wanda Maximoff', '15 Scarlet Ln', '333-333-4444', 'wanda@example.com'),

(1009, 'Vision', '42 Vibranium Rd', '222-222-5555', 'vision@example.com'),

(1010, 'Stephen Strange', '177A Bleecker St', '111-111-1234', 'strange@example.com'),

(1011, 'Sam Wilson', '98 Falcon Rd', '321-654-9870', 'sam@example.com'),

(1012, 'Bucky Barnes', '34 Winter Soldier Ave', '432-987-1234', 'bucky@example.com'),

(1013, 'Scott Lang', '10 Ant Hill', '543-321-7890', 'scott@example.com'),

(1014, 'Hope Van Dyne', '88 Wasp Ln', '654-123-4321', 'hope@example.com'),

(1015, 'Carol Danvers', '99 Captain Rd', '765-987-3456', 'carol@example.com'),

(1016, 'Nick Fury', '7 S.H.I.E.L.D. HQ', '123-123-1234', 'nick@example.com'),

(1017, 'T’Challa', '1 Wakanda Ave', '999-999-1234', 'tchalla@example.com'),

(1018, 'Shuri', '2 Wakanda Tech St', '888-888-1234', 'shuri@example.com'),

(1019, 'Loki Laufeyson', '10 Mischief Ln', '777-777-4321', 'loki@example.com'),

(1020, 'Gamora', '30 Titan Way', '666-666-1234', 'gamora@example.com'),

(1021, 'Star-Lord', '50 Galaxy Rd', '555-555-4321', 'starlord@example.com'),

(1022, 'Drax', '25 Destroyer Rd', '444-444-1234', 'drax@example.com'),

(1023, 'Rocket Raccoon', '100 Trash Panda Ln', '333-333-1234', 'rocket@example.com'),

(1024, 'Groot', '200 Tree Rd', '222-222-4321', 'groot@example.com'),

(1025, 'Nebula', '300 Revenge Blvd', '111-111-1111', 'nebula@example.com'),

(1026, 'Mantis', '400 Empath Rd', '123-123-9999', 'mantis@example.com'),

(1027, 'Yondu', '500 Ravager Rd', '987-987-6543', 'yondu@example.com'),

(1028, 'Hawkeye', '600 Bow St', '888-888-6666', 'hawkeye@example.com'),

(1029, 'Ant-Man', '50 Smallville', '654-654-7654', 'antman@example.com'),

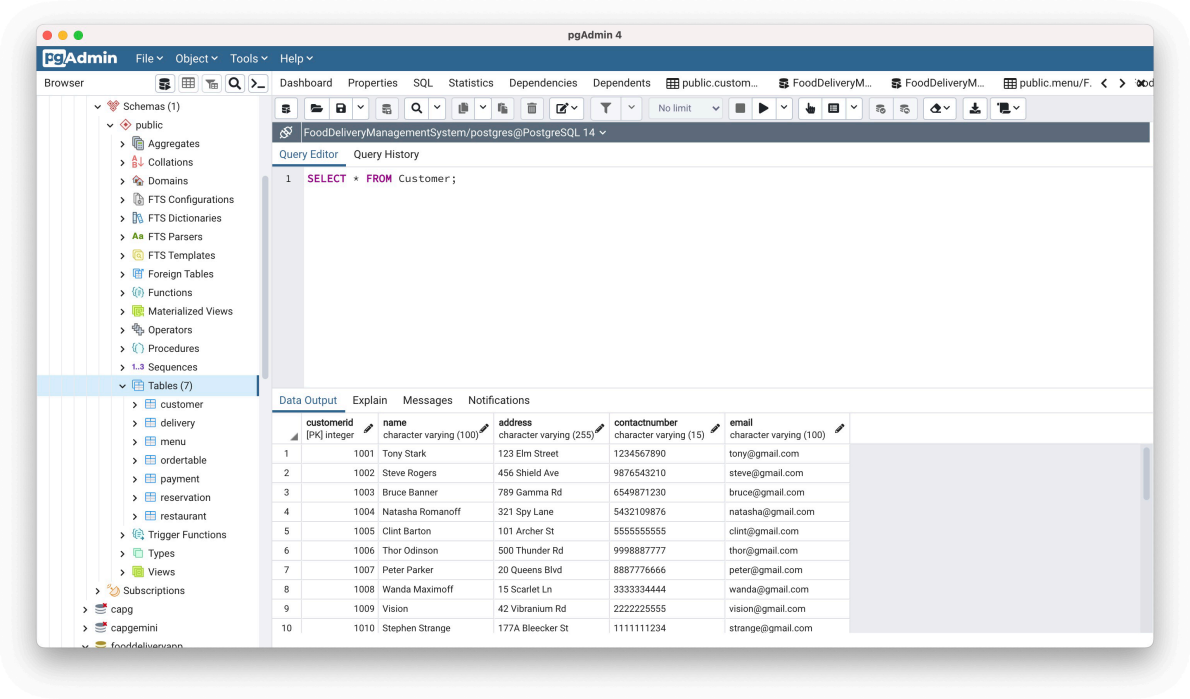
(1030, 'Black Panther', '1 Vibranium St', '567-567-5678', 'blackpanther@example.com');

Result:

Table Explanation: - Table Customer stores user information including contact details and addresses.

Query: SELECT * FROM Customer;

Screenshot:



Entity Generation and Data Entry for Table Restaurant:

Statements Explanation:

- The database FoodDeliveryManagementSystem is already created. Command Use FoodDeliveryManagementSystem is used to call the database.
- Then, the table Restaurant is created in the FoodDeliveryManagementSystem database. Command CREATE TABLE RESTAURANT is used to create the table.
- Next, table Restaurant is filled with the relevant data using the command INSERT INTO.
- Finally, the result will be displayed by using the command SELECT * FROM Restaurant to query all the inserted values into the table.

INSERT INTO Restaurant (restaurantId, restaurantName, restaurantAddress, restaurantRating, restaurantContact, restaurantEmail, cuisineType) VALUES

(3001, 'Food Paradise', '101 Main Street', 4.5, '111-111-1111', 'paradise@food.com', 'Italian'),

(3002, 'Spice Garden', '202 Spice Avenue', 4.3, '222-222-2222', 'spicegarden@food.com', 'Indian'),

(3003, 'Burger Hub', '303 Burger Lane', 4.0, '333-333-3333', 'burgerhub@food.com', 'American'),

(3004, 'Pizza Planet', '404 Cheese Street', 4.8, '444-444-4444', 'pizzaplanet@food.com', 'Italian'),

(3005, 'Taco Town', '505 Fiesta Road', 4.2, '555-555-5555', 'tacotown@food.com', 'Mexican'),

(3006, 'Noodle House', '606 Chopstick Lane', 4.1, '666-666-6666', 'noodlehouse@food.com', 'Chinese'),

(3007, 'Sushi World', '707 Wasabi Drive', 4.7, '777-777-7777', 'sushiworld@food.com', 'Japanese'),

(3008, 'Curry Corner', '808 Spice Blvd', 4.4, '888-888-8888', 'currycorner@food.com', 'Indian'),

(3009, 'Grill Station', '909 BBQ Street', 4.3, '999-999-9999', 'grillstation@food.com', 'Barbecue'),

(3010, 'Vegan Delight', '1010 Green Avenue', 4.6, '101-101-1010', 'vegandelight@food.com', 'Vegan'),

(3011, 'Pasta Palace', '1111 Italy Street', 4.7, '202-202-2020', 'pastapalace@food.com', 'Italian'),

(3012, 'The Salad Bar', '1212 Healthy Road', 4.2, '303-303-3030', 'saladbar@food.com', 'Healthy'),

(3013, 'Steakhouse Grill', '1313 Tenderloin Blvd', 4.5, '404-404-4040', 'steakhouse@food.com', 'Steakhouse'),

(3014, 'Breakfast Bliss', '1414 Sunrise Lane', 4.3, '505-505-5050', 'breakfastbliss@food.com', 'Breakfast'),

(3015, 'Falafel King', '1515 Pita Street', 4.4, '606-606-6060', 'falafelking@food.com', 'Middle Eastern'),

(3016, 'Seafood Shack', '1616 Ocean Drive', 4.6, '707-707-7070', 'seafoodshack@food.com', 'Seafood'),

(3017, 'Dessert Haven', '1717 Sweet Street', 4.9, '808-808-8080', 'desserthaven@food.com', 'Desserts'),

(3018, 'BBQ Bliss', '1818 Smoke Lane', 4.2, '909-909-9090', 'bbqbliss@food.com', 'Barbecue'),

(3019, 'Pho Place', '1919 Noodle Road', 4.4, '111-222-3333', 'phoplace@food.com', 'Vietnamese'),

(3020, 'Crepe Corner', '2020 French Lane', 4.3, '444-555-6666', 'crepecorner@food.com', 'French'),

(3021, 'Waffle Wonders', '2121 Breakfast Blvd', 4.8, '777-888-9999', 'wafflewonders@food.com', 'Breakfast'),

(3022, 'Dim Sum Delight', '2222 Dumpling Drive', 4.5, '888-999-0000', 'dimsumdelight@food.com', 'Chinese'),

(3023, 'Ramen Republic', '2323 Noodle Lane', 4.6, '111-333-4444', 'ramenrepublic@food.com', 'Japanese'),

(3024, 'Burger Barn', '2424 Patty Place', 4.0, '222-444-5555', 'burgerbarn@food.com', 'American'),

(3025, 'Tikka Junction', '2525 Spice Blvd', 4.3, '333-555-6666', 'tikkajunction@food.com', 'Indian'),

(3026, 'Gourmet Grains', '2626 Whole Grain Road', 4.4, '444-666-7777', 'gourmetgrains@food.com', 'Healthy'),

(3027, 'Fusion Flavors', '2727 Mix Street', 4.7, '555-777-8888', 'fusionflavors@food.com', 'Fusion'),

(3028, 'Biryani Bazaar', '2828 Rice Avenue', 4.6, '666-888-9999', 'biryanibazaar@food.com', 'Indian'),

(3029, 'Café Paris', '2929 Eiffel Lane', 4.9, '777-999-1111', 'cafeparis@food.com', 'French'),

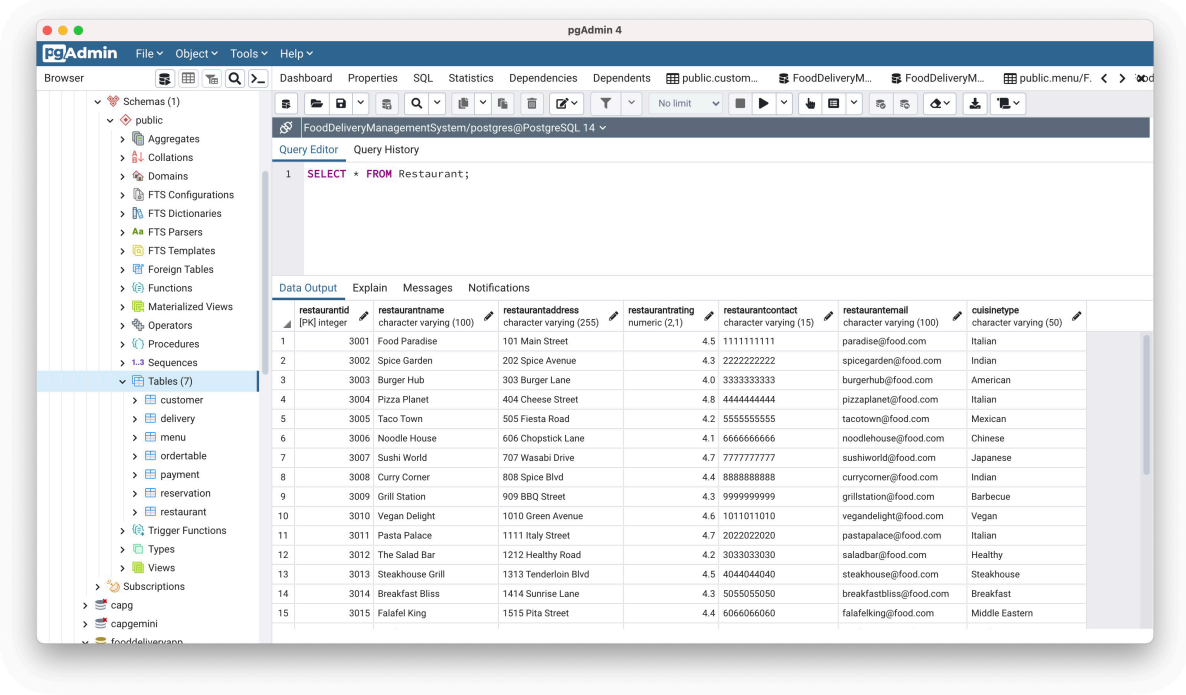
(3030, 'Kebab Kitchen', '3030 Grill Lane', 4.4, '888-000-2222', 'kebabkitchen@food.com', 'Middle Eastern');

Result:

Table Explanation: - Table stores all the information about restaurants, such as name, location, cuisine type, and contact info.

Query: SELECT * FROM Restaurant;

Screenshot:



Entity Generation and Data Entry for Table Menu:

Statements Explanation:

- The database FoodDeliveryManagementSystem is already created. Command Use FoodDeliveryManagementSystem is used to call the database.
- Then, the table Menu is created in the FoodDeliveryManagementSystem database. Command CREATE TABLE MENU is used to create the table.
- Next, table Menu is filled with the relevant data using the command INSERT INTO.
- Finally, the result will be displayed by using the command SELECT * FROM Menu to query all the inserted values into the table.

Query:

```
INSERT INTO Menu (itemId, restaurantId, itemName, itemDescription, itemPrice, itemType)

VALUES

-- Food Paradise (Italian)

(4001, 3001, 'Margherita Pizza', 'Classic pizza topped with tomato, mozzarella, and basil', 350.00, 'Main Course'),

(4002, 3001, 'Pasta Alfredo', 'Creamy white sauce pasta with parmesan cheese', 300.00, 'Main Course'),

(4003, 3001, 'Bruschetta', 'Toasted bread topped with tomato, garlic, and basil', 150.00, 'Appetizer'),

(4004, 3001, 'Tiramisu', 'Classic Italian dessert made with mascarpone and coffee', 250.00, 'Dessert'),

(4005, 3001, 'Minestrone Soup', 'Healthy Italian vegetable soup with pasta', 180.00, 'Starter'),



-- Spice Garden (Indian)

(4006, 3002, 'Chicken Tikka Masala', 'Grilled chicken in rich tomato-based gravy', 280.00, 'Main Course'),

(4007, 3002, 'Naan Bread', 'Soft Indian flatbread cooked in a tandoor', 40.00, 'Bread'),

(4008, 3002, 'Palak Paneer', 'Cottage cheese in creamy spinach curry', 240.00, 'Main Course'),

(4009, 3002, 'Samosa', 'Fried pastry filled with spiced potatoes and peas', 50.00, 'Appetizer'),

(4010, 3002, 'Rasgulla', 'Spongy cottage cheese balls soaked in syrup', 80.00, 'Dessert'),
```

-- Burger Hub (American)

(4011, 3003, 'Cheeseburger', 'Beef patty with melted cheese, lettuce, and tomato', 200.00, 'Main Course'),

(4012, 3003, 'French Fries', 'Crispy potato fries served with ketchup', 100.00, 'Side'),

(4013, 3003, 'BBQ Chicken Burger', 'Burger with tangy BBQ chicken and lettuce', 220.00, 'Main Course'),

(4014, 3003, 'Onion Rings', 'Crispy fried onion rings', 120.00, 'Appetizer'),

(4015, 3003, 'Chocolate Milkshake', 'Thick milkshake with chocolate syrup and cream', 150.00, 'Beverage'),

-- Pizza Planet (Italian)

(4016, 3004, 'Pepperoni Pizza', 'Pizza with tomato sauce, mozzarella, and pepperoni', 380.00, 'Main Course'),

(4017, 3004, 'Four Cheese Pizza', 'Pizza topped with four types of cheese', 400.00, 'Main Course'),

(4018, 3004, 'Garlic Bread', 'Toasted bread with garlic butter and herbs', 150.00, 'Appetizer'),

(4019, 3004, 'Panna Cotta', 'Creamy Italian dessert served with berry sauce', 220.00, 'Dessert'),

(4020, 3004, 'Caprese Salad', 'Tomato, mozzarella, and basil drizzled with olive oil', 180.00, 'Starter'),

-- Taco Town (Mexican)

(4021, 3005, 'Beef Tacos', 'Soft tacos filled with spiced beef and vegetables', 200.00, 'Main Course'),

(4022, 3005, 'Nachos', 'Crispy tortilla chips topped with cheese and jalapenos', 180.00, 'Appetizer'),

(4023, 3005, 'Chicken Enchiladas', 'Tortillas filled with chicken and baked with sauce', 250.00, 'Main Course'),

(4024, 3005, 'Churros', 'Fried dough pastry coated in cinnamon sugar', 120.00, 'Dessert'),

(4025, 3005, 'Guacamole', 'Creamy avocado dip served with tortilla chips', 150.00, 'Starter'),

-- Noodle House (Chinese)

(4026, 3006, 'Hakka Noodles', 'Stir-fried noodles with vegetables and soy sauce', 200.00, 'Main Course'),

(4027, 3006, 'Spring Rolls', 'Crispy rolls filled with spiced vegetables', 150.00, 'Appetizer'),

(4028, 3006, 'Kung Pao Chicken', 'Spicy stir-fried chicken with peanuts', 280.00, 'Main Course'),

(4029, 3006, 'Manchow Soup', 'Spicy Chinese soup with fried noodles', 120.00, 'Starter'),

(4030, 3006, 'Fried Rice', 'Rice stir-fried with vegetables and soy sauce', 180.00, 'Main Course'),

-- Sushi World (Japanese)

(4031, 3007, 'California Roll', 'Sushi roll with crab, avocado, and cucumber', 300.00, 'Main Course'),

(4032, 3007, 'Miso Soup', 'Traditional Japanese soup with tofu and seaweed', 150.00, 'Starter'),

(4033, 3007, 'Teriyaki Chicken', 'Grilled chicken with sweet soy-based sauce', 320.00, 'Main Course'),

(4034, 3007, 'Tempura', 'Deep-fried shrimp and vegetables', 250.00, 'Appetizer'),

(4035, 3007, 'Green Tea Ice Cream', 'Ice cream flavored with matcha', 150.00, 'Dessert'),

-- Curry Corner (Indian)

(4036, 3008, 'Butter Chicken', 'Tender chicken cooked in creamy tomato gravy', 300.00, 'Main Course'),

(4037, 3008, 'Dal Tadka', 'Lentils cooked with aromatic spices and butter', 180.00, 'Main Course'),

(4038, 3008, 'Tandoori Roti', 'Whole wheat flatbread cooked in a tandoor', 30.00, 'Bread'),

(4039, 3008, 'Gulab Jamun', 'Sweet dumplings soaked in sugar syrup', 90.00, 'Dessert'),

(4040, 3008, 'Chicken Biryani', 'Aromatic basmati rice cooked with spiced chicken', 250.00, 'Main Course'),

-- Grill Station (Barbecue)

(4041, 3009, 'BBQ Ribs', 'Slow-cooked ribs glazed with BBQ sauce', 350.00, 'Main Course'),

(4042, 3009, 'Grilled Chicken Wings', 'Spicy and smoky grilled chicken wings', 200.00, 'Appetizer'),

(4043, 3009, 'Smoked Brisket', 'Juicy beef brisket smoked to perfection', 400.00, 'Main Course'),

(4044, 3009, 'Cornbread', 'Classic BBQ side with a hint of sweetness', 80.00, 'Side'),

(4045, 3009, 'Coleslaw', 'Crunchy cabbage salad with creamy dressing', 100.00, 'Side'),

-- Vegan Delight (Vegan)

(4046, 3010, 'Quinoa Salad', 'Healthy salad with quinoa, veggies, and vinaigrette', 200.00, 'Main Course'),

(4047, 3010, 'Vegan Burger', 'Plant-based patty with lettuce and tomato', 250.00, 'Main Course'),

(4048, 3010, 'Vegan Brownie', 'Rich chocolate brownie made without dairy', 150.00, 'Dessert'),

(4049, 3010, 'Lentil Soup', 'Hearty soup made with lentils and spices', 120.00, 'Starter'),

(4050, 3010, 'Stuffed Bell Peppers', 'Bell peppers filled with quinoa and veggies', 220.00, 'Main Course'),

-- Pasta Palace (Italian)

(4051, 3011, 'Fettuccine Carbonara', 'Creamy pasta with bacon and parmesan', 320.00, 'Main Course'),

(4052, 3011, 'Garlic Knots', 'Soft bread knots brushed with garlic butter', 140.00, 'Appetizer'),

(4053, 3011, 'Lasagna', 'Layered pasta with meat, cheese, and tomato sauce', 400.00, 'Main Course'),

(4054, 3011, 'Cannoli', 'Sweet pastry filled with ricotta cheese', 180.00, 'Dessert'),

(4055, 3011, 'Caesar Salad', 'Crispy romaine lettuce with Caesar dressing', 160.00, 'Starter'),

-- The Salad Bar (Healthy)

(4056, 3012, 'Greek Salad', 'Salad with olives, feta, and fresh veggies', 180.00, 'Main Course'),

(4057, 3012, 'Kale Smoothie', 'Healthy smoothie made with kale and fruits', 150.00, 'Beverage'),

(4058, 3012, 'Avocado Toast', 'Whole-grain toast topped with mashed avocado', 200.00, 'Appetizer'),

(4059, 3012, 'Fruit Salad', 'Seasonal fruits tossed in honey and lime', 180.00, 'Dessert'),

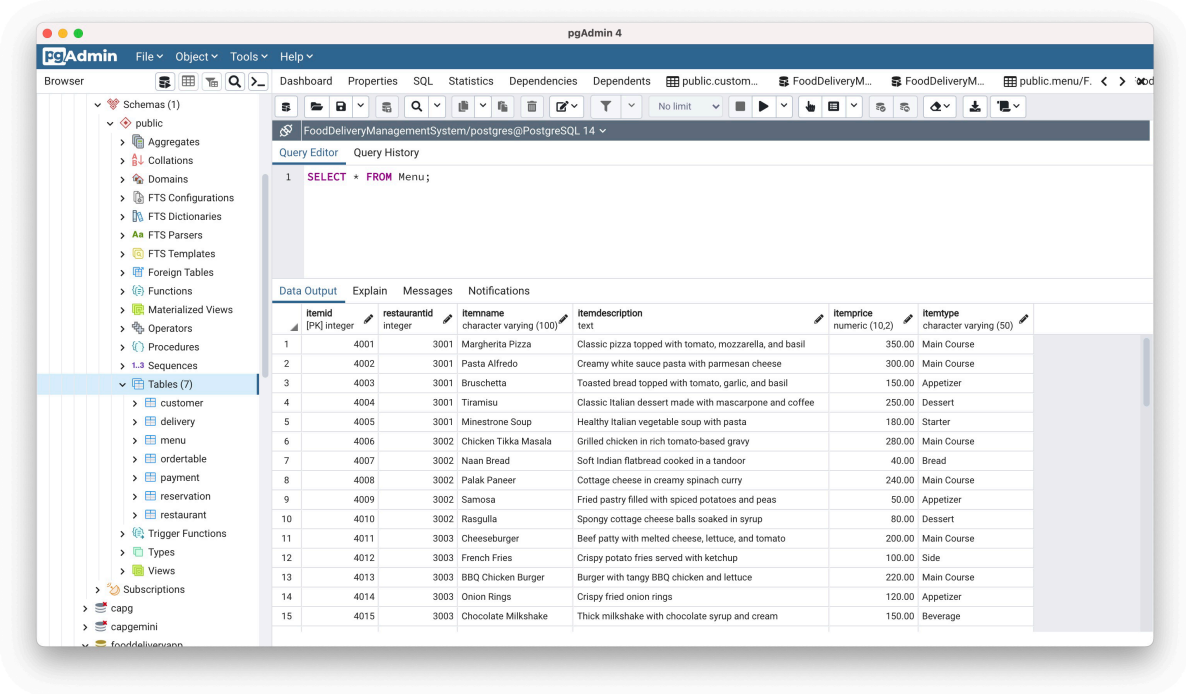
(4060, 3012, 'Lentil Salad', 'Protein-packed salad with lentils and veggies', 190.00, 'Main Course');

Result:

Table Explanation: - This table contains details of restaurant menus, including dish names, descriptions, prices, and types.

Query: SELECT * FROM Menu;

Screenshot:



Entity Generation and Data Entry for Table Order:

Statements Explanation:

- The database FoodDeliveryManagementSystem is already created. Command Use FoodDeliveryManagementSystem is used to call the database.
- Then, the table Order is created in the FoodDeliveryManagementSystem database. Command CREATE TABLE ORDER is used to create the table.
- Next, table Order is filled with the relevant data using the command INSERT INTO.
- Finally, the result will be displayed by using the command SELECT * FROM Order to query all the inserted values into the table.

Query:

```
INSERT INTO OrderTable (orderId, customerId, restaurantId, orderDate)

VALUES

(2001, 1001, 3001, '2024-10-28'),

(2002, 1002, 3002, '2024-10-29'),

(2003, 1003, 3003, '2024-10-30'),

(2004, 1004, 3004, '2024-10-31'),

(2005, 1005, 3005, '2024-11-01'),

(2006, 1006, 3006, '2024-11-02'),

(2007, 1007, 3007, '2024-11-03'),

(2008, 1008, 3008, '2024-11-04'),

(2009, 1009, 3009, '2024-11-05'),

(2010, 1010, 3010, '2024-11-06'),

(2011, 1011, 3011, '2024-11-07'),

(2012, 1012, 3012, '2024-11-08'),

(2013, 1013, 3013, '2024-11-09'),

(2014, 1014, 3014, '2024-11-10'),

(2015, 1015, 3015, '2024-11-11'),

(2016, 1016, 3016, '2024-11-12'),
```

(2017, 1017, 3017, '2024-11-13'),

(2018, 1018, 3018, '2024-11-14'),

(2019, 1019, 3019, '2024-11-15'),

(2020, 1020, 3020, '2024-11-16'),

(2021, 1021, 3021, '2024-11-17'),

(2022, 1022, 3022, '2024-11-18'),

(2023, 1023, 3023, '2024-11-19'),

(2024, 1024, 3024, '2024-11-20'),

(2025, 1025, 3025, '2024-11-21'),

(2026, 1026, 3026, '2024-11-22'),

(2027, 1027, 3027, '2024-11-23'),

(2028, 1028, 3028, '2024-11-24'),

(2029, 1029, 3029, '2024-11-25'),

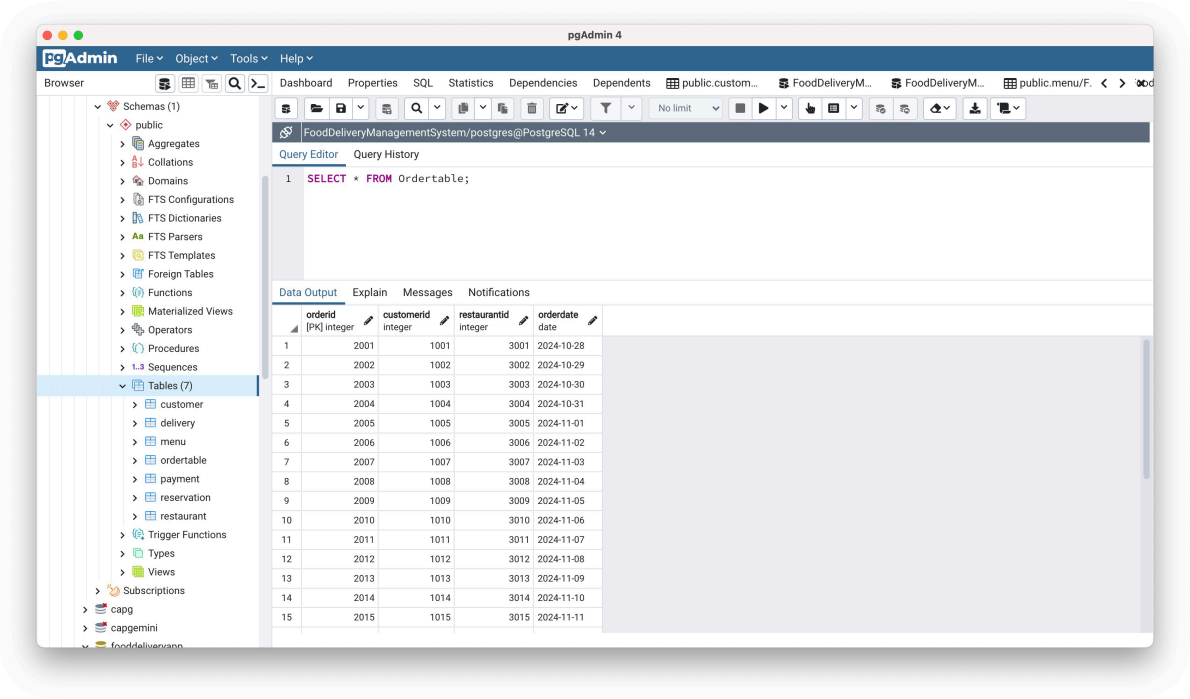
(2030, 1030, 3030, '2024-11-26');

Result:

Table Explanation: - This table tracks customer orders with details about items ordered, quantities, and prices.

Query: SELECT * FROM Ordertable;

Screenshot:



Entity Generation and Data Entry for Table Reservation:

Statements Explanation:

- The database FoodDeliveryManagementSystem is already created. Command Use FoodDeliveryManagementSystem is used to call the database.
- Then, the table Order is created in the FoodDeliveryManagementSystem database. Command CREATE TABLE Reservation is used to create the table.
- Next, table Reservation is filled with the relevant data using the command INSERT INTO.
- Finally, the result will be displayed by using the command SELECT * FROM Reservation to query all the inserted values into the table.

Query:

INSERT INTO Reservation (reservationId, customerId, restaurantId, reservationDate, numberOfPeople)

VALUES

(7001, 1001, 3001, '2024-11-05', 4),

(7002, 1002, 3002, '2024-11-06', 2),

(7003, 1003, 3003, '2024-11-07', 6),

(7004, 1004, 3004, '2024-11-08', 3),

(7005, 1005, 3005, '2024-11-09', 5),

(7006, 1006, 3006, '2024-11-10', 2),

(7007, 1007, 3007, '2024-11-11', 7),

(7008, 1008, 3008, '2024-11-12', 4),

(7009, 1009, 3009, '2024-11-13', 3),

(7010, 1010, 3010, '2024-11-14', 6),

(7011, 1011, 3011, '2024-11-15', 2),

(7012, 1012, 3012, '2024-11-16', 5),

(7013, 1013, 3013, '2024-11-17', 4),

(7014, 1014, 3014, '2024-11-18', 8),

(7015, 1015, 3015, '2024-11-19', 3),

(7016, 1016, 3016, '2024-11-20', 7),

(7017, 1017, 3017, '2024-11-21', 2),

(7018, 1018, 3018, '2024-11-22', 5),

(7019, 1019, 3019, '2024-11-23', 4),

(7020, 1020, 3020, '2024-11-24', 6),

(7021, 1021, 3021, '2024-11-25', 3),

(7022, 1022, 3022, '2024-11-26', 4),

(7023, 1023, 3023, '2024-11-27', 8),

(7024, 1024, 3024, '2024-11-28', 2),

(7025, 1025, 3025, '2024-11-29', 5),

(7026, 1026, 3026, '2024-11-30', 3),

(7027, 1027, 3027, '2024-12-01', 7),

(7028, 1028, 3028, '2024-12-02', 6),

(7029, 1029, 3029, '2024-12-03', 2),

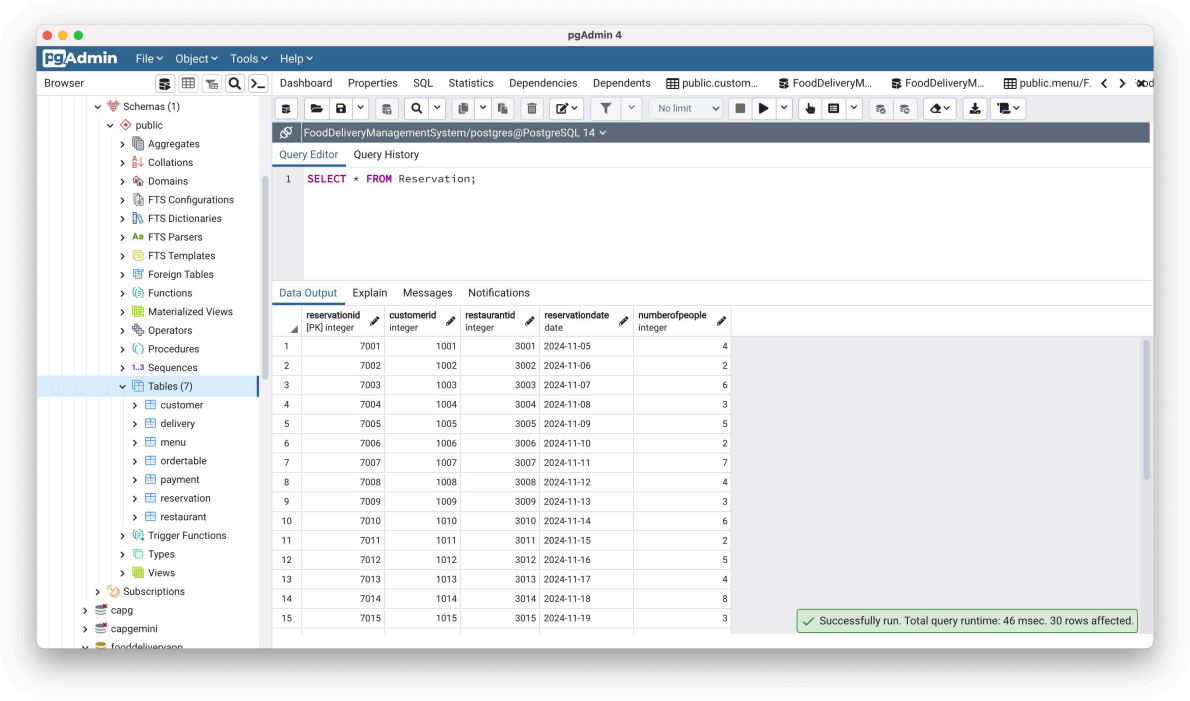
(7030, 1030, 3030, '2024-12-04', 4);

Result:

Table Explanation: - This table contains records of reservations with reservation dates, times, and guest counts.

Query: SELECT * FROM Reservation;

Screenshot:



Entity Generation and Data Entry for Table Payment:

Statements Explanation:

- The database FoodDeliveryManagementSystem is already created. Command Use FoodDeliveryManagementSystem is used to call the database.
- Then, the table Payment is created in the FoodDeliveryManagementSystem database. Command CREATE TABLE PAYMENT is used to create the table.
- Next, table Payment is filled with the relevant data using the command INSERT INTO.
- Finally, the result will be displayed by using the command SELECT * FROM Payment to query all the inserted values into the table.

Query:

INSERT INTO Payment (paymentId, orderId, transactionId, paymentDate, totalAmount, paymentMethod, paymentStatus)

VALUES

(4001, 2001, 'TXN12345', '2024-10-28', 150.00, 'Card', 'Paid'),

(4002, 2002, 'TXN67890', '2024-10-29', 200.00, 'Cash', 'Paid'),

(4003, 2003, 'TXN34567', '2024-10-30', 120.00, 'UPI', 'Paid'),

(4004, 2004, 'TXN45678', '2024-10-31', 180.00, 'Net Banking', 'Paid'),

(4005, 2005, 'TXN56789', '2024-11-01', 250.00, 'Card', 'Paid'),

(4006, 2006, 'TXN67891', '2024-11-02', 175.00, 'Cash', 'Paid'),

(4007, 2007, 'TXN78912', '2024-11-03', 220.00, 'UPI', 'Pending'),

(4008, 2008, 'TXN89123', '2024-11-04', 130.00, 'Card', 'Paid'),

(4009, 2009, 'TXN91234', '2024-11-05', 145.00, 'Net Banking', 'Paid'),

(4010, 2010, 'TXN12346', '2024-11-06', 165.00, 'Card', 'Paid'),

(4011, 2011, 'TXN23457', '2024-11-07', 210.00, 'Cash', 'Paid'),

(4012, 2012, 'TXN34568', '2024-11-08', 140.00, 'UPI', 'Paid'),

(4013, 2013, 'TXN45679', '2024-11-09', 195.00, 'Net Banking', 'Pending'),

(4014, 2014, 'TXN56790', '2024-11-10', 205.00, 'Card', 'Paid'),

(4015, 2015, 'TXN67892', '2024-11-11', 135.00, 'Cash', 'Paid'),

(4016, 2016, 'TXN78913', '2024-11-12', 185.00, 'UPI', 'Paid'),

(4017, 2017, 'TXN89124', '2024-11-13', 160.00, 'Net Banking', 'Paid'),

(4018, 2018, 'TXN91235', '2024-11-14', 175.00, 'Card', 'Paid'),

(4019, 2019, 'TXN12347', '2024-11-15', 190.00, 'Cash', 'Paid'),

(4020, 2020, 'TXN23458', '2024-11-16', 225.00, 'UPI', 'Paid'),

(4021, 2021, 'TXN34569', '2024-11-17', 155.00, 'Card', 'Pending'),

(4022, 2022, 'TXN45680', '2024-11-18', 175.00, 'Net Banking', 'Paid'),

(4023, 2023, 'TXN56791', '2024-11-19', 195.00, 'Cash', 'Paid'),

(4024, 2024, 'TXN67893', '2024-11-20', 180.00, 'UPI', 'Paid'),

(4025, 2025, 'TXN78914', '2024-11-21', 165.00, 'Card', 'Paid'),

(4026, 2026, 'TXN89125', '2024-11-22', 140.00, 'Cash', 'Paid'),

(4027, 2027, 'TXN91236', '2024-11-23', 200.00, 'Net Banking', 'Paid'),

(4028, 2028, 'TXN12348', '2024-11-24', 230.00, 'Card', 'Pending'),

(4029, 2029, 'TXN23459', '2024-11-25', 215.00, 'UPI', 'Paid'),

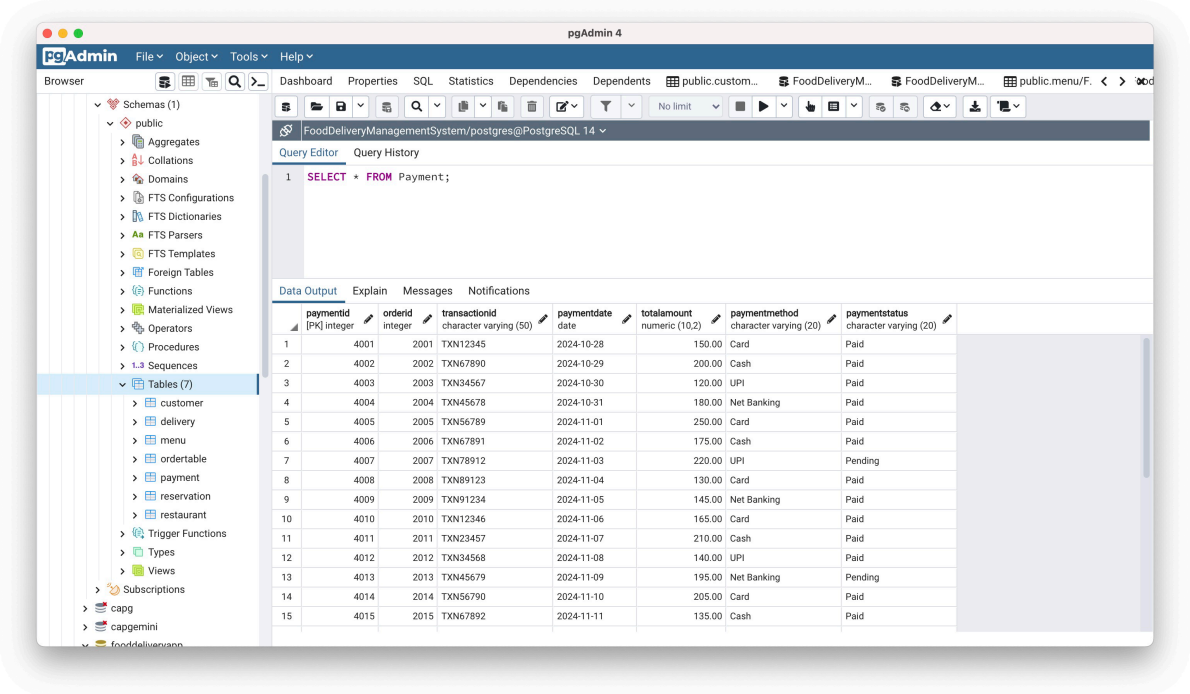
(4030, 2030, 'TXN34570', '2024-11-26', 145.00, 'Cash', 'Paid');

Result:

Table Explanation: - This table contain payment transaction details, including method, amount, and payment status.

Query: SELECT * FROM Payment;

Screenshot:



Entity Generation and Data Entry for Table Delivery:

Statements Explanation:

- The database FoodDeliveryManagementSystem is already created. Command Use FoodDeliveryManagementSystem is used to call the database.

- Then, the table Delivery is created in the FoodDeliveryManagementSystem database. Command CREATE TABLE DELIVERY is used to create the table.
- Next, table Delivery is filled with the relevant data using the command INSERT INTO.
- Finally, the result will be displayed by using the command SELECT * FROM Delivery to query all the inserted values into the table.

Query:

```
INSERT INTO Delivery (deliveryId, orderId, deliveryStatus, deliveryDate)

VALUES

(5001, 2001, 'Delivered', '2024-10-29'),

(5002, 2002, 'Pending', '2024-10-30'),

(5003, 2003, 'Delivered', '2024-10-31'),

(5004, 2004, 'Pending', '2024-11-01'),

(5005, 2005, 'Delivered', '2024-11-02'),

(5006, 2006, 'Pending', '2024-11-03'),

(5007, 2007, 'Delivered', '2024-11-04'),

(5008, 2008, 'Pending', '2024-11-05'),

(5009, 2009, 'Delivered', '2024-11-06'),

(5010, 2010, 'Pending', '2024-11-07'),

(5011, 2011, 'Delivered', '2024-11-08'),

(5012, 2012, 'Pending', '2024-11-09'),

(5013, 2013, 'Delivered', '2024-11-10'),

(5014, 2014, 'Pending', '2024-11-11'),

(5015, 2015, 'Delivered', '2024-11-12'),

(5016, 2016, 'Pending', '2024-11-13'),

(5017, 2017, 'Delivered', '2024-11-14'),

(5018, 2018, 'Pending', '2024-11-15'),

(5019, 2019, 'Delivered', '2024-11-16'),
```

(5020, 2020, 'Pending', '2024-11-17'),

(5021, 2021, 'Delivered', '2024-11-18'),

(5022, 2022, 'Pending', '2024-11-19'),

(5023, 2023, 'Delivered', '2024-11-20'),

(5024, 2024, 'Pending', '2024-11-21'),

(5025, 2025, 'Delivered', '2024-11-22'),

(5026, 2026, 'Pending', '2024-11-23'),

(5027, 2027, 'Delivered', '2024-11-24'),

(5028, 2028, 'Pending', '2024-11-25'),

(5029, 2029, 'Delivered', '2024-11-26'),

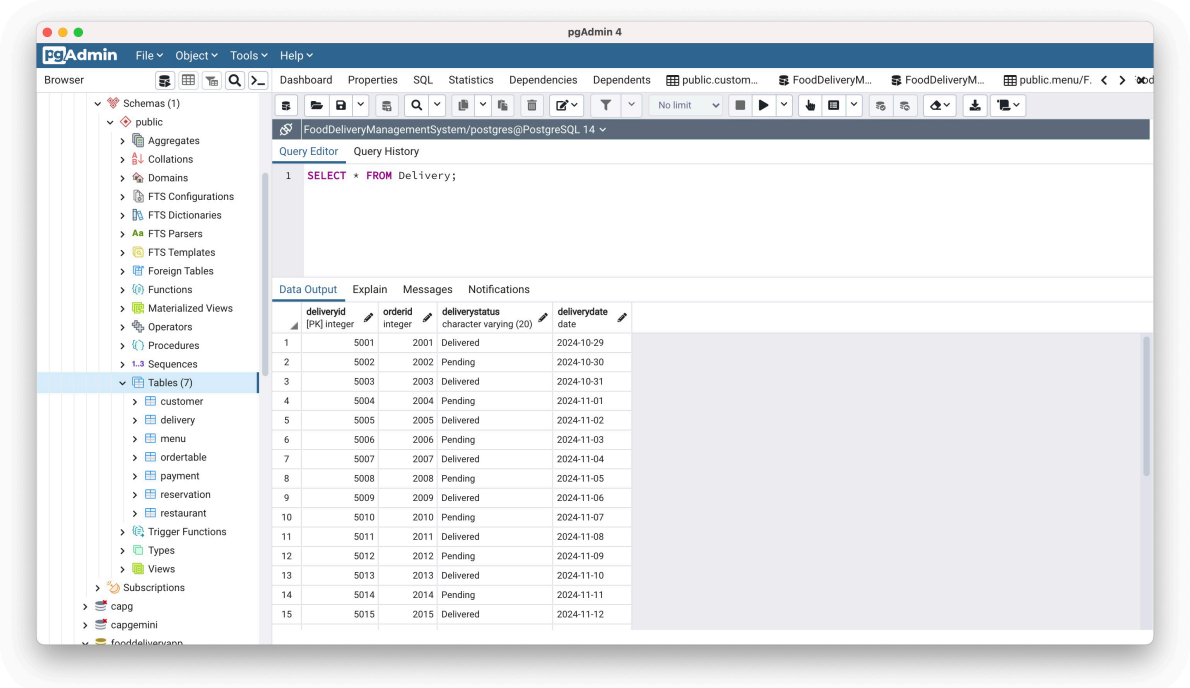
(5030, 2030, 'Pending', '2024-11-27');

Result:

1. **Table Explanation:** - This table contains delivery information, such as order delivery status, delivery dates, and payment type.

   Query: SELECT * FROM Delivery;

   Screenshot:



Section 2: Data Retrieval and Analytical Reports

Overview:

Extracting meaningful information from raw data can be beneficial for almost any company in the current data-driven decision-making trends. The queries in this section are designed to assist higher management in making data-informed decisions that result in lower costs of operation, higher profits and maintaining high customer satisfaction. First, for each of the data analyses queries the statement will be explained, then the key SQL command will be described, and at the end, the SQL SELECT statement will be written with the result of the query outcome shown in a screenshot.

***Comment:** Before executing queries in this section (Section 2), queries in Section 1 should be run in the provided order, to create DB, create tables, and insert values into the tables. Appendix 1 contains all combined SQL code for section 1 to provide easy copy and paste into MySQL.

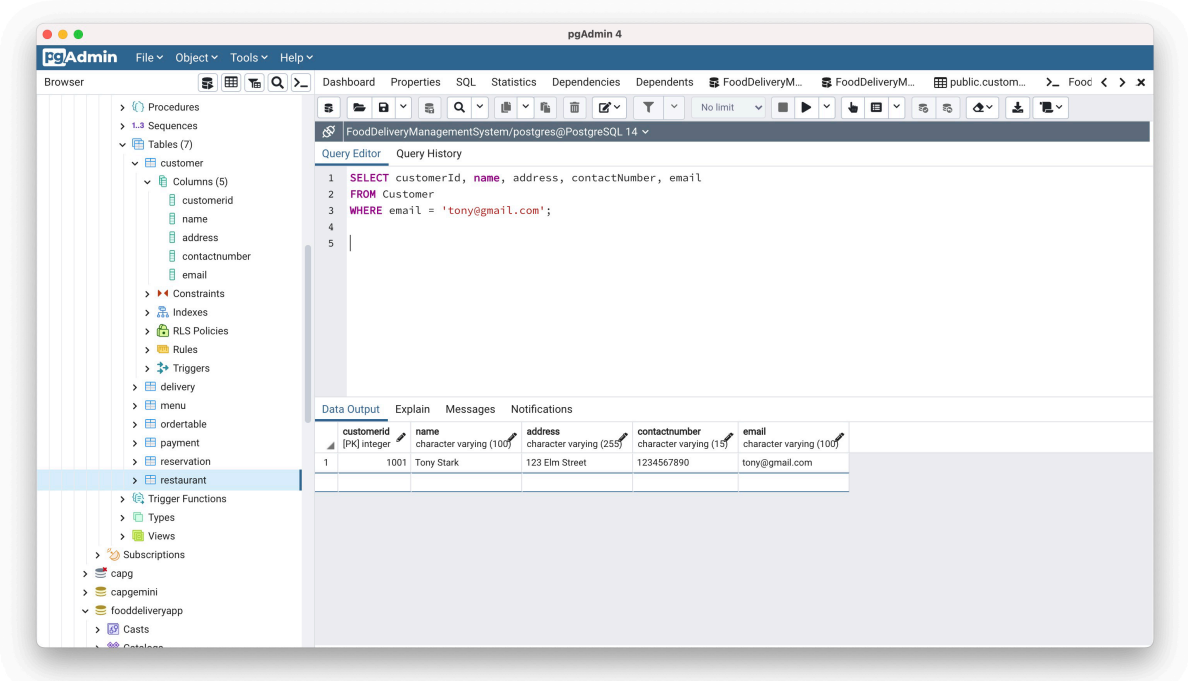
Data Analysis 1

Statement: Retrieve customer details based on email Id.

SQL command:

```
SELECT customerId, name, address, contactNumber, email

FROM Customer

WHERE email = 'tony@example.com';
```

Result:



SQL command explanation:

- Command **FROM** is used to retrieve the data from the Customer table in the database.
- Command **WHERE** is used to filter the data to match the criteria where the email column is equal to 'tony@example.com'. This ensures only data associated with the specified email is retrieved.
- Command **SELECT** is used to Specify the columns (customerId, name, address, contactNumber, and email) that should be shown in the result set.

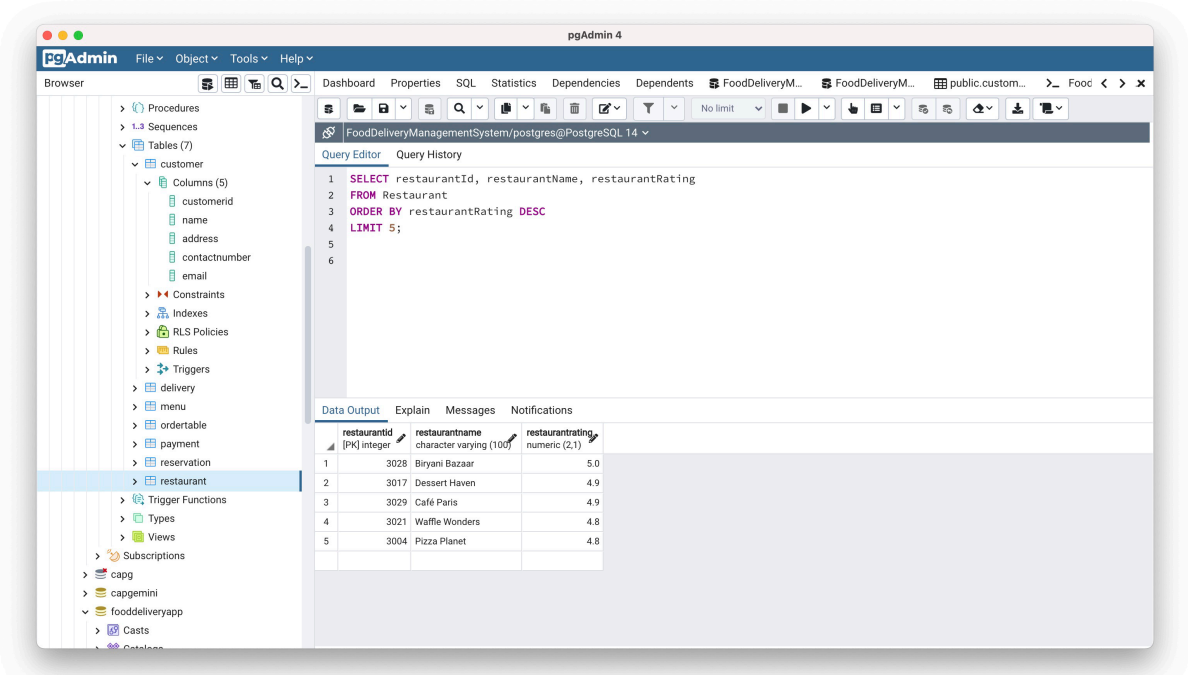
Data Analysis 2

Statement: Retrieve top 5 restaurants by rating in descending order of rating.

SQL command:

```
SELECT restaurantId, restaurantName, restaurantRating

FROM Restaurant

ORDER BY restaurantRating DESC

LIMIT 5;
```

Result:



SQL command explanation:

- Command **FROM** is used to retrieve the data from the Restaurant table in the database.
- Command **SELECT** is used to specify the columns (restaurantId, restaurantName, and restaurantRating) to be shown in the result set.
- Command **ORDER BY** is used to sort the results in descending order (DESC) based on the restaurantRating column. This ensures the restaurants with the highest ratings appear first.
- Command **LIMIT** is used to restrict the number of rows returned to the top 5 results.

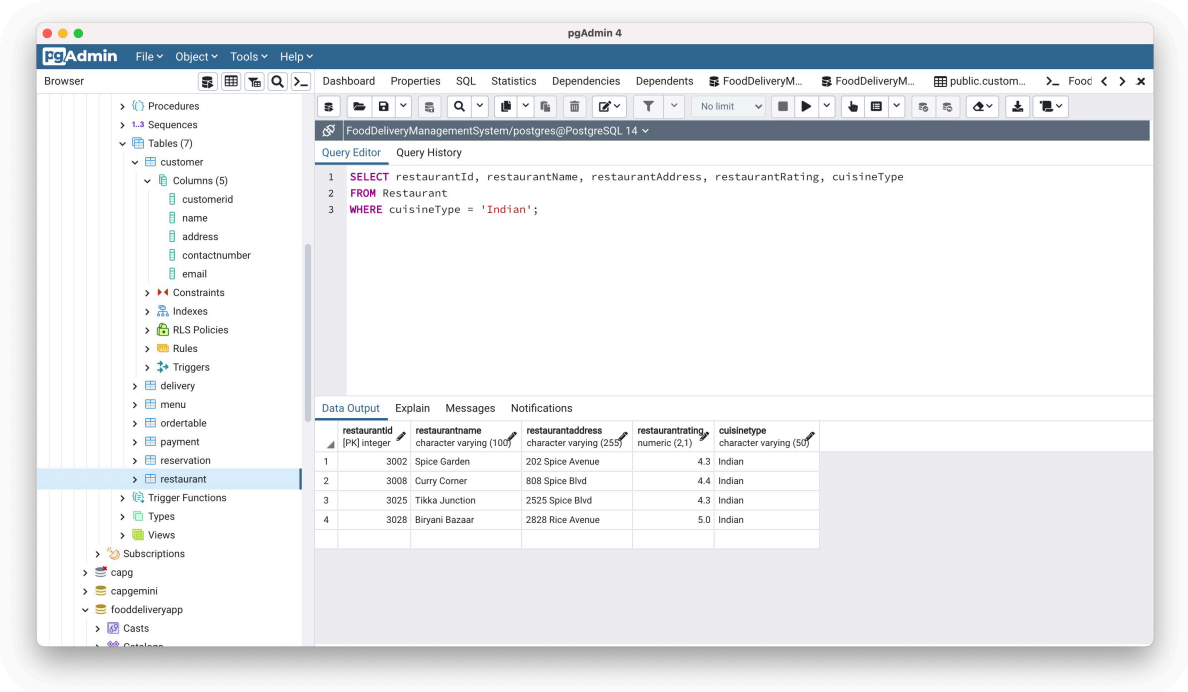
Data Analysis 3

Statement: Retrieve restaurants with Indian cuisine.

SQL command:

```
SELECT restaurantId, restaurantName, restaurantAddress, restaurantRating, cuisineType

FROM Restaurant

WHERE cuisineType = 'Indian';
```

Result:



SQL command explanation:

- Command **FROM**: Retrieves the data from the Restaurant table in the database.
- Command **WHERE**: Filters the data to match the criteria where the cuisineType column is equal to 'Indian'.
- Command **SELECT**: Specifies the columns (restaurantId, restaurantName, restaurantAddress, restaurantRating, and cuisineType) to be included in the result.

Data Analysis 4

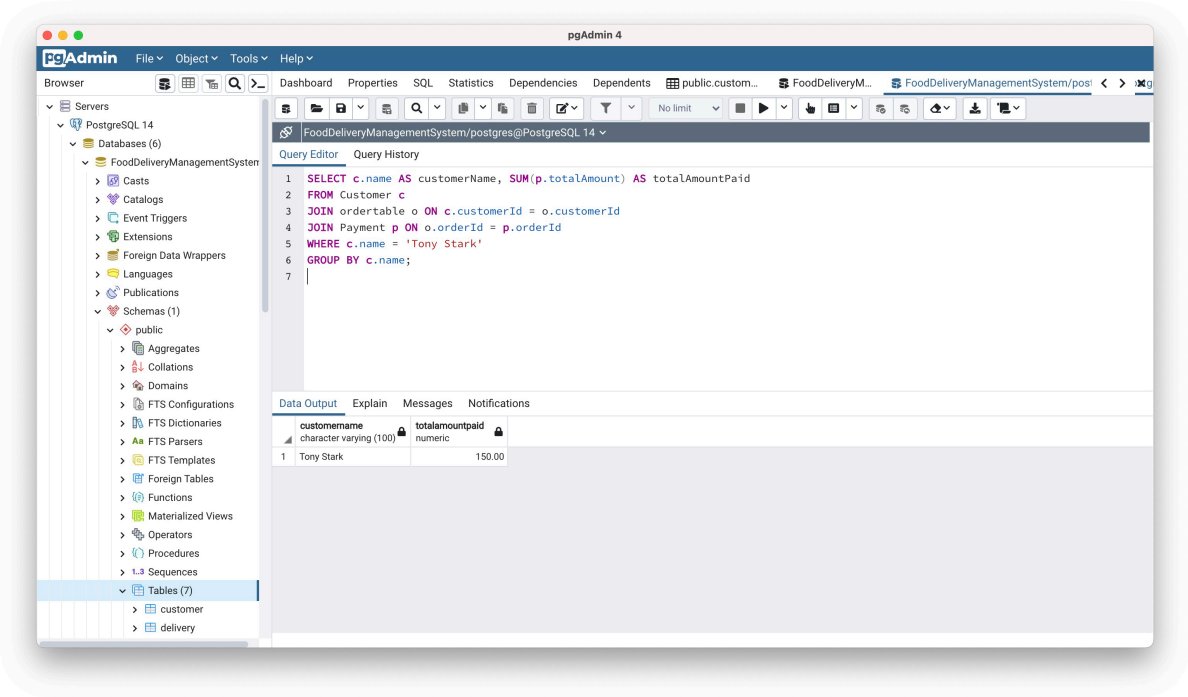
Statement: Retrieve the total amount of sale from a single customer.

SQL command:

```
SELECT c.name AS customerName, SUM(p.totalAmount) AS totalAmountPaid

FROM Customer c

JOIN ordertable o ON c.customerId = o.customerId

JOIN Payment p ON o.orderId = p.orderId

WHERE c.name = 'Tony Stark'

GROUP BY c.name;
```

Explanation:

1. **Customer c:** This is the alias for the Customer table.
2. **ordertable o:** This is the alias for the ordertable table, representing the orders placed by the customer.
3. **Payment p:** This is the alias for the Payment table, which links the payments made for specific orders.
4. **Condition WHERE c.name = 'Tony Stark':** Filters for the customer whose name is Tony Stark.
5. **SUM(p.totalAmount):** Calculates the total amount paid by the customer for all orders.
6. **GROUP BY c.name:** Ensures results are grouped by customer name.



SQL command explanation:

- Command FROM is used to retrieve the data from the Customer, ordertable, and Payment tables in the database.
- Command JOIN is used to join the Customer table (c) with the ordertable table (o)
- Command SELECT is used to specify the columns to be displayed in the result set:
- Command WHERE is used to filter the data to include only rows where the customer's name (c.name) is equal to 'Tony Stark'. This ensures the result focuses only on this specific customer.
- Command GROUP BY is used to group the data by the customer's name (c.name).

Data Analysis 5

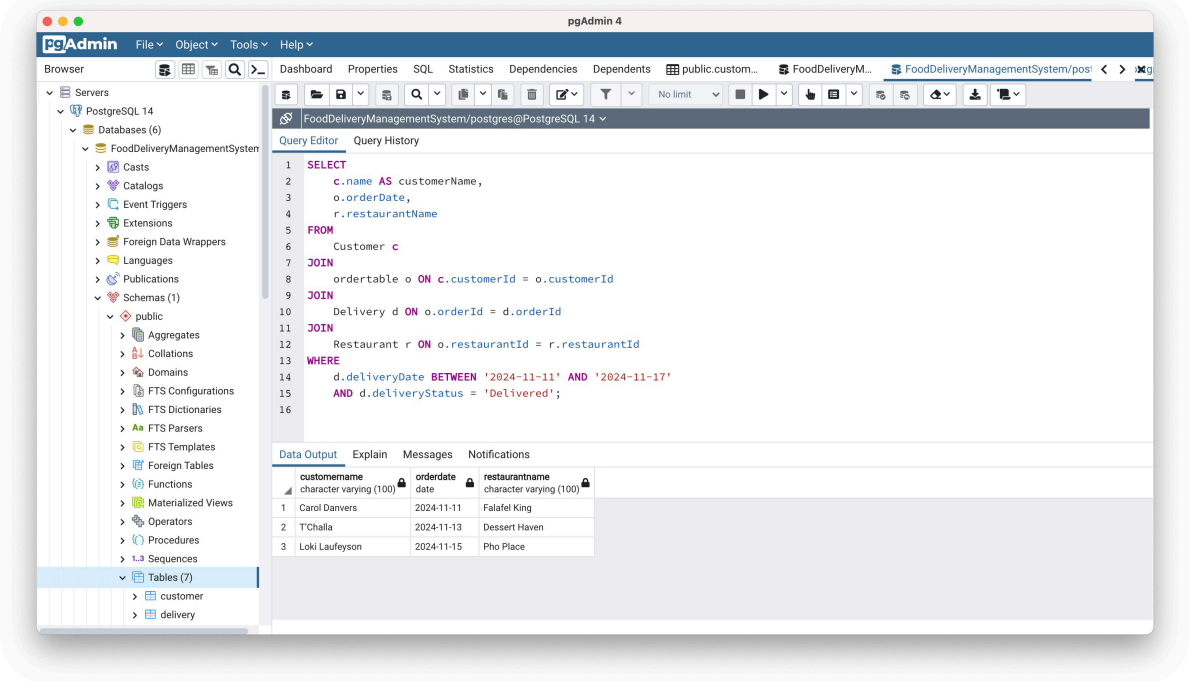
Statement: Retrieve the customer name, order date, and restaurant name for all orders delivered in the second week of November

SQL command:

```
SELECT

    c.name AS customerName,

    o.orderDate,

    r.restaurantName

FROM
```

```
    Customer c
JOIN
    ordertable o ON c.customerId = o.customerId
JOIN
    Delivery d ON o.orderId = d.orderId
JOIN
    Restaurant r ON o.restaurantId = r.restaurantId
WHERE
    d.deliveryDate BETWEEN '2024-11-11' AND '2024-11-17'
    AND d.deliveryStatus = 'Delivered';
```

Result:



SQL command explanation:

- Command **FROM** is used to retrieve the data from the Customer, ordertable, Delivery, and Restaurant tables in the database.
- Command **JOIN** is used to join the Customer table (c) with the ordertable table (o), the ordertable table (o) with the Delivery table (d) and the ordertable table (o) with the Restaurant table (r).

- Command **SELECT** to specifies the columns.
- Command **WHERE** is used to filter the data based on specific conditions

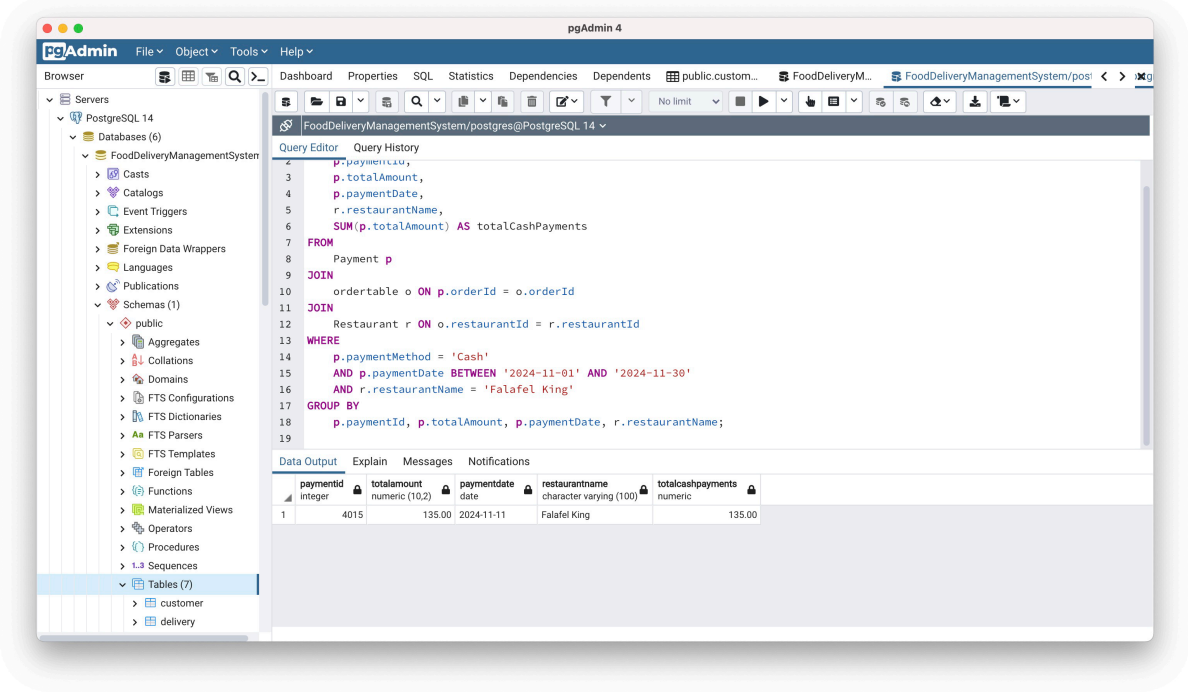
Data Analysis 6

Statement: Retrieve all cash type payments and their total sum for a specific restaurant for the month of November.

SQL command:

```
SELECT
    p.paymentId,
    p.totalAmount,
    p.paymentDate,
    r.restaurantName,
    SUM(p.totalAmount) AS totalCashPayments
FROM
    Payment p
JOIN
    ordertable o ON p.orderId = o.orderId
JOIN
    Restaurant r ON o.restaurantId = r.restaurantId
WHERE
    p.paymentMethod = 'Cash'
    AND p.paymentDate BETWEEN '2024-11-01' AND '2024-11-30'
    AND r.restaurantName = 'Falafel King'
GROUP BY
    p.paymentId, p.totalAmount, p.paymentDate, r.restaurantName;
```

Result:



SQL command explanation:

- Command **FROM** is used to retrieve the data from the Payment, ordertable, and Restaurant tables in the database.
- Command **JOIN** is used to join the Tables.
- Command **SELECT** is used to specify the columns to display in the result set.
- Command **WHERE** is used to filter the data based on specific conditions.
- Command **GROUP BY** is used to group the results by the specified columns

Data Analysis 7

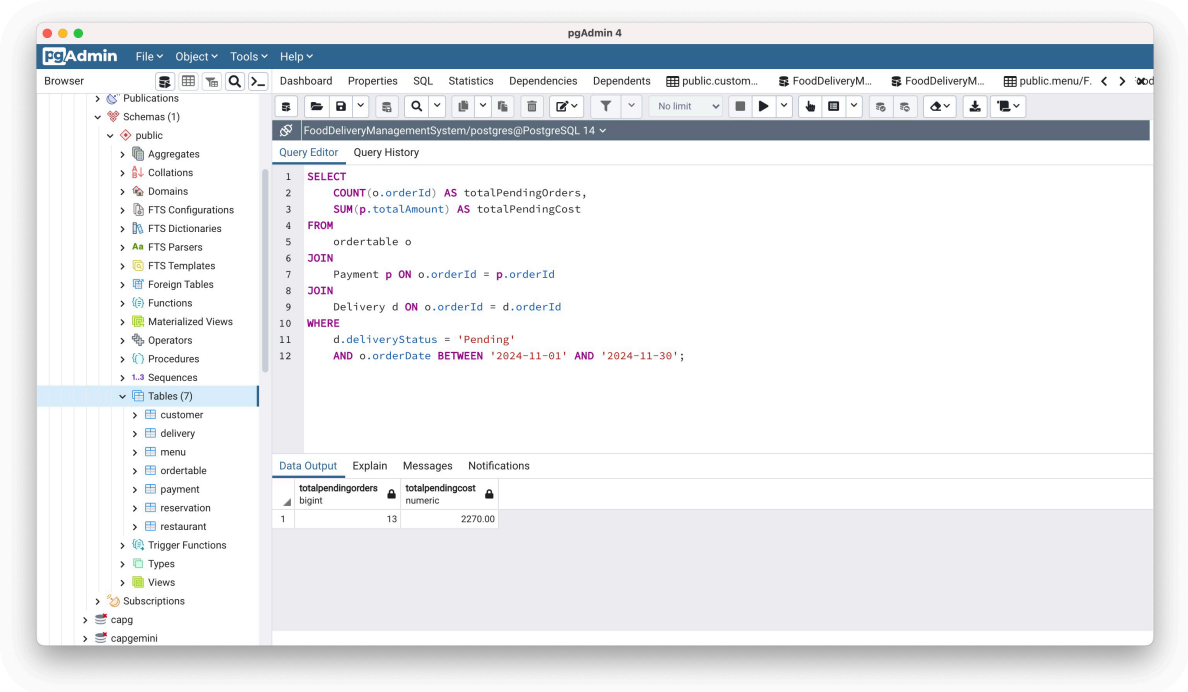
Statement: Retrieve number of pending deliveries and its total cost

SQL command:

```
SELECT
    COUNT(o.orderId) AS totalPendingOrders,
    SUM(p.totalAmount) AS totalPendingCost
FROM
    ordertable o
JOIN
```

```
    Payment p ON o.orderId = p.orderId
JOIN
    Delivery d ON o.orderId = d.orderId
WHERE
    d.deliveryStatus = 'Pending'
    AND o.orderDate BETWEEN '2024-11-01' AND '2024-11-30';
```

Result:



SQL command explanation:

- Command **FROM** is used to retrieve the data from the Payment, ordertable, and Restaurant tables in the database.
- Command **JOIN** is used to join the Payment table (p) with the ordertable table (o) and the ordertable table (o) with the Restaurant table (r). This associates the orders with the corresponding restaurants.
- Command **SELECT** is used to specify the columns to display in the result set.
- Command **WHERE** is used to filter the data based on specific conditions.
- Command **GROUP BY** is used to group the results by the specified columns.

Data Analysis 8

Statement: Retrieve customer name, contact details with number of times ordered and its total cost who ordered more than the average cost of all the orders.

SQL command:

```
WITH AverageCost AS (
    SELECT AVG(p.totalAmount) AS avgOrderCost
    FROM Payment p
),
CustomerOrders AS (
    SELECT
        c.name AS customerName,
        c.contactNumber AS contact,
        c.email AS emailId,
        COUNT(o.orderId) AS numberOfOrders,
        SUM(p.totalAmount) AS totalOrderCost
    FROM
        Customer c
    JOIN
        ordertable o ON c.customerId = o.customerId
    JOIN
        Payment p ON o.orderId = p.orderId
    GROUP BY
        c.name, c.contactNumber, c.email
)
SELECT
    customerName,
```

contact,

emailId,

numberOfOrders,

totalOrderCost

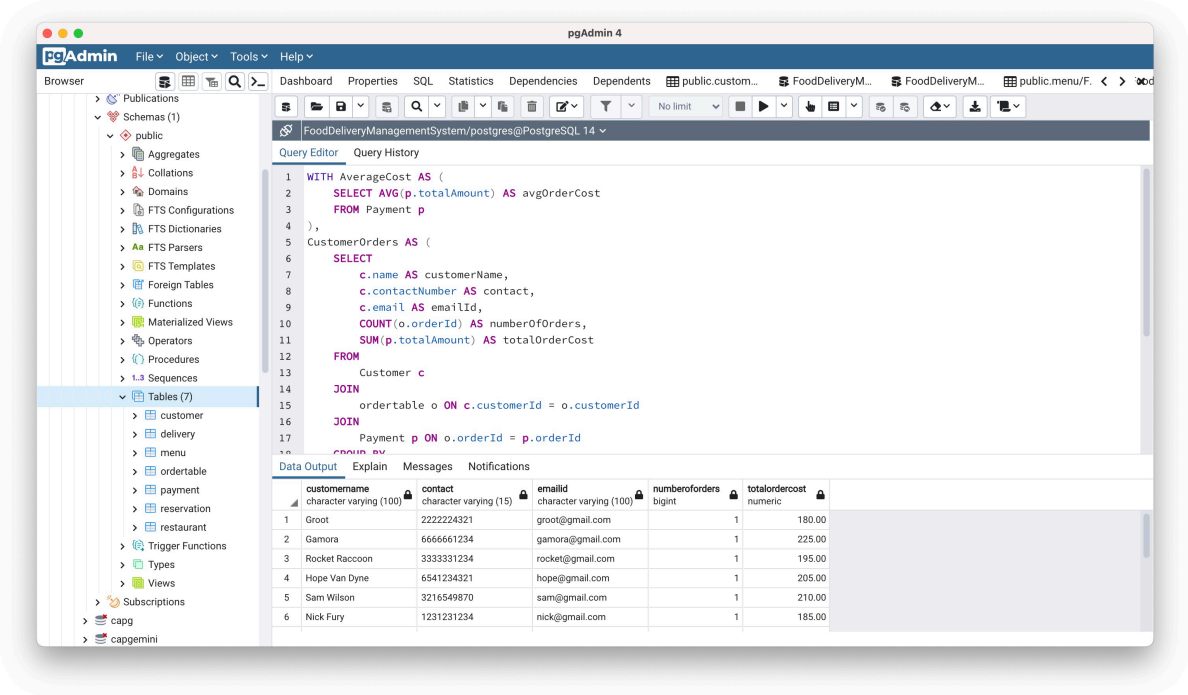
FROM

CustomerOrders, AverageCost

WHERE

totalOrderCost > avgOrderCost;

Result:



SQL command explanation:

- Command **WITH** is used to define two Common Table Expressions (CTEs).
- Command **SELECT** is used to specify the columns to display in the final result set.
- Command **FROM** is used to retrieve the data from the CustomerOrders CTE and the AverageCost CTE.
- Command **WHERE** is used to filter the data to include only those customers whose total order cost is greater than the average total order cost, which was calculated in the AverageCost CTE.

- Command **JOIN** is not explicitly used in the main SELECT query, but the CustomerOrders CTE is implicitly joined with the AverageCost CTE by referencing avgOrderCost in the WHERE clause.
- Command **GROUP BY** is used to group the data by customer attributes: c.name, c.contactNumber, and c.email, ensuring that the number of orders and total order cost are calculated per customer.

Data Analysis 9

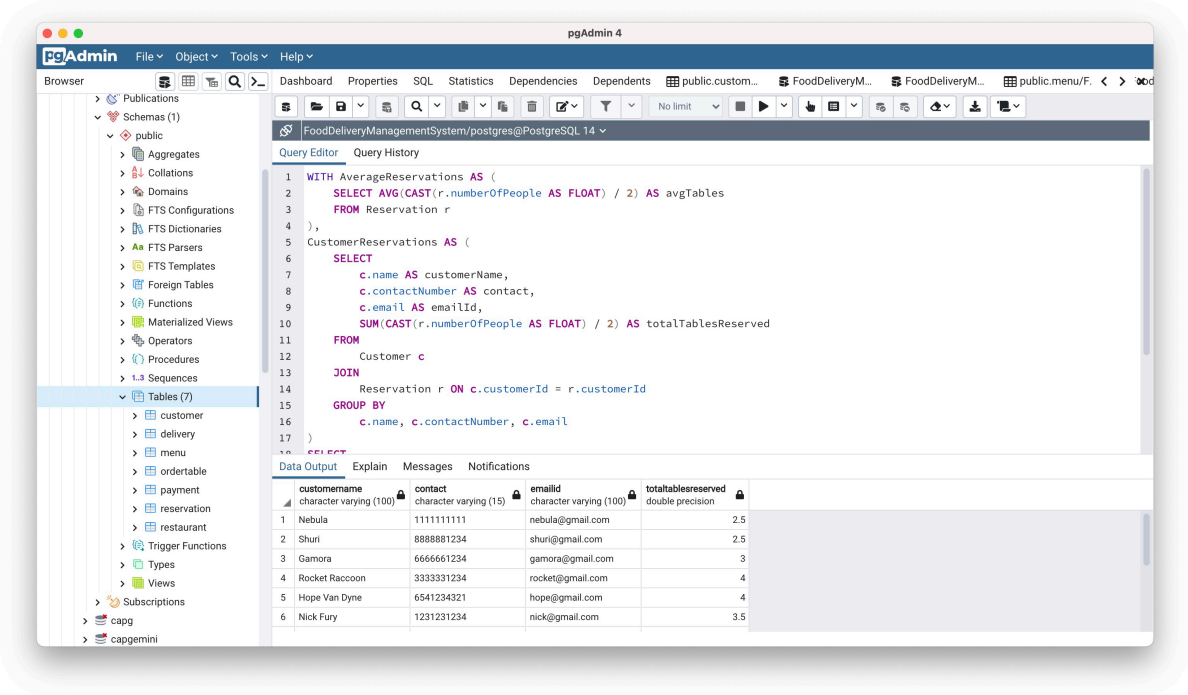
Statement: Retrieve customer details who reserved more than average number of tables with number of tables reserved.

SQL command:

```
WITH AverageReservations AS (
    SELECT AVG(CAST(r.numberOfPeople AS FLOAT) / 2) AS avgTables
    FROM Reservation r
),
CustomerReservations AS (
    SELECT
        c.name AS customerName,
        c.contactNumber AS contact,
        c.email AS emailId,
        SUM(CAST(r.numberOfPeople AS FLOAT) / 2) AS totalTablesReserved
    FROM
        Customer c
    JOIN
        Reservation r ON c.customerId = r.customerId
    GROUP BY
        c.name, c.contactNumber, c.email
)
SELECT
```

```
    customerName,
    contact,
    emailId,
    totalTablesReserved
FROM
    CustomerReservations, AverageReservations
WHERE
    totalTablesReserved > avgTables;
```

Result:



customername	contact	emailid	totaltablesreserved
Nebula	1111111111	nebula@gmail.com	2.5
Shuri	8888881234	shuri@gmail.com	2.5
Gamora	6666661234	gamora@gmail.com	3
Rocket Raccoon	3333331234	rocket@gmail.com	4
Hope Van Dyne	6541234321	hope@gmail.com	4
Nick Fury	1231231234	nick@gmail.com	3.5

SQL command explanation:

- Command **WITH** is used to define two Common Table Expressions (CTEs).
- Command **SELECT** is used to specify the columns to display in the final result set.
- Command **FROM** is used to retrieve the data from the CustomerReservations CTE and the AverageReservations CTE.
- Command **WHERE** is used to filters the data to include only those customers whose total number of tables reserved is greater than the average number of tables reserved across all customers.

- Command **JOIN** is not explicitly used in the main SELECT query, but the data from the CustomerReservations CTE is implicitly joined with the AverageReservations CTE by referencing avgTables in the WHERE clause.
- Command **GROUP BY** is used to group the data by customer attributes: c.name, c.contactNumber, and c.email, ensuring that the total tables reserved are calculated per customer.

Appendix 1:

```
/*----------------------------------- Create the database and tables----------------------------*/

CREATE DATABASE FoodDeliveryManagementSystem;


-- Use the database

\c FoodDeliveryManagementSystem;


-- Create Customer table

CREATE TABLE Customer (

    customerId INT PRIMARY KEY,

    name VARCHAR(100) NOT NULL,

    address VARCHAR(255) NOT NULL,

    contactNumber VARCHAR(15) NOT NULL,

    email VARCHAR(100) NOT NULL

);


-- Create Restaurant table

CREATE TABLE Restaurant (

    restaurantId INT PRIMARY KEY,

    restaurantName VARCHAR(100) NOT NULL,

    restaurantAddress VARCHAR(255) NOT NULL,

    restaurantRating DECIMAL(2,1),

    restaurantContact VARCHAR(15) NOT NULL,

    restaurantEmail VARCHAR(100) NOT NULL,

    cuisineType VARCHAR(50) NOT NULL
```

```
);

-- Create Menu table
CREATE TABLE Menu (
    itemId INT PRIMARY KEY,
    restaurantId INT REFERENCES Restaurant(restaurantId),
    itemName VARCHAR(100) NOT NULL,
    itemDescription TEXT,
    itemPrice DECIMAL(10,2) NOT NULL,
    itemType VARCHAR(50) NOT NULL
);

-- Create Order table
CREATE TABLE OrderTable (
    orderId INT PRIMARY KEY,
    customerId INT REFERENCES Customer(customerId),
    restaurantId INT REFERENCES Restaurant(restaurantId),
    orderDate DATE NOT NULL
);

-- Create Reservation table
CREATE TABLE Reservation (
    reservationId INT PRIMARY KEY,
    customerId INT REFERENCES Customer(customerId),
    restaurantId INT REFERENCES Restaurant(restaurantId),
```

```
    reservationDate DATE NOT NULL,
    numberOfPeople INT NOT NULL
);

-- Create Payment table
CREATE TABLE Payment (
    paymentId INT PRIMARY KEY,
    orderId INT REFERENCES OrderTable(orderId),
    transactionId VARCHAR(50) NOT NULL,
    paymentDate DATE NOT NULL,
    totalAmount DECIMAL(10,2) NOT NULL,
    paymentMethod VARCHAR(20) NOT NULL,
    paymentStatus VARCHAR(20) NOT NULL
);

-- Create Delivery table
CREATE TABLE Delivery (
    deliveryId INT PRIMARY KEY,
    orderId INT REFERENCES OrderTable(orderId),
    deliveryStatus VARCHAR(20) NOT NULL,
    deliveryDate DATE NOT NULL
);
```

```
/*---------------------------------- Insert values to Customer Table----------------------------*/

INSERT INTO Customer (customerId, name, address, contactNumber, email) VALUES

(1001, 'Tony Stark', '123 Elm Street', '123-456-7890', 'tony@example.com'),

(1002, 'Steve Rogers', '456 Shield Ave', '987-654-3210', 'steve@example.com'),

(1003, 'Bruce Banner', '789 Gamma Rd', '654-987-1230', 'bruce@example.com'),

(1004, 'Natasha Romanoff', '321 Spy Lane', '543-210-9876', 'natasha@example.com'),

(1005, 'Clint Barton', '101 Archer St', '555-555-5555', 'clint@example.com'),

(1006, 'Thor Odinson', '500 Thunder Rd', '999-888-7777', 'thor@example.com'),

(1007, 'Peter Parker', '20 Queens Blvd', '888-777-6666', 'peter@example.com'),

(1008, 'Wanda Maximoff', '15 Scarlet Ln', '333-333-4444', 'wanda@example.com'),

(1009, 'Vision', '42 Vibranium Rd', '222-222-5555', 'vision@example.com'),

(1010, 'Stephen Strange', '177A Bleecker St', '111-111-1234', 'strange@example.com'),

(1011, 'Sam Wilson', '98 Falcon Rd', '321-654-9870', 'sam@example.com'),

(1012, 'Bucky Barnes', '34 Winter Soldier Ave', '432-987-1234', 'bucky@example.com'),

(1013, 'Scott Lang', '10 Ant Hill', '543-321-7890', 'scott@example.com'),

(1014, 'Hope Van Dyne', '88 Wasp Ln', '654-123-4321', 'hope@example.com'),

(1015, 'Carol Danvers', '99 Captain Rd', '765-987-3456', 'carol@example.com'),

(1016, 'Nick Fury', '7 S.H.I.E.L.D. HQ', '123-123-1234', 'nick@example.com'),

(1017, 'T’Challa', '1 Wakanda Ave', '999-999-1234', 'tchalla@example.com'),

(1018, 'Shuri', '2 Wakanda Tech St', '888-888-1234', 'shuri@example.com'),

(1019, 'Loki Laufeyson', '10 Mischief Ln', '777-777-4321', 'loki@example.com'),

(1020, 'Gamora', '30 Titan Way', '666-666-1234', 'gamora@example.com'),

(1021, 'Star-Lord', '50 Galaxy Rd', '555-555-4321', 'starlord@example.com'),

(1022, 'Drax', '25 Destroyer Rd', '444-444-1234', 'drax@example.com'),

(1023, 'Rocket Raccoon', '100 Trash Panda Ln', '333-333-1234', 'rocket@example.com'),
```

(1024, 'Groot', '200 Tree Rd', '222-222-4321', 'groot@example.com'),

(1025, 'Nebula', '300 Revenge Blvd', '111-111-1111', 'nebula@example.com'),

(1026, 'Mantis', '400 Empath Rd', '123-123-9999', 'mantis@example.com'),

(1027, 'Yondu', '500 Ravager Rd', '987-987-6543', 'yondu@example.com'),

(1028, 'Hawkeye', '600 Bow St', '888-888-6666', 'hawkeye@example.com'),

(1029, 'Ant-Man', '50 Smallville', '654-654-7654', 'antman@example.com'),

(1030, 'Black Panther', '1 Vibranium St', '567-567-5678', 'blackpanther@example.com');

/*---------- Insert values to Restaurant Table----------*/

INSERT INTO Restaurant (restaurantId, restaurantName, restaurantAddress, restaurantRating, restaurantContact, restaurantEmail, cuisineType) VALUES

(3001, 'Food Paradise', '101 Main Street', 4.5, '111-111-1111', 'paradise@food.com', 'Italian'),

(3002, 'Spice Garden', '202 Spice Avenue', 4.3, '222-222-2222', 'spicegarden@food.com', 'Indian'),

(3003, 'Burger Hub', '303 Burger Lane', 4.0, '333-333-3333', 'burgerhub@food.com', 'American'),

(3004, 'Pizza Planet', '404 Cheese Street', 4.8, '444-444-4444', 'pizzaplanet@food.com', 'Italian'),

(3005, 'Taco Town', '505 Fiesta Road', 4.2, '555-555-5555', 'tacotown@food.com', 'Mexican'),

(3006, 'Noodle House', '606 Chopstick Lane', 4.1, '666-666-6666', 'noodlehouse@food.com', 'Chinese'),

(3007, 'Sushi World', '707 Wasabi Drive', 4.7, '777-777-7777', 'sushiworld@food.com', 'Japanese'),

(3008, 'Curry Corner', '808 Spice Blvd', 4.4, '888-888-8888', 'currycorner@food.com', 'Indian'),

(3009, 'Grill Station', '909 BBQ Street', 4.3, '999-999-9999', 'grillstation@food.com', 'Barbecue'),

(3010, 'Vegan Delight', '1010 Green Avenue', 4.6, '101-101-1010', 'vegandelight@food.com', 'Vegan'),

(3011, 'Pasta Palace', '1111 Italy Street', 4.7, '202-202-2020', 'pastapalace@food.com', 'Italian'),

(3012, 'The Salad Bar', '1212 Healthy Road', 4.2, '303-303-3030', 'saladbar@food.com', 'Healthy'),

(3013, 'Steakhouse Grill', '1313 Tenderloin Blvd', 4.5, '404-404-4040', 'steakhouse@food.com', 'Steakhouse'),

(3014, 'Breakfast Bliss', '1414 Sunrise Lane', 4.3, '505-505-5050', 'breakfastbliss@food.com', 'Breakfast'),

(3015, 'Falafel King', '1515 Pita Street', 4.4, '606-606-6060', 'falafelking@food.com', 'Middle Eastern'),

(3016, 'Seafood Shack', '1616 Ocean Drive', 4.6, '707-707-7070', 'seafoodshack@food.com', 'Seafood'),

(3017, 'Dessert Haven', '1717 Sweet Street', 4.9, '808-808-8080', 'desserthaven@food.com', 'Desserts'),

(3018, 'BBQ Bliss', '1818 Smoke Lane', 4.2, '909-909-9090', 'bbqbliss@food.com', 'Barbecue'),

(3019, 'Pho Place', '1919 Noodle Road', 4.4, '111-222-3333', 'phoplace@food.com', 'Vietnamese'),

(3020, 'Crepe Corner', '2020 French Lane', 4.3, '444-555-6666', 'crepecorner@food.com', 'French'),

(3021, 'Waffle Wonders', '2121 Breakfast Blvd', 4.8, '777-888-9999', 'wafflewonders@food.com', 'Breakfast'),

(3022, 'Dim Sum Delight', '2222 Dumpling Drive', 4.5, '888-999-0000', 'dimsumdelight@food.com', 'Chinese'),

(3023, 'Ramen Republic', '2323 Noodle Lane', 4.6, '111-333-4444', 'ramenrepublic@food.com', 'Japanese'),

(3024, 'Burger Barn', '2424 Patty Place', 4.0, '222-444-5555', 'burgerbarn@food.com', 'American'),

(3025, 'Tikka Junction', '2525 Spice Blvd', 4.3, '333-555-6666', 'tikkajunction@food.com', 'Indian'),

(3026, 'Gourmet Grains', '2626 Whole Grain Road', 4.4, '444-666-7777', 'gourmetgrains@food.com', 'Healthy'),

(3027, 'Fusion Flavors', '2727 Mix Street', 4.7, '555-777-8888', 'fusionflavors@food.com', 'Fusion'),

(3028, 'Biryani Bazaar', '2828 Rice Avenue', 4.6, '666-888-9999', 'biryanibazaar@food.com', 'Indian'),

(3029, 'Café Paris', '2929 Eiffel Lane', 4.9, '777-999-1111', 'cafeparis@food.com', 'French'),

(3030, 'Kebab Kitchen', '3030 Grill Lane', 4.4, '888-000-2222', 'kebabkitchen@food.com', 'Middle Eastern');

/*---------- Insert values to Menu Table----------*/

INSERT INTO Menu (itemId, restaurantId, itemName, itemDescription, itemPrice, itemType)

VALUES

-- Food Paradise (Italian)

(4001, 3001, 'Margherita Pizza', 'Classic pizza topped with tomato, mozzarella, and basil', 350.00, 'Main Course'),

(4002, 3001, 'Pasta Alfredo', 'Creamy white sauce pasta with parmesan cheese', 300.00, 'Main Course'),

(4003, 3001, 'Bruschetta', 'Toasted bread topped with tomato, garlic, and basil', 150.00, 'Appetizer'),

(4004, 3001, 'Tiramisu', 'Classic Italian dessert made with mascarpone and coffee', 250.00, 'Dessert'),

(4005, 3001, 'Minestrone Soup', 'Healthy Italian vegetable soup with pasta', 180.00, 'Starter'),

-- Spice Garden (Indian)

(4006, 3002, 'Chicken Tikka Masala', 'Grilled chicken in rich tomato-based gravy', 280.00, 'Main Course'),

(4007, 3002, 'Naan Bread', 'Soft Indian flatbread cooked in a tandoor', 40.00, 'Bread'),

(4008, 3002, 'Palak Paneer', 'Cottage cheese in creamy spinach curry', 240.00, 'Main Course'),

(4009, 3002, 'Samosa', 'Fried pastry filled with spiced potatoes and peas', 50.00, 'Appetizer'),

(4010, 3002, 'Rasgulla', 'Spongy cottage cheese balls soaked in syrup', 80.00, 'Dessert'),

-- Burger Hub (American)

(4011, 3003, 'Cheeseburger', 'Beef patty with melted cheese, lettuce, and tomato', 200.00, 'Main Course'),

(4012, 3003, 'French Fries', 'Crispy potato fries served with ketchup', 100.00, 'Side'),

(4013, 3003, 'BBQ Chicken Burger', 'Burger with tangy BBQ chicken and lettuce', 220.00, 'Main Course'),

(4014, 3003, 'Onion Rings', 'Crispy fried onion rings', 120.00, 'Appetizer'),

(4015, 3003, 'Chocolate Milkshake', 'Thick milkshake with chocolate syrup and cream', 150.00, 'Beverage'),

-- Pizza Planet (Italian)

(4016, 3004, 'Pepperoni Pizza', 'Pizza with tomato sauce, mozzarella, and pepperoni', 380.00, 'Main Course'),

(4017, 3004, 'Four Cheese Pizza', 'Pizza topped with four types of cheese', 400.00, 'Main Course'),

(4018, 3004, 'Garlic Bread', 'Toasted bread with garlic butter and herbs', 150.00, 'Appetizer'),

(4019, 3004, 'Panna Cotta', 'Creamy Italian dessert served with berry sauce', 220.00, 'Dessert'),

(4020, 3004, 'Caprese Salad', 'Tomato, mozzarella, and basil drizzled with olive oil', 180.00, 'Starter'),

-- Taco Town (Mexican)

(4021, 3005, 'Beef Tacos', 'Soft tacos filled with spiced beef and vegetables', 200.00, 'Main Course'),

(4022, 3005, 'Nachos', 'Crispy tortilla chips topped with cheese and jalapenos', 180.00, 'Appetizer'),

(4023, 3005, 'Chicken Enchiladas', 'Tortillas filled with chicken and baked with sauce', 250.00, 'Main Course'),

(4024, 3005, 'Churros', 'Fried dough pastry coated in cinnamon sugar', 120.00, 'Dessert'),

(4025, 3005, 'Guacamole', 'Creamy avocado dip served with tortilla chips', 150.00, 'Starter'),

-- Noodle House (Chinese)

(4026, 3006, 'Hakka Noodles', 'Stir-fried noodles with vegetables and soy sauce', 200.00, 'Main Course'),

(4027, 3006, 'Spring Rolls', 'Crispy rolls filled with spiced vegetables', 150.00, 'Appetizer'),

(4028, 3006, 'Kung Pao Chicken', 'Spicy stir-fried chicken with peanuts', 280.00, 'Main Course'),

(4029, 3006, 'Manchow Soup', 'Spicy Chinese soup with fried noodles', 120.00, 'Starter'),

(4030, 3006, 'Fried Rice', 'Rice stir-fried with vegetables and soy sauce', 180.00, 'Main Course'),

-- Sushi World (Japanese)

(4031, 3007, 'California Roll', 'Sushi roll with crab, avocado, and cucumber', 300.00, 'Main Course'),

(4032, 3007, 'Miso Soup', 'Traditional Japanese soup with tofu and seaweed', 150.00, 'Starter'),

(4033, 3007, 'Teriyaki Chicken', 'Grilled chicken with sweet soy-based sauce', 320.00, 'Main Course'),

(4034, 3007, 'Tempura', 'Deep-fried shrimp and vegetables', 250.00, 'Appetizer'),

(4035, 3007, 'Green Tea Ice Cream', 'Ice cream flavored with matcha', 150.00, 'Dessert'),

-- Curry Corner (Indian)

(4036, 3008, 'Butter Chicken', 'Tender chicken cooked in creamy tomato gravy', 300.00, 'Main Course'),

(4037, 3008, 'Dal Tadka', 'Lentils cooked with aromatic spices and butter', 180.00, 'Main Course'),

(4038, 3008, 'Tandoori Roti', 'Whole wheat flatbread cooked in a tandoor', 30.00, 'Bread'),

(4039, 3008, 'Gulab Jamun', 'Sweet dumplings soaked in sugar syrup', 90.00, 'Dessert'),

(4040, 3008, 'Chicken Biryani', 'Aromatic basmati rice cooked with spiced chicken', 250.00, 'Main Course'),

-- Grill Station (Barbecue)

(4041, 3009, 'BBQ Ribs', 'Slow-cooked ribs glazed with BBQ sauce', 350.00, 'Main Course'),

(4042, 3009, 'Grilled Chicken Wings', 'Spicy and smoky grilled chicken wings', 200.00, 'Appetizer'),

(4043, 3009, 'Smoked Brisket', 'Juicy beef brisket smoked to perfection', 400.00, 'Main Course'),

(4044, 3009, 'Cornbread', 'Classic BBQ side with a hint of sweetness', 80.00, 'Side'),

(4045, 3009, 'Coleslaw', 'Crunchy cabbage salad with creamy dressing', 100.00, 'Side'),

-- Vegan Delight (Vegan)

(4046, 3010, 'Quinoa Salad', 'Healthy salad with quinoa, veggies, and vinaigrette', 200.00, 'Main Course'),

(4047, 3010, 'Vegan Burger', 'Plant-based patty with lettuce and tomato', 250.00, 'Main Course'),

(4048, 3010, 'Vegan Brownie', 'Rich chocolate brownie made without dairy', 150.00, 'Dessert'),

(4049, 3010, 'Lentil Soup', 'Hearty soup made with lentils and spices', 120.00, 'Starter'),

(4050, 3010, 'Stuffed Bell Peppers', 'Bell peppers filled with quinoa and veggies', 220.00, 'Main Course'),

-- Pasta Palace (Italian)

(4051, 3011, 'Fettuccine Carbonara', 'Creamy pasta with bacon and parmesan', 320.00, 'Main Course'),

(4052, 3011, 'Garlic Knots', 'Soft bread knots brushed with garlic butter', 140.00, 'Appetizer'),

(4053, 3011, 'Lasagna', 'Layered pasta with meat, cheese, and tomato sauce', 400.00, 'Main Course'),

(4054, 3011, 'Cannoli', 'Sweet pastry filled with ricotta cheese', 180.00, 'Dessert'),

(4055, 3011, 'Caesar Salad', 'Crispy romaine lettuce with Caesar dressing', 160.00, 'Starter'),

-- The Salad Bar (Healthy)

(4056, 3012, 'Greek Salad', 'Salad with olives, feta, and fresh veggies', 180.00, 'Main Course'),

(4057, 3012, 'Kale Smoothie', 'Healthy smoothie made with kale and fruits', 150.00, 'Beverage'),

(4058, 3012, 'Avocado Toast', 'Whole-grain toast topped with mashed avocado', 200.00, 'Appetizer'),

(4059, 3012, 'Fruit Salad', 'Seasonal fruits tossed in honey and lime', 180.00, 'Dessert'),

(4060, 3012, 'Lentil Salad', 'Protein-packed salad with lentils and veggies', 190.00, 'Main Course');

/*---------- Insert values to Order Table----------*/

INSERT INTO OrderTable (orderId, customerId, restaurantId, orderDate)

VALUES

(2001, 1001, 3001, '2024-10-28'),

(2002, 1002, 3002, '2024-10-29'),

(2003, 1003, 3003, '2024-10-30'),

(2004, 1004, 3004, '2024-10-31'),

(2005, 1005, 3005, '2024-11-01'),

(2006, 1006, 3006, '2024-11-02'),

(2007, 1007, 3007, '2024-11-03'),

(2008, 1008, 3008, '2024-11-04'),

(2009, 1009, 3009, '2024-11-05'),

(2010, 1010, 3010, '2024-11-06'),

(2011, 1011, 3011, '2024-11-07'),

(2012, 1012, 3012, '2024-11-08'),

(2013, 1013, 3013, '2024-11-09'),

(2014, 1014, 3014, '2024-11-10'),

(2015, 1015, 3015, '2024-11-11'),

(2016, 1016, 3016, '2024-11-12'),

(2017, 1017, 3017, '2024-11-13'),

(2018, 1018, 3018, '2024-11-14'),

(2019, 1019, 3019, '2024-11-15'),

(2020, 1020, 3020, '2024-11-16'),

(2021, 1021, 3021, '2024-11-17'),

(2022, 1022, 3022, '2024-11-18'),

(2023, 1023, 3023, '2024-11-19'),

(2024, 1024, 3024, '2024-11-20'),

(2025, 1025, 3025, '2024-11-21'),

(2026, 1026, 3026, '2024-11-22'),

(2027, 1027, 3027, '2024-11-23'),

(2028, 1028, 3028, '2024-11-24'),

(2029, 1029, 3029, '2024-11-25'),

(2030, 1030, 3030, '2024-11-26');

/*---------- Insert values to Reservation Table----------*/

INSERT INTO Reservation (reservationId, customerId, restaurantId, reservationDate, numberOfPeople)

VALUES

(7001, 1001, 3001, '2024-11-05', 4),

(7002, 1002, 3002, '2024-11-06', 2),

(7003, 1003, 3003, '2024-11-07', 6),

(7004, 1004, 3004, '2024-11-08', 3),

(7005, 1005, 3005, '2024-11-09', 5),

(7006, 1006, 3006, '2024-11-10', 2),

(7007, 1007, 3007, '2024-11-11', 7),

(7008, 1008, 3008, '2024-11-12', 4),

(7009, 1009, 3009, '2024-11-13', 3),

(7010, 1010, 3010, '2024-11-14', 6),

(7011, 1011, 3011, '2024-11-15', 2),

(7012, 1012, 3012, '2024-11-16', 5),

(7013, 1013, 3013, '2024-11-17', 4),

(7014, 1014, 3014, '2024-11-18', 8),

(7015, 1015, 3015, '2024-11-19', 3),

(7016, 1016, 3016, '2024-11-20', 7),

(7017, 1017, 3017, '2024-11-21', 2),

(7018, 1018, 3018, '2024-11-22', 5),

(7019, 1019, 3019, '2024-11-23', 4),

(7020, 1020, 3020, '2024-11-24', 6),

(7021, 1021, 3021, '2024-11-25', 3),

(7022, 1022, 3022, '2024-11-26', 4),

(7023, 1023, 3023, '2024-11-27', 8),

(7024, 1024, 3024, '2024-11-28', 2),

(7025, 1025, 3025, '2024-11-29', 5),

(7026, 1026, 3026, '2024-11-30', 3),

(7027, 1027, 3027, '2024-12-01', 7),

(7028, 1028, 3028, '2024-12-02', 6),

(7029, 1029, 3029, '2024-12-03', 2),

(7030, 1030, 3030, '2024-12-04', 4);

```
/*---------------------------------- Insert values to Payment Table----------------------------*/

INSERT INTO Payment (paymentId, orderId, transactionId, paymentDate, totalAmount, paymentMethod, paymentStatus)

VALUES

(4001, 2001, 'TXN12345', '2024-10-28', 150.00, 'Card', 'Paid'),

(4002, 2002, 'TXN67890', '2024-10-29', 200.00, 'Cash', 'Paid'),

(4003, 2003, 'TXN34567', '2024-10-30', 120.00, 'UPI', 'Paid'),

(4004, 2004, 'TXN45678', '2024-10-31', 180.00, 'Net Banking', 'Paid'),

(4005, 2005, 'TXN56789', '2024-11-01', 250.00, 'Card', 'Paid'),

(4006, 2006, 'TXN67891', '2024-11-02', 175.00, 'Cash', 'Paid'),

(4007, 2007, 'TXN78912', '2024-11-03', 220.00, 'UPI', 'Pending'),

(4008, 2008, 'TXN89123', '2024-11-04', 130.00, 'Card', 'Paid'),

(4009, 2009, 'TXN91234', '2024-11-05', 145.00, 'Net Banking', 'Paid'),

(4010, 2010, 'TXN12346', '2024-11-06', 165.00, 'Card', 'Paid'),

(4011, 2011, 'TXN23457', '2024-11-07', 210.00, 'Cash', 'Paid'),

(4012, 2012, 'TXN34568', '2024-11-08', 140.00, 'UPI', 'Paid'),

(4013, 2013, 'TXN45679', '2024-11-09', 195.00, 'Net Banking', 'Pending'),

(4014, 2014, 'TXN56790', '2024-11-10', 205.00, 'Card', 'Paid'),

(4015, 2015, 'TXN67892', '2024-11-11', 135.00, 'Cash', 'Paid'),

(4016, 2016, 'TXN78913', '2024-11-12', 185.00, 'UPI', 'Paid'),

(4017, 2017, 'TXN89124', '2024-11-13', 160.00, 'Net Banking', 'Paid'),

(4018, 2018, 'TXN91235', '2024-11-14', 175.00, 'Card', 'Paid'),

(4019, 2019, 'TXN12347', '2024-11-15', 190.00, 'Cash', 'Paid'),

(4020, 2020, 'TXN23458', '2024-11-16', 225.00, 'UPI', 'Paid'),

(4021, 2021, 'TXN34569', '2024-11-17', 155.00, 'Card', 'Pending'),

(4022, 2022, 'TXN45680', '2024-11-18', 175.00, 'Net Banking', 'Paid'),
```

(4023, 2023, 'TXN56791', '2024-11-19', 195.00, 'Cash', 'Paid'),

(4024, 2024, 'TXN67893', '2024-11-20', 180.00, 'UPI', 'Paid'),

(4025, 2025, 'TXN78914', '2024-11-21', 165.00, 'Card', 'Paid'),

(4026, 2026, 'TXN89125', '2024-11-22', 140.00, 'Cash', 'Paid'),

(4027, 2027, 'TXN91236', '2024-11-23', 200.00, 'Net Banking', 'Paid'),

(4028, 2028, 'TXN12348', '2024-11-24', 230.00, 'Card', 'Pending'),

(4029, 2029, 'TXN23459', '2024-11-25', 215.00, 'UPI', 'Paid'),

(4030, 2030, 'TXN34570', '2024-11-26', 145.00, 'Cash', 'Paid');

/*---------- Insert values to Delivery Table----------*/

INSERT INTO Delivery (deliveryId, orderId, deliveryStatus, deliveryDate)

VALUES

(5001, 2001, 'Delivered', '2024-10-29'),

(5002, 2002, 'Pending', '2024-10-30'),

(5003, 2003, 'Delivered', '2024-10-31'),

(5004, 2004, 'Pending', '2024-11-01'),

(5005, 2005, 'Delivered', '2024-11-02'),

(5006, 2006, 'Pending', '2024-11-03'),

(5007, 2007, 'Delivered', '2024-11-04'),

(5008, 2008, 'Pending', '2024-11-05'),

(5009, 2009, 'Delivered', '2024-11-06'),

(5010, 2010, 'Pending', '2024-11-07'),

(5011, 2011, 'Delivered', '2024-11-08'),

(5012, 2012, 'Pending', '2024-11-09'),

(5013, 2013, 'Delivered', '2024-11-10'),

(5014, 2014, 'Pending', '2024-11-11'),

(5015, 2015, 'Delivered', '2024-11-12'),

(5016, 2016, 'Pending', '2024-11-13'),

(5017, 2017, 'Delivered', '2024-11-14'),

(5018, 2018, 'Pending', '2024-11-15'),

(5019, 2019, 'Delivered', '2024-11-16'),

(5020, 2020, 'Pending', '2024-11-17'),

(5021, 2021, 'Delivered', '2024-11-18'),

(5022, 2022, 'Pending', '2024-11-19'),

(5023, 2023, 'Delivered', '2024-11-20'),

(5024, 2024, 'Pending', '2024-11-21'),

(5025, 2025, 'Delivered', '2024-11-22'),

(5026, 2026, 'Pending', '2024-11-23'),

(5027, 2027, 'Delivered', '2024-11-24'),

(5028, 2028, 'Pending', '2024-11-25'),

(5029, 2029, 'Delivered', '2024-11-26'),

(5030, 2030, 'Pending', '2024-11-27');