RESTAURANT MANAGEMENT SYSTEM

Database Design Document

V 3.0

By

1	Abu Bakar	NUM-BSCS-2022-41
2	Danish Abdullah Khan	NUM-BSCS-2022-05
3	Zunaira Akbar	NUM-BSCS-2022-34

Department of Computer Sciences

Namal University

Mianwali, Pakistan

Submission Date: 30th June 2024

REVISION HISTORY

Date	**Version**	**Description**	**Approved by**
30/6/24	*V.3.0*	*Change the Database Structure a little bit just Physical Structure* *Change the Project Descriptions we can't put that context we give in Project proposal.* *Change the name of Project from "Food Management System" to "Rstaurant management System"*	**Mam Asiya**
10/6/24	*V.2.0*	*Change the ERD* *Give the attributes datatypes according to their nature.* *Define primary keys and foreign keys.* *Define the Relations between the tables*	**Mam Asiya**
22/04/24	*V 1.0*	*Specify the changes implemented after the submission of the previous document. These changes should be based on the suggestions given by the person who approved the document.*	

Instructions:

- *Place the latest revisions at the top of the table.*
- *The Revision History pertains only to changes in the document's content or any updates made after a suggestion from the approving authority. It does not apply to the template's formatting.*

TABLE OF CONTENTS

CHAPTER 1: PROJECT OVERVIEW 4

1.1. INTRODUCTION: 4

1.2. PROBLEM STATEMENT: 4

1.3. PROJECT OBJECTIVES: 4

1.4. DOCUMENT OBJECTIVES: 5

CHAPTER 2: DETAILED DATABASE DESIGN 6

2.1. ENTITY: 6

2.2. DATA DICTIONARY: 6

2.3. RELATIONSHIPS: 8

2.4. ENTITY RELATIONSHIP DIAGRAM: 8

Chapter 3 : Logical DATABASE DESIGN 9

3.1. RELATIONAL SCHEMA: 9

3.2. FUNCTIONAL DEPENDENCIES: 9

3.3. NORMALIZATION: 10

Chapter 4 : Physical DATABASE DESIGN 11

4.1. STRUCTURE OF THE TABLES: 11

4.2. DATA SAMPLES INSIDE TABLES: 12

4.3. QUERIES RESULTS: 13

Chapter 5 Interface Design 17

5.1. LANGUAGE/FRAMEWORK: 17

5.2. DATABASE CONNECTIVITY: 17

5.3. STORED PROCEDURES AND FUNCTIONS: 17

5.4. INTERFACES: 20

Chapter 6 : CONCLUSION 23

6.1. Lessons Learned 23

6.2. Challenges and Solutions 23

6.3. Future Work and Improvements 23

6.4. Final Thoughts 24

REFERENCES 24

CHAPTER 1: PROJECT OVERVIEW

1.1. INTRODUCTION:

Welcome to the Restaurant Management System project! Our goal is to create a system that helps restaurants keep track of their food inventory, monitor expiration dates, and manage usage efficiently. This system will replace the old, time-consuming, and error-prone methods, reducing food waste and improving resource management for everyone from kitchen staff to managers and suppliers.

1.2. PROBLEM STATEMENT:

Managing food items in a restaurant can be quite challenging. Currently, there's no single place to keep track of everything, making it difficult to know when food is about to spoil. Most tracking is done manually with paper files or separate spreadsheets, which can lead to mistakes and inefficiency. Our Restaurant Management System aims to solve this by offering a central database where all food-related information can be stored. This will help track available stock and automatically alert staff when items are about to expire. Additionally, it will generate detailed reports to keep everyone informed about the food inventory.

1.3. PROJECT OBJECTIVES:

Centralized Database Development:

- **Objective:** Establish a centralized database to manage food inventory data, encompassing item specifics, quantities, and expiration dates.
- **Measurable:** Finalize the database setup by the end of the semester.
- **Achievable:** Within reach using the current resources available.
- **Relevance:** Directly fulfills the requirement for structured data storage.

Reporting Capabilities:

- **Objective:** Produce reports detailing food consumption, waste, and stock levels to support data-centric decision-making.
- **Measurable:** Create and validate reporting features by the semester's conclusion.
- **Achievable:** Aligned with the project's resource capability.
- **Relevance:** Enables informed decision-making processes.

Enhancing Inventory Management Efficiency:

- **Objective:** Improve overall inventory management workflows by introducing tools for monitoring, ordering, and restocking.
- **Measurable:** Demonstrate enhanced efficiency within the semester.
- **Achievable:** Realistic considering the project's circumstances.
- **Relevance:** Addresses identified issues concerning food management.

1.4. DOCUMENT OBJECTIVES:

Introduction:

- **Purpose:** Provide an overview of the project and its importance.
- **Content:** Briefly describe the necessity for effective food inventory management and outline the system's objectives.

Problem Statement:

- **Purpose:** Clearly outline the challenges or issues targeted by the system.
- **Content:** Address the current issues concerning food inventory tracking, wastage, and data management.

Project Objectives:

- **Purpose:** Specify the system's objectives.
- **Content:**
    - **Centralized Database Creation:** Detail the objective of establishing a centralized database for food inventory data.
    - **Automated Expiration Date Tracking:** Specify the aim of efficient expiration date monitoring.
    - **Reporting Functionality:** Emphasize the importance of generating pertinent reports.
    - **Inventory Management Efficiency Improvement:** Discuss the goal of optimizing overall inventory management processes.

Database Schema:

- **Purpose:** Explain the database structure.
- **Content:** Provide information on the tables, fields, and relationships pertaining to food inventory data.

CHAPTER 2: DETAILED DATABASE DESIGN

2.1. ENTITY:

Sr. No	Entity Name	Description
01	Customer	This entity represents the customer who places the order.
02	Order	An order represents a transaction made by a customer.
03	Payment	A payment records the transaction details of an order.
04	Menu	A menu item represents the food or drink options available.
05	MenuType	A menu type categorizes the menu items.
06	OrderDetail	Order detail captures specific items and quantities in an order.
07	Rating	A rating provides customer feedback on menu items.

2.2. DATA DICTIONARY:

Customer:

Sr. No	Name	Data Type	Constraint	Description
01	CustomerID	INT	PK	Unique identifier for each customer.
02	CustomerType	VARCHAR(20)		Type of customer (e.g., regular, VIP).
03	Email	VARCHAR(50)	UNIQUE	Email address of the customer.
04	Phone	VARCHAR(20)		Phone number of the customer.
05	Address	VARCHAR(100)		Address of the customer.

Orders:

Sr. No	Name	Data Type	Constraint	Description
01	OrderID	INT	PK	Unique identifier for each order.
02	CustomerID	INT	FK	Reference to the customer who placed the order.
03	OrderDate	DATE		Date when the order was placed.

Payment:

Sr. No	Name	Data Type	Constraint	Description
01	PaymentID	INT	PK	Unique identifier for each payment.
02	OrderID	INT	FK	Reference to the order for the payment.
03	PaymentAmount	DECIMAL(10,2)		Amount paid.
04	PaymentDate	DATE		Date of the payment.
05	PaymentMethod	VARCHAR(50)		Method used for the payment.

Menu:

Sr. No	Name	Data Type	Constraint	Description
01	MenuItemID	INT	PK	Unique identifier for each menu item.
02	MenuName	VARCHAR(50)		Name of the menu item.
03	Price	DECIMAL(10,2)		Price of the menu item.
04	Description	VARCHAR(255)		Description of the menu item.
05	MenuTypeID	INT	FK	Reference to the type of menu item.

MenuType:

Sr. No	Name	Data Type	Constraint	Description
01	MenuTypeID	INT	PK	Unique identifier for each menu type.
02	TypeName	VARCHAR(50)		Name of the menu type.

OrderDetail:

Sr. No	Name	Data Type	Constraint	Description
01	OrderDetailID	INT	PK	Unique identifier for each order detail.
02	OrderID	INT	FK	Reference to the order.
03	MenuItemID	INT	FK	Reference to the menu item.
04	Quantity	INT		Quantity of the menu item ordered.
05	Price	DECIMAL(10,2)		Price of the ordered item.
06	SpecialInstructions	VARCHAR(255)		Any special instructions for the order.

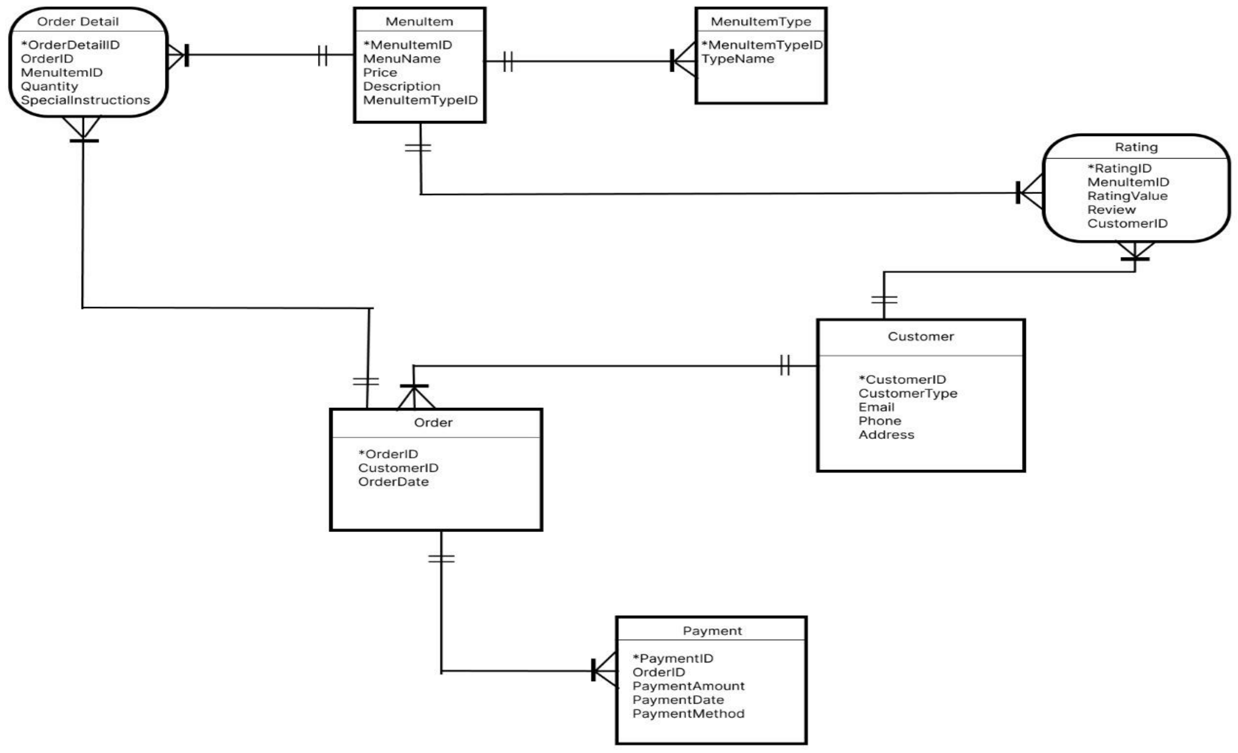
Rating:

Sr. No	Name	Data Type	Constraint	Description
01	RatingID	INT	PK	Unique identifier for each rating.
02	MenuItemID	INT	FK	Reference to the rated menu item.
03	RatingValue	INT		Rating value given by the customer.
04	Review	VARCHAR(255)		Customer review for the menu item.
05	CustomerID	INT	FK	Reference to the customer who gave the rating.

2.3. RELATIONSHIPS:

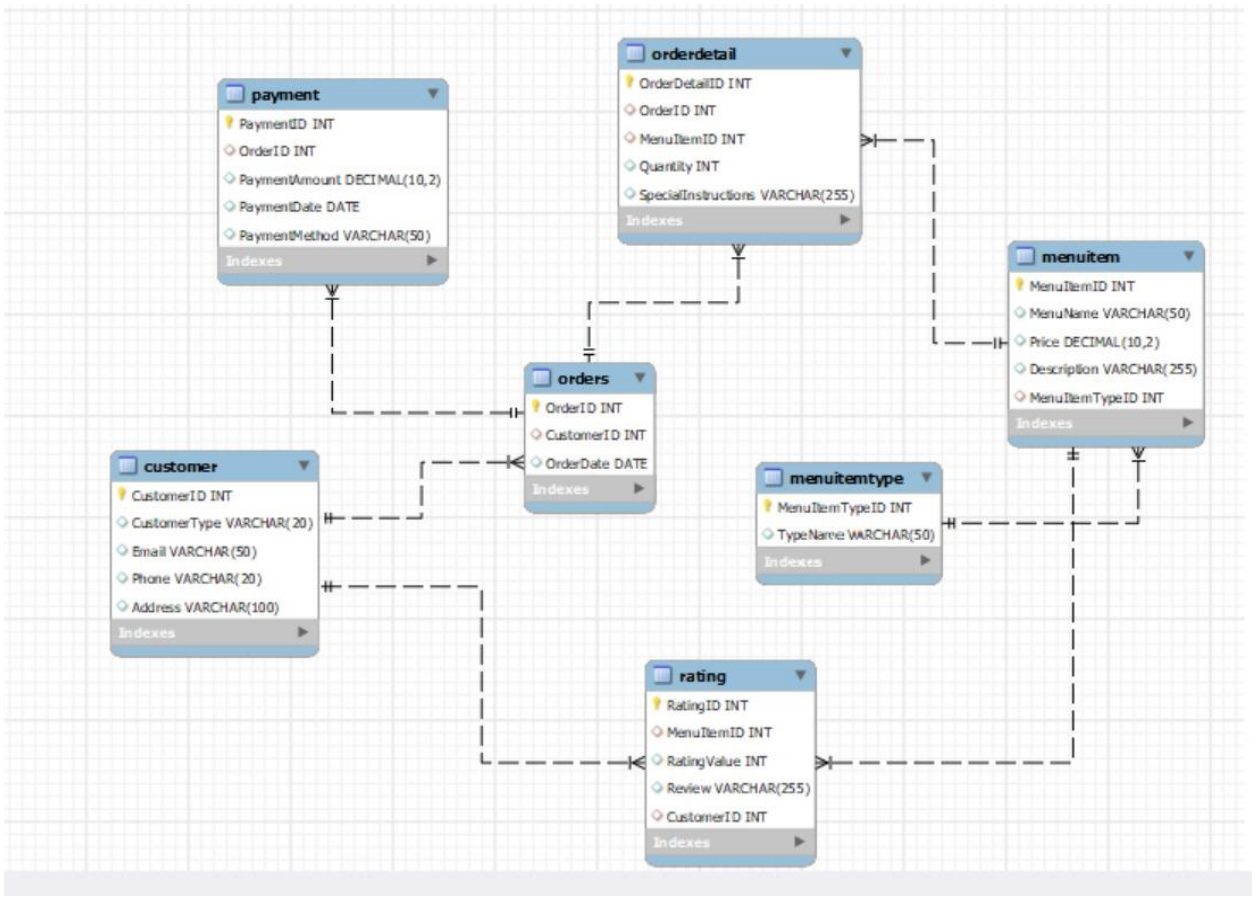
Sr. No	Participating Entities	Relation	Business Rule
01	User, Order	User places Order	A user may place multiple orders. An order is placed by exactly one user.
02	Order, Payment	Order has Payment	An order may have multiple payments. A payment is for exactly one order.
03	Order, OrderDetail	Order has OrderDetail	An order has multiple order details. An order detail belongs to exactly one order.
04	Menu, OrderDetail	Menu is in OrderDetail	A menu item may appear in multiple order details. An order detail references exactly one menu item.
05	Menu, Rating	Menu receives Rating	A menu item may receive multiple ratings. A rating is for exactly one menu item.
06	User, Rating	User gives Rating	A user may give multiple ratings. A rating is given by exactly one user.
07	MenuType, Menu	MenuType categorizes Menu	A menu type may categorize multiple menu items. A menu item belongs to exactly one menu type.

2.4. ENTITY RELATIONSHIP DIAGRAM:



CHAPTER 3 : LOGICAL DATABASE DESIGN

3.1. RELATIONAL SCHEMA:



3.2. FUNCTIONAL DEPENDENCIES:

1. Customer table:

- CustomerID → CustomerType, Email, Phone, Address

Example: If CustomerID is 1, it determines the CustomerType as 'Registered', Email as 'john@example.com', Phone as '123-456-7890', and Address as '123 Main St, Anytown'.

2. MenuType table:

- MenuTypeID → TypeName

Example: If MenuTypeID is 1, it determines the TypeName as 'Pizza'.

3. Menu table:

- MenuItemID → MenuName, Price, Description, MenuTypeID

Example: If MenuItemID is 301, it determines the MenuName as 'Margherita Pizza', Price as 12.99, Description as 'Classic pizza with tomatoes', and MenuTypeID as 1 (belonging to the 'Pizza' menu type).

4. Orders table:

- OrderID → CustomerID, OrderDate

Example: If OrderID is 101, it determines the CustomerID as 1 (the customer who placed the order) and the OrderDate as '2024-06-01'.

5. OrderDetail table:

- OrderDetailID → OrderID, MenuItemID, Quantity, Price, SpecialInstructions

Example: If OrderDetailID is 401, it determines the OrderID as 101 (the order it belongs to), MenuItemID as 301 (the specific menu item ordered), Quantity as 2, Price as 25.98 (the price of the menu item), and SpecialInstructions as 'No onions'.

6. Payment table:

- PaymentID → OrderID, PaymentAmount, PaymentDate, PaymentMethod

Example: If PaymentID is 201, it determines the OrderID as 101 (the order for which the payment was made), PaymentAmount as 50.00, PaymentDate as '2024-06-01', and PaymentMethod as 'Credit Card'.

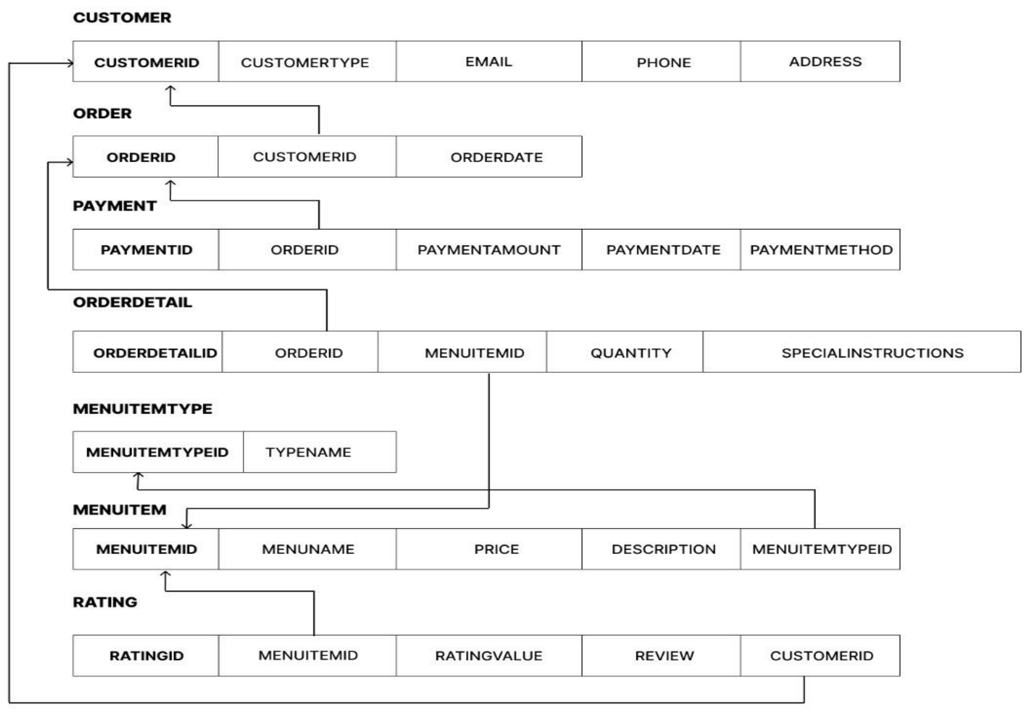
7. Rating table:

- RatingID → MenuItemID, RatingValue, Review, CustomerID

Example: If RatingID is 501, it determines the MenuItemID as 301 (the menu item being rated), RatingValue as 5 (out of 5), Review as 'Excellent taste!', and CustomerID as 1 (the customer who provided the rating).

3.3. NORMALIZATION:

Our ERD doesn't contain any anomaly, so we only draw the 3NF Normalization.



CHAPTER 4 : PHYSICAL DATABASE DESIGN

4.1. STRUCTURE OF THE TABLES:

DESCRIBE CUSTOMER;

DESCRIBE ORDERS;

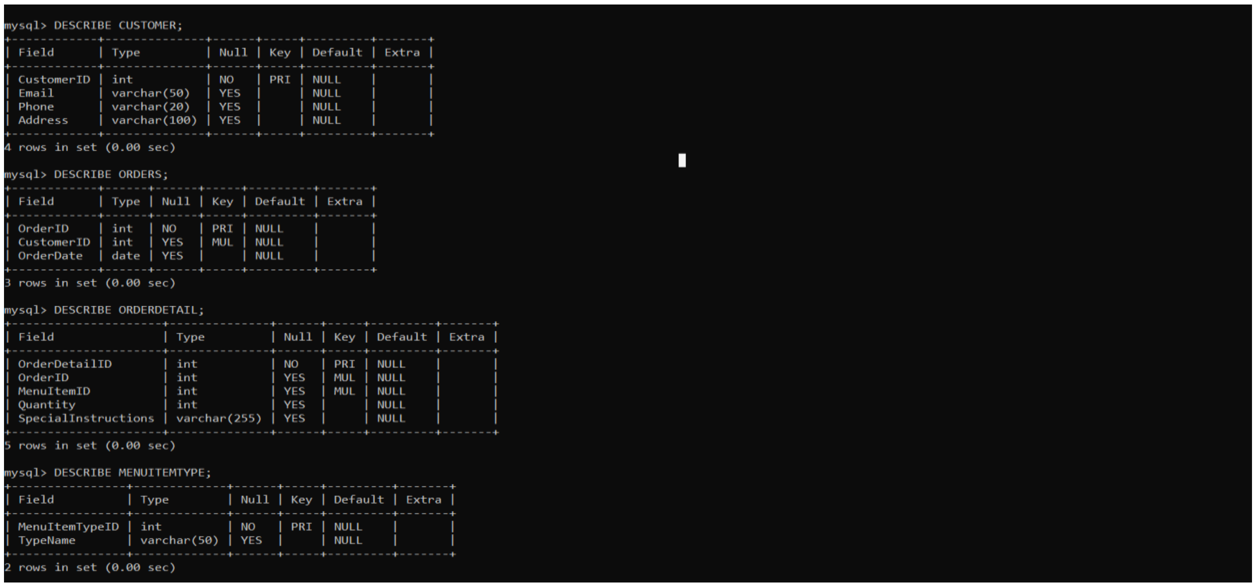
DESCRIBE ORDERDETAIL;

DESCRIBE MENUITEMTYPE;

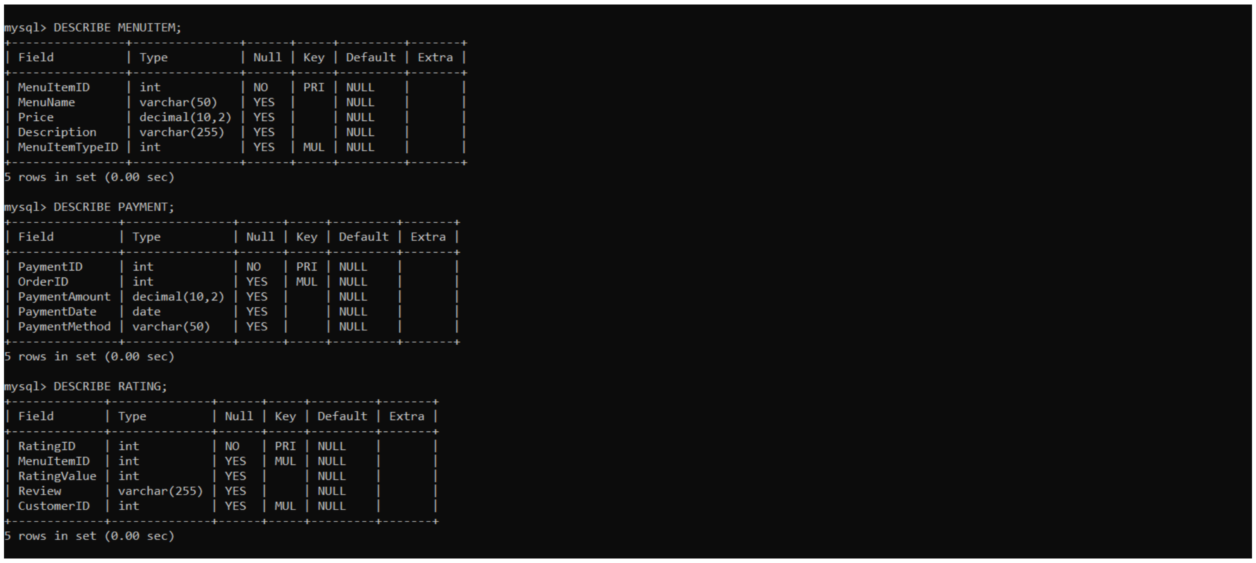
DESCRIBE MENUITEM;

DESCRIBE PAYMENT;

DESCRIBE RATING;



```
mysql> DESCRIBE CUSTOMER;
+------------+--------------+------+-----+---------+-------+
| Field      | Type         | Null | Key | Default | Extra |
+------------+--------------+------+-----+---------+-------+
| CustomerID | int          | NO   | PRI | NULL    |       |
| Email      | varchar(50)  | YES  |     | NULL    |       |
| Phone      | varchar(20)  | YES  |     | NULL    |       |
| Address    | varchar(100) | YES  |     | NULL    |       |
+------------+--------------+------+-----+---------+-------+
4 rows in set (0.00 sec)

mysql> DESCRIBE ORDERS;
+------------+------+------+-----+---------+-------+
| Field      | Type | Null | Key | Default | Extra |
+------------+------+------+-----+---------+-------+
| OrderID    | int  | NO   | PRI | NULL    |       |
| CustomerID | int  | YES  | MUL | NULL    |       |
| OrderDate  | date | YES  |     | NULL    |       |
+------------+------+------+-----+---------+-------+
3 rows in set (0.00 sec)

mysql> DESCRIBE ORDERDETAIL;
+---------------------+--------------+------+-----+---------+-------+
| Field               | Type         | Null | Key | Default | Extra |
+---------------------+--------------+------+-----+---------+-------+
| OrderDetailID       | int          | NO   | PRI | NULL    |       |
| OrderID             | int          | YES  | MUL | NULL    |       |
| MenuItemID          | int          | YES  | MUL | NULL    |       |
| Quantity            | int          | YES  |     | NULL    |       |
| SpecialInstructions | varchar(255) | YES  |     | NULL    |       |
+---------------------+--------------+------+-----+---------+-------+
5 rows in set (0.00 sec)

mysql> DESCRIBE MENUITEMTYPE;
+----------------+-------------+------+-----+---------+-------+
| Field          | Type        | Null | Key | Default | Extra |
+----------------+-------------+------+-----+---------+-------+
| MenuItemTypeID | int         | NO   | PRI | NULL    |       |
| TypeName       | varchar(50) | YES  |     | NULL    |       |
+----------------+-------------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```



```
mysql> DESCRIBE MENUITEM;
+----------------+---------------+------+-----+---------+-------+
| Field          | Type          | Null | Key | Default | Extra |
+----------------+---------------+------+-----+---------+-------+
| MenuItemID     | int           | NO   | PRI | NULL    |       |
| MenuName       | varchar(50)   | YES  |     | NULL    |       |
| Price          | decimal(10,2) | YES  |     | NULL    |       |
| Description    | varchar(255)  | YES  |     | NULL    |       |
| MenuItemTypeID | int           | YES  | MUL | NULL    |       |
+----------------+---------------+------+-----+---------+-------+
5 rows in set (0.00 sec)

mysql> DESCRIBE PAYMENT;
+---------------+---------------+------+-----+---------+-------+
| Field         | Type          | Null | Key | Default | Extra |
+---------------+---------------+------+-----+---------+-------+
| PaymentID     | int           | NO   | PRI | NULL    |       |
| OrderID       | int           | YES  | MUL | NULL    |       |
| PaymentAmount | decimal(10,2) | YES  |     | NULL    |       |
| PaymentDate   | date          | YES  |     | NULL    |       |
| PaymentMethod | varchar(50)   | YES  |     | NULL    |       |
+---------------+---------------+------+-----+---------+-------+
5 rows in set (0.00 sec)

mysql> DESCRIBE RATING;
+-------------+--------------+------+-----+---------+-------+
| Field       | Type         | Null | Key | Default | Extra |
+-------------+--------------+------+-----+---------+-------+
| RatingID    | int          | NO   | PRI | NULL    |       |
| MenuItemID  | int          | YES  | MUL | NULL    |       |
| RatingValue | int          | YES  |     | NULL    |       |
| Review      | varchar(255) | YES  |     | NULL    |       |
| CustomerID  | int          | YES  | MUL | NULL    |       |
+-------------+--------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

4.2. DATA SAMPLES INSIDE TABLES:

SELECT * FROM Customer;

SELECT * FROM Orders;

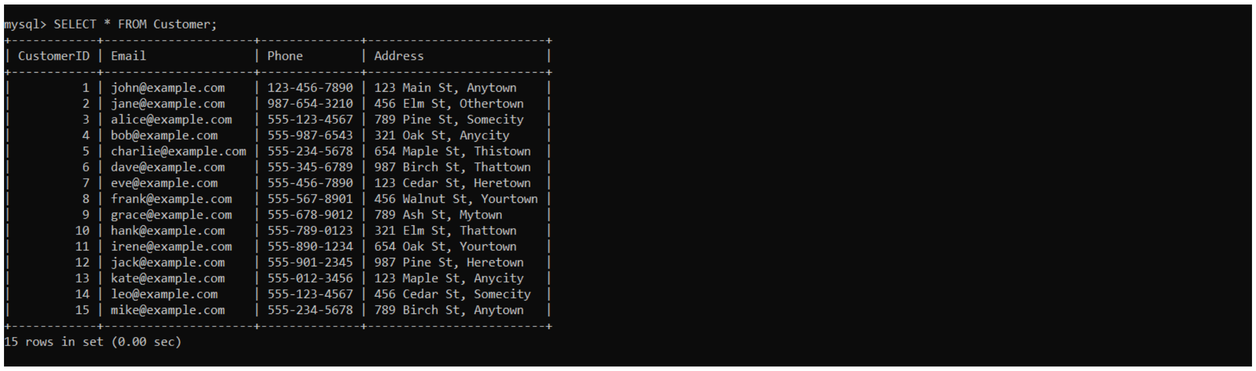
SELECT * FROM OrderDetail;

SELECT * FROM MenuItemType;

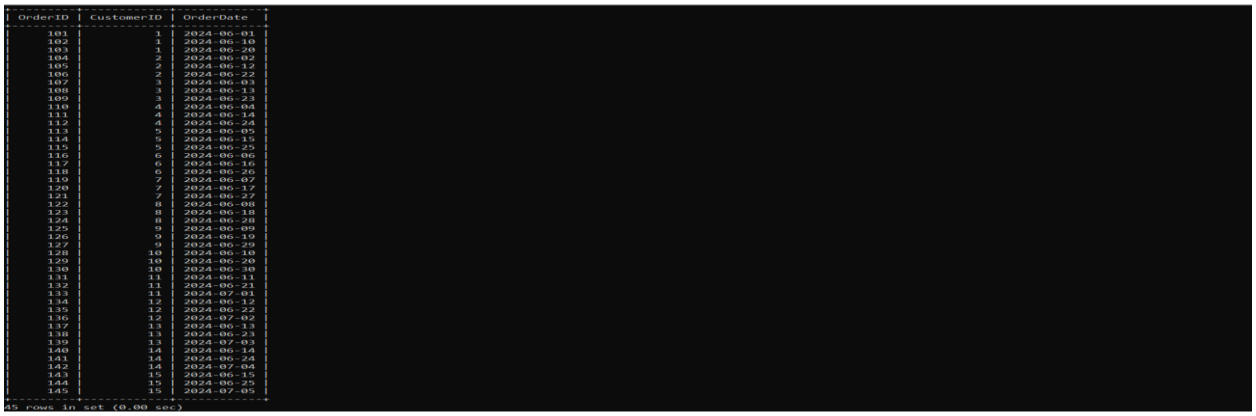
SELECT * FROM MenuItem;

SELECT * FROM Payment;

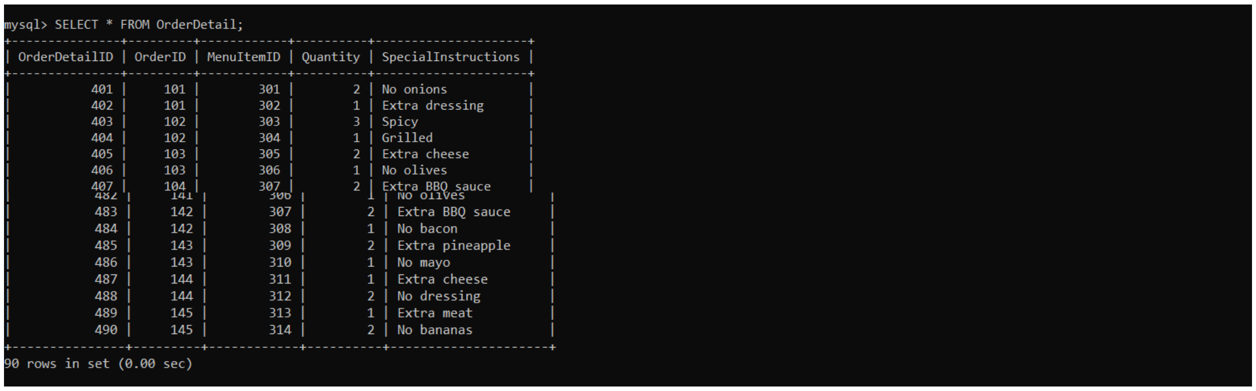
SELECT * FROM Rating;



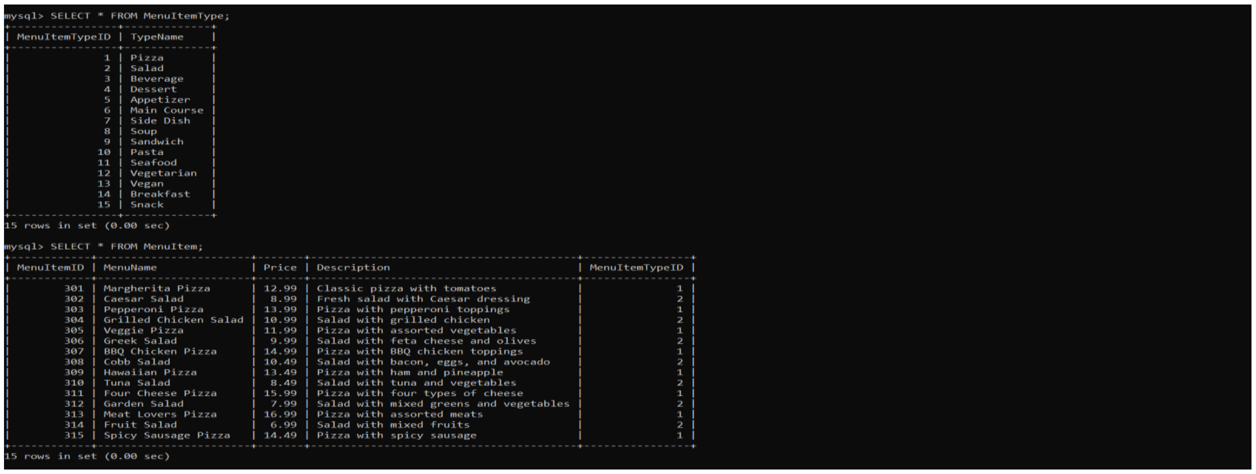
```
mysql> SELECT * FROM Customer;
+------------+---------------------+--------------+------------------------+
| CustomerID | Email               | Phone        | Address                |
+------------+---------------------+--------------+------------------------+
|          1 | john@example.com    | 123-456-7890 | 123 Main St, Anytown   |
|          2 | jane@example.com    | 987-654-3210 | 456 Elm St, Othertown  |
|          3 | alice@example.com   | 555-123-4567 | 789 Pine St, Somecity  |
|          4 | bob@example.com     | 555-987-6543 | 321 Oak St, Anycity    |
|          5 | charlie@example.com | 555-234-5678 | 654 Maple St, Thistown |
|          6 | dave@example.com    | 555-345-6789 | 987 Birch St, Thattown |
|          7 | eve@example.com     | 555-456-7890 | 123 Cedar St, Heretown |
|          8 | frank@example.com   | 555-567-8901 | 456 Walnut St, Yourtown |
|          9 | grace@example.com   | 555-678-9012 | 789 Ash St, Mytown     |
|         10 | hank@example.com    | 555-789-0123 | 321 Elm St, Thattown   |
|         11 | irene@example.com   | 555-890-1234 | 654 Oak St, Yourtown   |
|         12 | jack@example.com    | 555-901-2345 | 987 Pine St, Heretown  |
|         13 | kate@example.com    | 555-012-3456 | 123 Maple St, Anycity  |
|         14 | leo@example.com     | 555-123-4567 | 456 Cedar St, Somecity |
|         15 | mike@example.com    | 555-234-5678 | 789 Birch St, Anytown  |
+------------+---------------------+--------------+------------------------+
15 rows in set (0.00 sec)
```



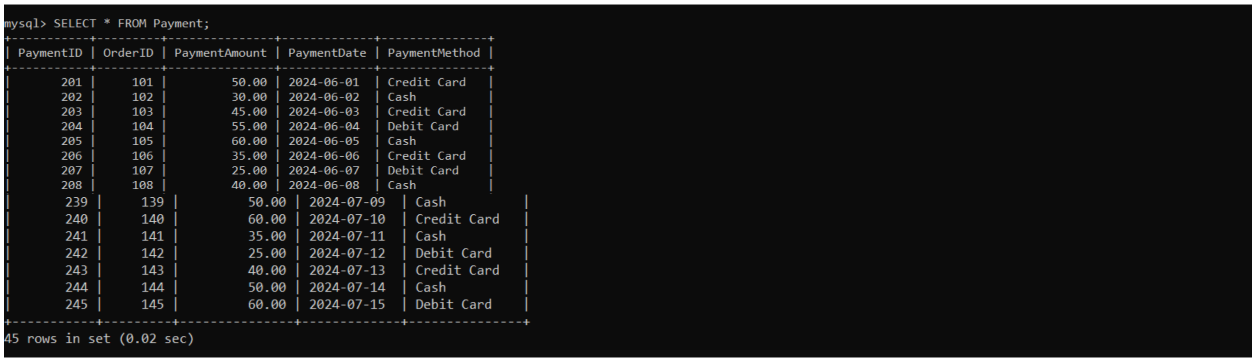
```
+---------+------------+------------+
| OrderID | CustomerID | OrderDate  |
+---------+------------+------------+
|     101 |          1 | 2024-06-01 |
|     102 |          1 | 2024-06-10 |
|     103 |          1 | 2024-06-20 |
|     104 |          2 | 2024-06-02 |
|     105 |          2 | 2024-06-12 |
|     106 |          2 | 2024-06-22 |
|     107 |          3 | 2024-06-03 |
|     108 |          3 | 2024-06-13 |
|     109 |          3 | 2024-06-23 |
|     110 |          4 | 2024-06-04 |
|     111 |          4 | 2024-06-14 |
|     112 |          4 | 2024-06-24 |
|     113 |          5 | 2024-06-05 |
|     114 |          5 | 2024-06-15 |
|     115 |          5 | 2024-06-25 |
|     116 |          6 | 2024-06-06 |
|     117 |          6 | 2024-06-16 |
|     118 |          6 | 2024-06-26 |
|     119 |          7 | 2024-06-07 |
|     120 |          7 | 2024-06-17 |
|     121 |          7 | 2024-06-27 |
|     122 |          8 | 2024-06-08 |
|     123 |          8 | 2024-06-18 |
|     124 |          8 | 2024-06-28 |
|     125 |          9 | 2024-06-09 |
|     126 |          9 | 2024-06-19 |
|     127 |          9 | 2024-06-29 |
|     128 |         10 | 2024-06-10 |
|     129 |         10 | 2024-06-20 |
|     130 |         10 | 2024-06-30 |
|     131 |         11 | 2024-06-11 |
|     132 |         11 | 2024-06-21 |
|     133 |         11 | 2024-07-01 |
|     134 |         12 | 2024-06-12 |
|     135 |         12 | 2024-06-22 |
|     136 |         12 | 2024-07-02 |
|     137 |         13 | 2024-06-13 |
|     138 |         13 | 2024-06-23 |
|     139 |         13 | 2024-07-03 |
|     140 |         14 | 2024-06-14 |
|     141 |         14 | 2024-06-24 |
|     142 |         14 | 2024-07-04 |
|     143 |         15 | 2024-06-15 |
|     144 |         15 | 2024-06-25 |
|     145 |         15 | 2024-07-05 |
+---------+------------+------------+
45 rows in set (0.00 sec)
```



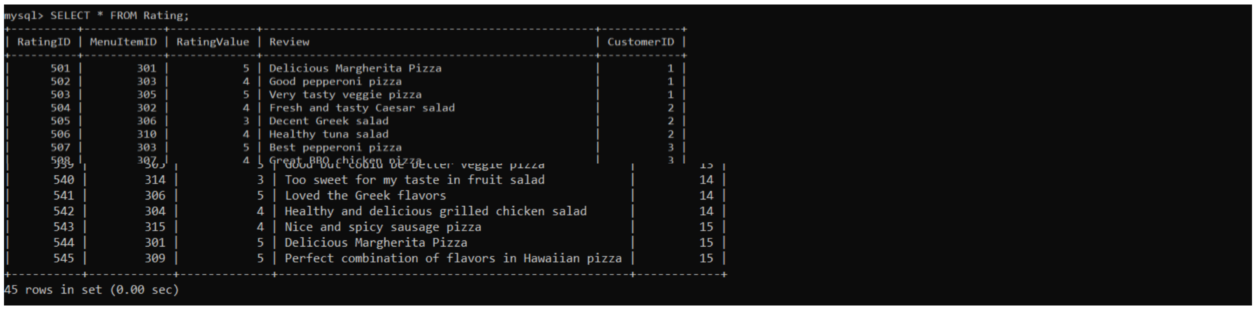
```
mysql> SELECT * FROM OrderDetail;
+---------------+---------+------------+----------+---------------------+
| OrderDetailID | OrderID | MenuItemID | Quantity | SpecialInstructions |
+---------------+---------+------------+----------+---------------------+
|           401 |     101 |        301 |        2 | No onions           |
|           402 |     101 |        302 |        1 | Extra dressing      |
|           403 |     102 |        303 |        3 | Spicy               |
|           404 |     102 |        304 |        1 | Grilled             |
|           405 |     103 |        305 |        2 | Extra cheese        |
|           406 |     103 |        306 |        1 | No olives           |
|           407 |     104 |        307 |        2 | Extra BBQ sauce     |
|           482 |     141 |        306 |        1 | No olives           |
|           483 |     142 |        307 |        2 | Extra BBQ sauce     |
|           484 |     142 |        308 |        1 | No bacon            |
|           485 |     143 |        309 |        2 | Extra pineapple     |
|           486 |     143 |        310 |        1 | No mayo             |
|           487 |     144 |        311 |        1 | Extra cheese        |
|           488 |     144 |        312 |        2 | No dressing         |
|           489 |     145 |        313 |        1 | Extra meat          |
|           490 |     145 |        314 |        2 | No bananas          |
+---------------+---------+------------+----------+---------------------+
90 rows in set (0.00 sec)
```



```
mysql> SELECT * FROM MenuItemType;
+----------------+-------------+
| MenuItemTypeID | TypeName    |
+----------------+-------------+
|              1 | Pizza       |
|              2 | Salad       |
|              3 | Beverage    |
|              4 | Dessert     |
|              5 | Appetizer   |
|              6 | Main Course |
|              7 | Side Dish   |
|              8 | Soup        |
|              9 | Sandwich    |
|             10 | Pasta       |
|             11 | Seafood     |
|             12 | Vegetarian  |
|             13 | Vegan       |
|             14 | Breakfast   |
|             15 | Snack       |
+----------------+-------------+
15 rows in set (0.00 sec)

mysql> SELECT * FROM MenuItem;
+------------+-----------------------+-------+-----------------------------------------+----------------+
| MenuItemID | MenuName              | Price | Description                             | MenuItemTypeID |
+------------+-----------------------+-------+-----------------------------------------+----------------+
|        301 | Margherita Pizza      | 12.99 | Classic pizza with tomatoes             |              1 |
|        302 | Caesar Salad          |  8.99 | Fresh salad with Caesar dressing        |              2 |
|        303 | Pepperoni Pizza       | 13.99 | Pizza with pepperoni toppings           |              1 |
|        304 | Grilled Chicken Salad | 10.99 | Salad with grilled chicken              |              2 |
|        305 | Veggie Pizza          | 11.99 | Pizza with assorted vegetables          |              1 |
|        306 | Greek Salad           |  9.99 | Salad with feta cheese and olives       |              2 |
|        307 | BBQ Chicken Pizza     | 14.99 | Pizza with BBQ chicken toppings         |              1 |
|        308 | Cobb Salad            | 10.49 | Salad with bacon, eggs, and avocado     |              2 |
|        309 | Hawaiian Pizza        | 13.49 | Pizza with ham and pineapple            |              1 |
|        310 | Tuna Salad            |  8.49 | Salad with tuna and vegetables          |              2 |
|        311 | Four Cheese Pizza     | 15.99 | Pizza with four types of cheese         |              1 |
|        312 | Garden Salad          |  7.99 | Salad with mixed greens and vegetables  |              2 |
|        313 | Meat Lovers Pizza     | 16.99 | Pizza with assorted meats               |              1 |
|        314 | Fruit Salad           |  6.99 | Salad with mixed fruits                 |              2 |
|        315 | Spicy Sausage Pizza   | 14.49 | Pizza with spicy sausage                |              1 |
+------------+-----------------------+-------+-----------------------------------------+----------------+
15 rows in set (0.00 sec)
```



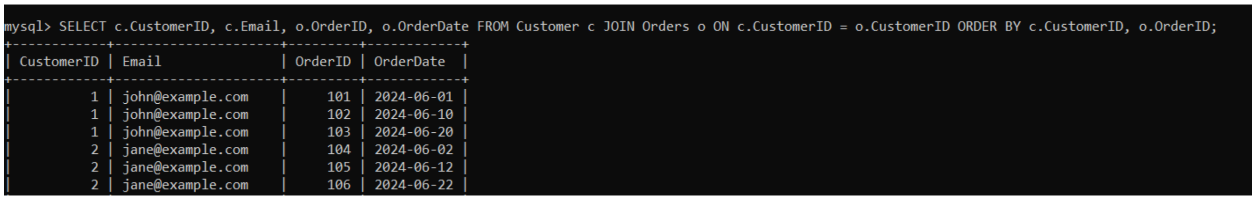
```
mysql> SELECT * FROM Payment;
+-----------+---------+---------------+-------------+---------------+
| PaymentID | OrderID | PaymentAmount | PaymentDate | PaymentMethod |
+-----------+---------+---------------+-------------+---------------+
|       201 |     101 |         50.00 | 2024-06-01  | Credit Card   |
|       202 |     102 |         30.00 | 2024-06-02  | Cash          |
|       203 |     103 |         45.00 | 2024-06-03  | Credit Card   |
|       204 |     104 |         55.00 | 2024-06-04  | Debit Card    |
|       205 |     105 |         60.00 | 2024-06-05  | Cash          |
|       206 |     106 |         35.00 | 2024-06-06  | Credit Card   |
|       207 |     107 |         25.00 | 2024-06-07  | Debit Card    |
|       208 |     108 |         40.00 | 2024-06-08  | Cash          |
|       239 |     139 |         50.00 | 2024-07-09  | Cash          |
|       240 |     140 |         60.00 | 2024-07-10  | Credit Card   |
|       241 |     141 |         35.00 | 2024-07-11  | Cash          |
|       242 |     142 |         25.00 | 2024-07-12  | Debit Card    |
|       243 |     143 |         40.00 | 2024-07-13  | Credit Card   |
|       244 |     144 |         50.00 | 2024-07-14  | Cash          |
|       245 |     145 |         60.00 | 2024-07-15  | Debit Card    |
+-----------+---------+---------------+-------------+---------------+
45 rows in set (0.02 sec)
```



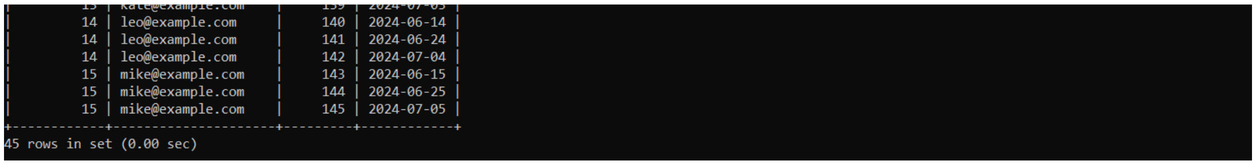
```
mysql> SELECT * FROM Rating;
+----------+------------+-------------+----------------------------------------------+------------+
| RatingID | MenuItemID | RatingValue | Review                                       | CustomerID |
+----------+------------+-------------+----------------------------------------------+------------+
|      501 |        301 |           5 | Delicious Margherita Pizza                   |          1 |
|      502 |        303 |           4 | Good pepperoni pizza                         |          1 |
|      503 |        305 |           5 | Very tasty veggie pizza                      |          1 |
|      504 |        302 |           4 | Fresh and tasty Caesar salad                 |          2 |
|      505 |        306 |           3 | Decent Greek salad                           |          2 |
|      506 |        310 |           4 | Healthy tuna salad                           |          2 |
|      507 |        303 |           5 | Best pepperoni pizza                         |          3 |
|      508 |        307 |           4 | Great BBQ chicken pizza                      |          3 |
|      539 |        305 |           5 | Good but could be better veggie pizza        |         13 |
|      540 |        314 |           3 | Too sweet for my taste in fruit salad        |         14 |
|      541 |        306 |           5 | Loved the Greek flavors                      |         14 |
|      542 |        304 |           4 | Healthy and delicious grilled chicken salad  |         14 |
|      543 |        315 |           4 | Nice and spicy sausage pizza                 |         15 |
|      544 |        301 |           5 | Delicious Margherita Pizza                   |         15 |
|      545 |        309 |           5 | Perfect combination of flavors in Hawaiian pizza |     15 |
+----------+------------+-------------+----------------------------------------------+------------+
45 rows in set (0.00 sec)
```

4.3. QUERIES RESULTS:

- SELECT c.CustomerID, c.Email, o.OrderID, o.OrderDate FROM Customer c JOIN Orders o ON c.CustomerID = o.CustomerID ORDER BY c.CustomerID, o.OrderID;

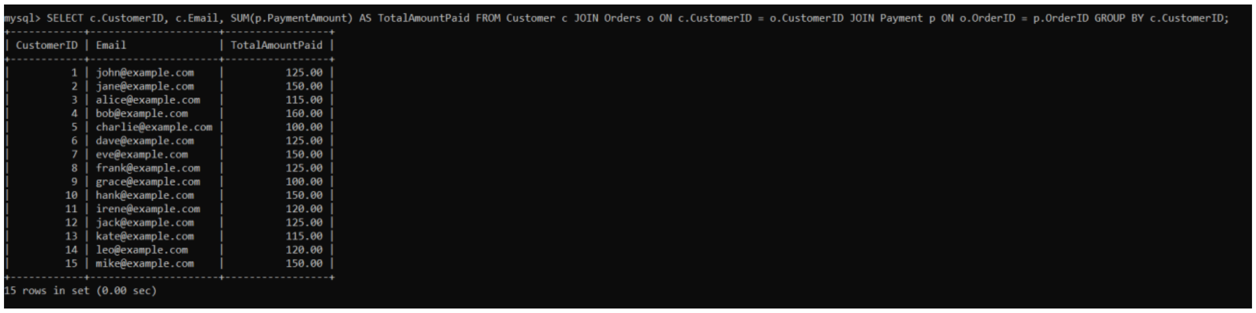


```
mysql> SELECT c.CustomerID, c.Email, o.OrderID, o.OrderDate FROM Customer c JOIN Orders o ON c.CustomerID = o.CustomerID ORDER BY c.CustomerID, o.OrderID;
+------------+---------------------+---------+------------+
| CustomerID | Email               | OrderID | OrderDate  |
+------------+---------------------+---------+------------+
|          1 | john@example.com    |     101 | 2024-06-01 |
|          1 | john@example.com    |     102 | 2024-06-10 |
|          1 | john@example.com    |     103 | 2024-06-20 |
|          2 | jane@example.com    |     104 | 2024-06-02 |
|          2 | jane@example.com    |     105 | 2024-06-12 |
|          2 | jane@example.com    |     106 | 2024-06-22 |
```



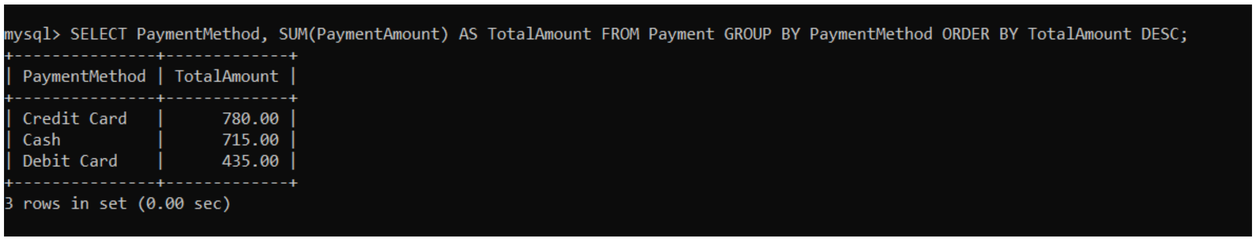
```
|         13 | kate@example.com    |     139 | 2024-07-03 |
|         14 | leo@example.com     |     140 | 2024-06-14 |
|         14 | leo@example.com     |     141 | 2024-06-24 |
|         14 | leo@example.com     |     142 | 2024-07-04 |
|         15 | mike@example.com    |     143 | 2024-06-15 |
|         15 | mike@example.com    |     144 | 2024-06-25 |
|         15 | mike@example.com    |     145 | 2024-07-05 |
+------------+---------------------+---------+------------+
45 rows in set (0.00 sec)
```

- SELECT c.CustomerID, c.Email, SUM(p.PaymentAmount) AS TotalAmountPaid FROM Customer c JOIN Orders o ON c.CustomerID = o.CustomerID JOIN Payment p ON o.OrderID = p.OrderID GROUP BY c.CustomerID;



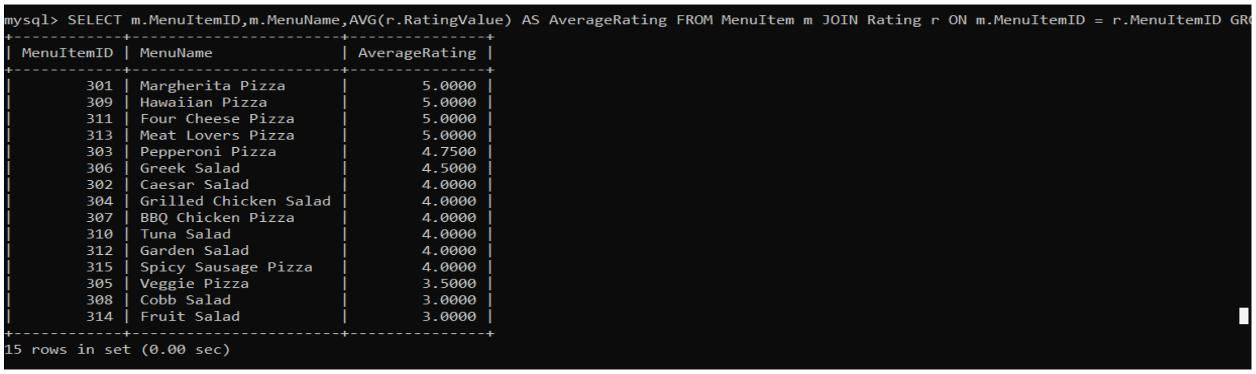
```
mysql> SELECT c.CustomerID, c.Email, SUM(p.PaymentAmount) AS TotalAmountPaid FROM Customer c JOIN Orders o ON c.CustomerID = o.CustomerID JOIN Payment p ON o.OrderID = p.OrderID GROUP BY c.CustomerID;
+------------+-------------------+-----------------+
| CustomerID | Email             | TotalAmountPaid |
+------------+-------------------+-----------------+
|          1 | john@example.com  |          125.00 |
|          2 | jane@example.com  |          150.00 |
|          3 | alice@example.com |          115.00 |
|          4 | bob@example.com   |          160.00 |
|          5 | charlie@example.com |        100.00 |
|          6 | dave@example.com  |          125.00 |
|          7 | eve@example.com   |          150.00 |
|          8 | frank@example.com |          125.00 |
|          9 | grace@example.com |          100.00 |
|         10 | hank@example.com  |          150.00 |
|         11 | irene@example.com |          120.00 |
|         12 | jack@example.com  |          125.00 |
|         13 | kate@example.com  |          115.00 |
|         14 | leo@example.com   |          120.00 |
|         15 | mike@example.com  |          150.00 |
+------------+-------------------+-----------------+
15 rows in set (0.00 sec)
```

- SELECT PaymentMethod, SUM(PaymentAmount) AS TotalAmount FROM Payment GROUP BY PaymentMethod ORDER BY TotalAmount DESC;



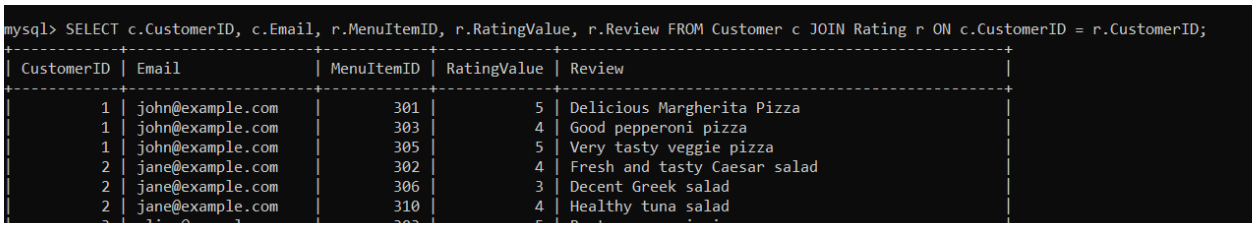
```
mysql> SELECT PaymentMethod, SUM(PaymentAmount) AS TotalAmount FROM Payment GROUP BY PaymentMethod ORDER BY TotalAmount DESC;
+---------------+-------------+
| PaymentMethod | TotalAmount |
+---------------+-------------+
| Credit Card   |      780.00 |
| Cash          |      715.00 |
| Debit Card    |      435.00 |
+---------------+-------------+
3 rows in set (0.00 sec)
```

- SELECT m.MenuItemID,m.MenuName,AVG(r.RatingValue) AS AverageRating FROM Menu m JOIN Rating r ON m.MenuItemID = r.MenuItemID GROUP BY m.MenuItemID, m.MenuName ORDER BY AverageRating DESC;

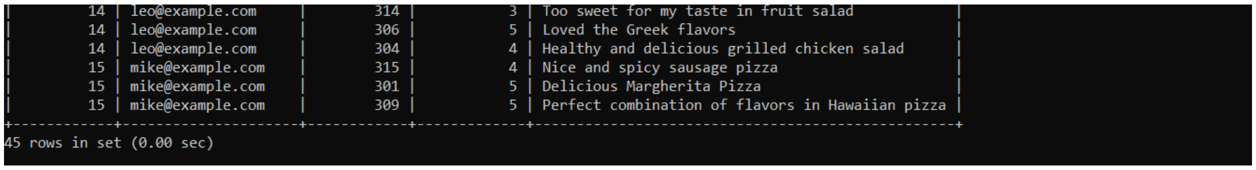


```
mysql> SELECT m.MenuItemID,m.MenuName,AVG(r.RatingValue) AS AverageRating FROM MenuItem m JOIN Rating r ON m.MenuItemID = r.MenuItemID GR
+------------+-----------------------+---------------+
| MenuItemID | MenuName              | AverageRating |
+------------+-----------------------+---------------+
|        301 | Margherita Pizza      |        5.0000 |
|        309 | Hawaiian Pizza        |        5.0000 |
|        311 | Four Cheese Pizza     |        5.0000 |
|        313 | Meat Lovers Pizza     |        5.0000 |
|        303 | Pepperoni Pizza       |        4.7500 |
|        306 | Greek Salad           |        4.5000 |
|        302 | Caesar Salad          |        4.0000 |
|        304 | Grilled Chicken Salad |        4.0000 |
|        307 | BBQ Chicken Pizza     |        4.0000 |
|        310 | Tuna Salad            |        4.0000 |
|        312 | Garden Salad          |        4.0000 |
|        315 | Spicy Sausage Pizza   |        4.0000 |
|        305 | Veggie Pizza          |        3.5000 |
|        308 | Cobb Salad            |        3.0000 |
|        314 | Fruit Salad           |        3.0000 |
+------------+-----------------------+---------------+
15 rows in set (0.00 sec)
```

- SELECT c.CustomerID, c.Email, r.MenuItemID, r.RatingValue, r.Review FROM Customer c JOIN Rating r ON c.CustomerID = r.CustomerID;

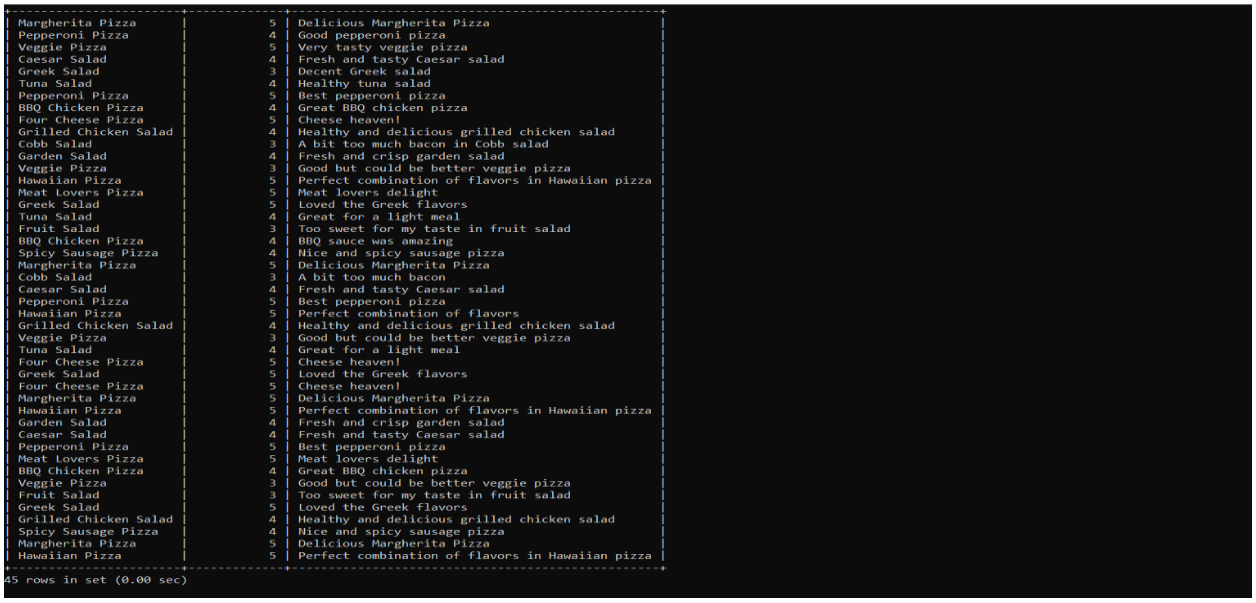


```
mysql> SELECT c.CustomerID, c.Email, r.MenuItemID, r.RatingValue, r.Review FROM Customer c JOIN Rating r ON c.CustomerID = r.CustomerID;
+------------+-------------------+------------+-------------+----------------------------------------------+
| CustomerID | Email             | MenuItemID | RatingValue | Review                                       |
+------------+-------------------+------------+-------------+----------------------------------------------+
|          1 | john@example.com  |        301 |           5 | Delicious Margherita Pizza                   |
|          1 | john@example.com  |        303 |           4 | Good pepperoni pizza                         |
|          1 | john@example.com  |        305 |           5 | Very tasty veggie pizza                      |
|          2 | jane@example.com  |        302 |           4 | Fresh and tasty Caesar salad                 |
|          2 | jane@example.com  |        306 |           3 | Decent Greek salad                           |
|          2 | jane@example.com  |        310 |           4 | Healthy tuna salad                           |
```



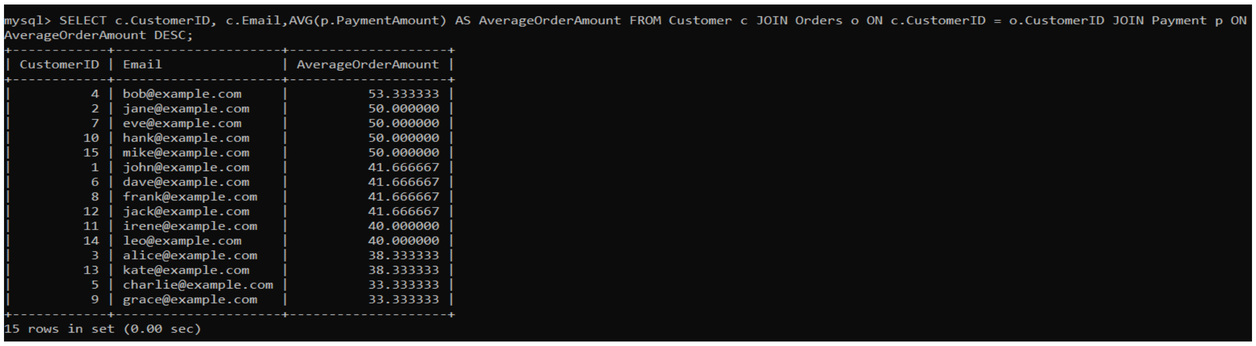
```
|         14 | leo@example.com   |        314 |           3 | Too sweet for my taste in fruit salad        |
|         14 | leo@example.com   |        306 |           5 | Loved the Greek flavors                      |
|         14 | leo@example.com   |        304 |           4 | Healthy and delicious grilled chicken salad  |
|         15 | mike@example.com  |        315 |           4 | Nice and spicy sausage pizza                 |
|         15 | mike@example.com  |        301 |           5 | Delicious Margherita Pizza                   |
|         15 | mike@example.com  |        309 |           5 | Perfect combination of flavors in Hawaiian pizza |
+------------+-------------------+------------+-------------+----------------------------------------------+
45 rows in set (0.00 sec)
```

- SELECT m.MenuName, r.RatingValue, r.Review FROM Menu m JOIN Rating r ON m.MenuItemID = r.MenuItemID WHERE r.RatingValue >= 3;



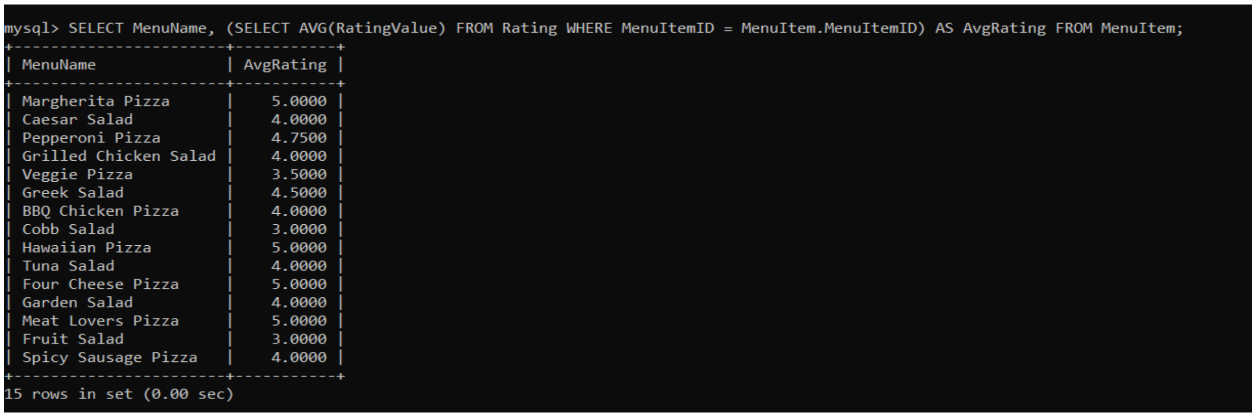
```
+-----------------------+------------+----------------------------------------------------+
| Margherita Pizza      |          5 | Delicious Margherita Pizza                         |
| Pepperoni Pizza       |          4 | Good pepperoni pizza                               |
| Veggie Pizza          |          5 | Very tasty veggie pizza                            |
| Caesar Salad          |          4 | Fresh and tasty Caesar salad                       |
| Greek Salad           |          3 | Decent Greek salad                                 |
| Tuna Salad            |          4 | Healthy tuna salad                                 |
| Pepperoni Pizza       |          5 | Best pepperoni pizza                               |
| BBQ Chicken Pizza     |          4 | Great BBQ chicken pizza                            |
| Four Cheese Pizza     |          5 | Cheese heaven!                                     |
| Grilled Chicken Salad |          4 | Healthy and delicious grilled chicken salad        |
| Cobb Salad            |          3 | A bit too much bacon in Cobb salad                 |
| Garden Salad          |          4 | Fresh and crisp garden salad                       |
| Veggie Pizza          |          3 | Good but could be better veggie pizza              |
| Hawaiian Pizza        |          5 | Perfect combination of flavors in Hawaiian pizza   |
| Meat Lovers Pizza     |          5 | Meat lovers delight                                |
| Greek Salad           |          5 | Loved the Greek flavors                            |
| Tuna Salad            |          4 | Great for a light meal                             |
| Fruit Salad           |          3 | Too sweet for my taste in fruit salad              |
| BBQ Chicken Pizza     |          4 | BBQ sauce was amazing                              |
| Spicy Sausage Pizza   |          4 | Nice and spicy sausage pizza                       |
| Margherita Pizza      |          5 | Delicious Margherita Pizza                         |
| Cobb Salad            |          3 | A bit too much bacon                               |
| Caesar Salad          |          4 | Fresh and tasty Caesar salad                       |
| Pepperoni Pizza       |          5 | Best pepperoni pizza                               |
| Hawaiian Pizza        |          5 | Perfect combination of flavors                     |
| Grilled Chicken Salad |          4 | Healthy and delicious grilled chicken salad        |
| Veggie Pizza          |          3 | Good but could be better veggie pizza              |
| Tuna Salad            |          4 | Great for a light meal                             |
| Four Cheese Pizza     |          5 | Cheese heaven!                                     |
| Greek Salad           |          5 | Loved the Greek flavors                            |
| Four Cheese Pizza     |          5 | Cheese heaven!                                     |
| Margherita Pizza      |          5 | Delicious Margherita Pizza                         |
| Hawaiian Pizza        |          5 | Perfect combination of flavors in Hawaiian pizza   |
| Garden Salad          |          4 | Fresh and crisp garden salad                       |
| Caesar Salad          |          4 | Fresh and tasty Caesar salad                       |
| Pepperoni Pizza       |          5 | Best pepperoni pizza                               |
| Meat Lovers Pizza     |          5 | Meat lovers delight                                |
| BBQ Chicken Pizza     |          4 | Great BBQ chicken pizza                            |
| Veggie Pizza          |          3 | Good but could be better veggie pizza              |
| Fruit Salad           |          3 | Too sweet for my taste in fruit salad              |
| Greek Salad           |          5 | Loved the Greek flavors                            |
| Grilled Chicken Salad |          4 | Healthy and delicious grilled chicken salad        |
| Spicy Sausage Pizza   |          4 | Nice and spicy sausage pizza                       |
| Margherita Pizza      |          5 | Delicious Margherita Pizza                         |
| Hawaiian Pizza        |          5 | Perfect combination of flavors in Hawaiian pizza   |
+-----------------------+------------+----------------------------------------------------+
45 rows in set (0.00 sec)
```

- SELECT c.CustomerID,c.CustomerType,c.Email,AVG(p.PaymentAmount) AS AverageOrderAmount FROM Customer c JOIN Orders o ON c.CustomerID = o.CustomerID JOIN Payment p ON o.OrderID = p.OrderID GROUP BY c.CustomerID ORDER BY AverageOrderAmount DESC;



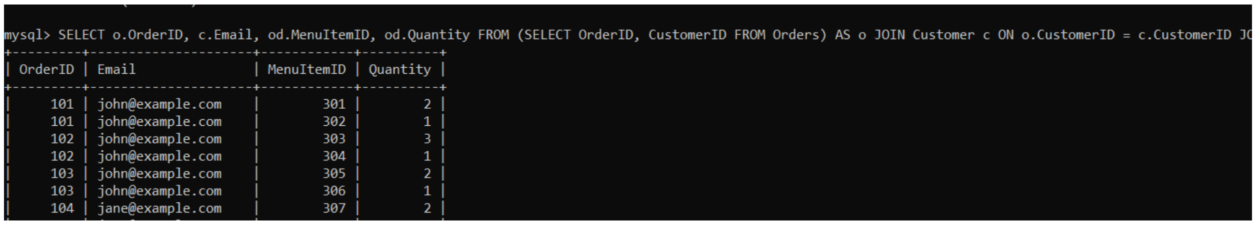
```
mysql> SELECT c.CustomerID, c.Email,AVG(p.PaymentAmount) AS AverageOrderAmount FROM Customer c JOIN Orders o ON c.CustomerID = o.CustomerID JOIN Payment p ON
AverageOrderAmount DESC;
+------------+---------------------+--------------------+
| CustomerID | Email               | AverageOrderAmount |
+------------+---------------------+--------------------+
|          4 | bob@example.com     |          53.333333 |
|          2 | jane@example.com    |          50.000000 |
|          7 | eve@example.com     |          50.000000 |
|         10 | hank@example.com    |          50.000000 |
|         15 | mike@example.com    |          50.000000 |
|          1 | john@example.com    |          41.666667 |
|          6 | dave@example.com    |          41.666667 |
|          8 | frank@example.com   |          41.666667 |
|         12 | jack@example.com    |          41.666667 |
|         11 | irene@example.com   |          40.000000 |
|         14 | leo@example.com     |          40.000000 |
|          3 | alice@example.com   |          38.333333 |
|         13 | kate@example.com    |          38.333333 |
|          5 | charlie@example.com |          33.333333 |
|          9 | grace@example.com   |          33.333333 |
+------------+---------------------+--------------------+
15 rows in set (0.00 sec)
```

- SELECT MenuName, (SELECT AVG(RatingValue) FROM Rating WHERE MenuItemID = Menu.MenuItemID) AS AvgRating FROM Menu;

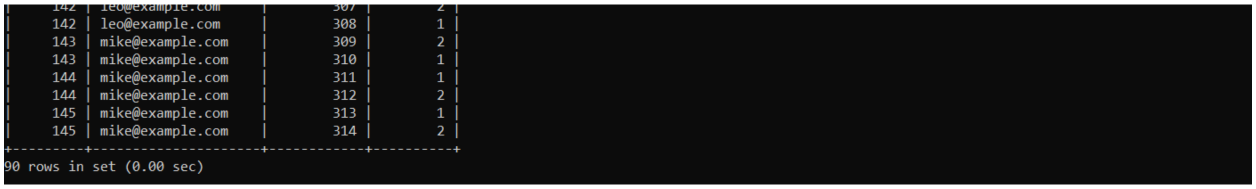


```
mysql> SELECT MenuName, (SELECT AVG(RatingValue) FROM Rating WHERE MenuItemID = MenuItem.MenuItemID) AS AvgRating FROM MenuItem;
+-----------------------+-----------+
| MenuName              | AvgRating |
+-----------------------+-----------+
| Margherita Pizza      |    5.0000 |
| Caesar Salad          |    4.0000 |
| Pepperoni Pizza       |    4.7500 |
| Grilled Chicken Salad |    4.0000 |
| Veggie Pizza          |    3.5000 |
| Greek Salad           |    4.5000 |
| BBQ Chicken Pizza     |    4.0000 |
| Cobb Salad            |    3.0000 |
| Hawaiian Pizza        |    5.0000 |
| Tuna Salad            |    4.0000 |
| Four Cheese Pizza     |    5.0000 |
| Garden Salad          |    4.0000 |
| Meat Lovers Pizza     |    5.0000 |
| Fruit Salad           |    3.0000 |
| Spicy Sausage Pizza   |    4.0000 |
+-----------------------+-----------+
15 rows in set (0.00 sec)
```

- SELECT o.OrderID, c.Email, od.MenuItemID, od.Quantity, od.Price FROM (SELECT OrderID, CustomerID FROM Orders) AS o JOIN Customer c ON o.CustomerID = c.CustomerID JOIN OrderDetail od ON o.OrderID = od.OrderID;

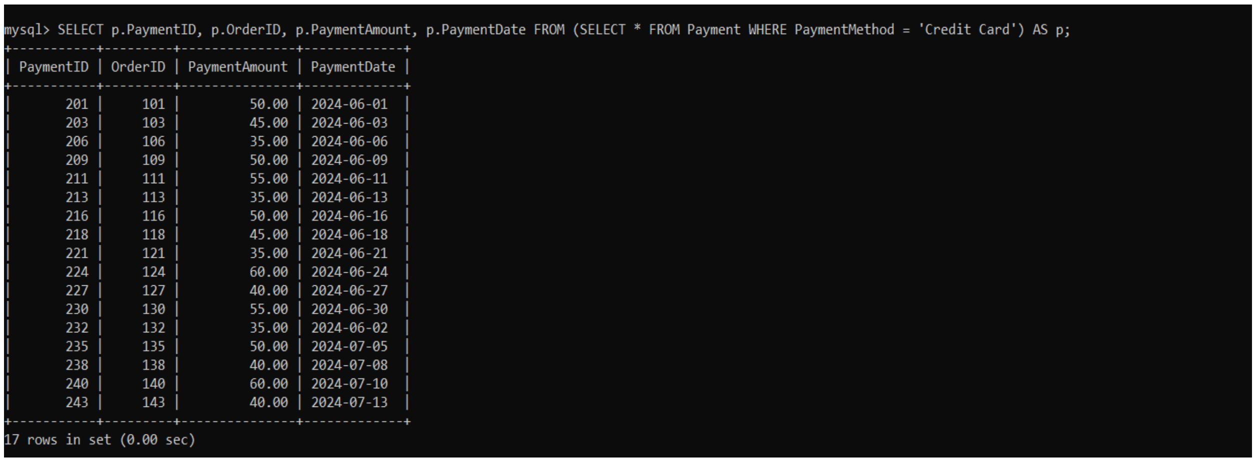


```
mysql> SELECT o.OrderID, c.Email, od.MenuItemID, od.Quantity FROM (SELECT OrderID, CustomerID FROM Orders) AS o JOIN Customer c ON o.CustomerID = c.CustomerID J
+---------+--------------------+------------+----------+
| OrderID | Email              | MenuItemID | Quantity |
+---------+--------------------+------------+----------+
|     101 | john@example.com   |        301 |        2 |
|     101 | john@example.com   |        302 |        1 |
|     102 | john@example.com   |        303 |        3 |
|     102 | john@example.com   |        304 |        1 |
|     103 | john@example.com   |        305 |        2 |
|     103 | john@example.com   |        306 |        1 |
|     104 | jane@example.com   |        307 |        2 |
```



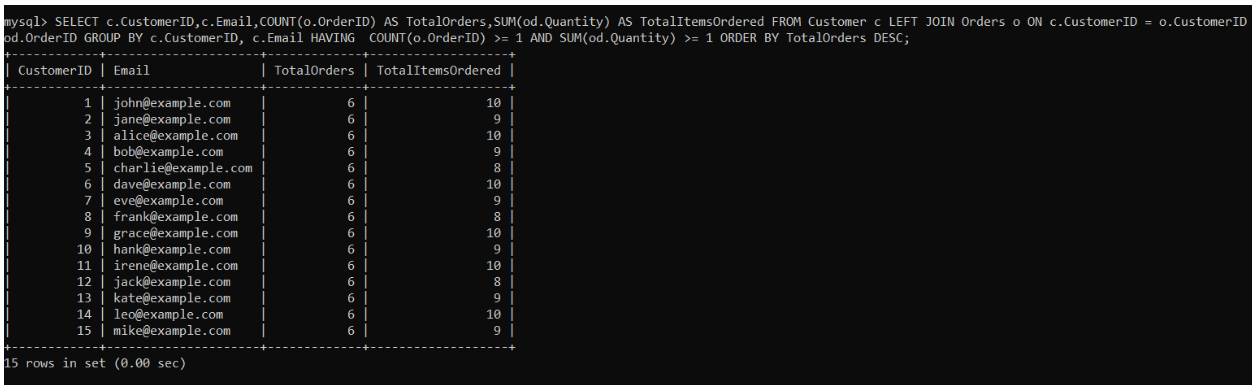
```
|     142 | leo@example.com    |        307 |        2 |
|     142 | leo@example.com    |        308 |        1 |
|     143 | mike@example.com   |        309 |        2 |
|     143 | mike@example.com   |        310 |        1 |
|     144 | mike@example.com   |        311 |        1 |
|     144 | mike@example.com   |        312 |        2 |
|     145 | mike@example.com   |        313 |        1 |
|     145 | mike@example.com   |        314 |        2 |
+---------+--------------------+------------+----------+
90 rows in set (0.00 sec)
```

- SELECT p.PaymentID, p.OrderID, p.PaymentAmount, p.PaymentDate FROM (SELECT * FROM Payment WHERE PaymentMethod = 'Credit Card') AS p;



```
mysql> SELECT p.PaymentID, p.OrderID, p.PaymentAmount, p.PaymentDate FROM (SELECT * FROM Payment WHERE PaymentMethod = 'Credit Card') AS p;
+-----------+---------+---------------+-------------+
| PaymentID | OrderID | PaymentAmount | PaymentDate |
+-----------+---------+---------------+-------------+
|       201 |     101 |         50.00 | 2024-06-01  |
|       203 |     103 |         45.00 | 2024-06-03  |
|       206 |     106 |         35.00 | 2024-06-06  |
|       209 |     109 |         50.00 | 2024-06-09  |
|       211 |     111 |         55.00 | 2024-06-11  |
|       213 |     113 |         35.00 | 2024-06-13  |
|       216 |     116 |         50.00 | 2024-06-16  |
|       218 |     118 |         45.00 | 2024-06-18  |
|       221 |     121 |         35.00 | 2024-06-21  |
|       224 |     124 |         60.00 | 2024-06-24  |
|       227 |     127 |         40.00 | 2024-06-27  |
|       230 |     130 |         55.00 | 2024-06-30  |
|       232 |     132 |         35.00 | 2024-06-02  |
|       235 |     135 |         50.00 | 2024-07-05  |
|       238 |     138 |         40.00 | 2024-07-08  |
|       240 |     140 |         60.00 | 2024-07-10  |
|       243 |     143 |         40.00 | 2024-07-13  |
+-----------+---------+---------------+-------------+
17 rows in set (0.00 sec)
```

- SELECT c.CustomerID,c.Email,COUNT(o.OrderID) AS TotalOrders,SUM(od.Quantity) AS TotalItemsOrdered FROM Customer c LEFT JOIN Orders o ON c.CustomerID = o.CustomerID LEFT JOIN OrderDetail od ON o.OrderID = od.OrderID GROUP BY c.CustomerID, c.Email HAVING COUNT(o.OrderID) >= 1 AND SUM(od.Quantity) >= 1 ORDER BY TotalOrders DESC;



```
mysql> SELECT c.CustomerID,c.Email,COUNT(o.OrderID) AS TotalOrders,SUM(od.Quantity) AS TotalItemsOrdered FROM Customer c LEFT JOIN Orders o ON c.CustomerID = o.CustomerID
od.OrderID GROUP BY c.CustomerID, c.Email HAVING  COUNT(o.OrderID) >= 1 AND SUM(od.Quantity) >= 1 ORDER BY TotalOrders DESC;
+------------+---------------------+-------------+-------------------+
| CustomerID | Email               | TotalOrders | TotalItemsOrdered |
+------------+---------------------+-------------+-------------------+
|          1 | john@example.com    |           6 |                10 |
|          2 | jane@example.com    |           6 |                 9 |
|          3 | alice@example.com   |           6 |                10 |
|          4 | bob@example.com     |           6 |                 9 |
|          5 | charlie@example.com |           6 |                 8 |
|          6 | dave@example.com    |           6 |                10 |
|          7 | eve@example.com     |           6 |                 9 |
|          8 | frank@example.com   |           6 |                 8 |
|          9 | grace@example.com   |           6 |                10 |
|         10 | hank@example.com    |           6 |                 9 |
|         11 | irene@example.com   |           6 |                10 |
|         12 | jack@example.com    |           6 |                 8 |
|         13 | kate@example.com    |           6 |                 9 |
|         14 | leo@example.com     |           6 |                10 |
|         15 | mike@example.com    |           6 |                 9 |
+------------+---------------------+-------------+-------------------+
15 rows in set (0.00 sec)
```

CHAPTER 5 INTERFACE DESIGN

5.1. LANGUAGE/FRAMEWORK:

We chose Python for our GUI client because it's easy to understand and write, allowing for quick development. We used Tkinter, a simple yet powerful library for building GUIs in Python. By combining Python and Tkinter with Visual Studio, we created a robust environment that makes coding and debugging efficient. Tkinter supports various widgets and features like event handling and customization, making our GUI client versatile and powerful for different use cases.

5.2. DATABASE CONNECTIVITY:

To connect our GUI client with the MySQL database, we used the MySQL-connector-python library. This library allows Python applications to interact easily with MySQL databases. After installing the library with "***pip install mysql-connector-python,"*** we used credentials like host, username, password, and database name to set up the connection. We wrapped our connection and database calls in a try-except-finally block to manage errors effectively and ensure that connections are properly closed to avoid resource leaks.

5.3. STORED PROCEDURES AND FUNCTIONS:

Here's a list of stored procedures and their objectives for our Restaurant Management System:

Customer Management System

- **InsertCustomer**: Add new customer records to the database.
- **UpdateCustomer**: Modify existing customer records.
- **DeleteCustomer**: Remove customer records from the database.
- **SearchCustomersByEmail**: Search for customers based on their email.

Order Management System

- **InsertOrder**: Add new orders.
- **UpdateOrder**: Update existing orders.
- **DeleteOrder**: Delete orders.
- **SearchOrdersByCustomerID**: Search orders by CustomerID.

Menu Item Management System

- **InsertMenuItem**: Insert a new menu item into the MenuItem table.
- **UpdateMenuItem**: Update an existing menu item in the MenuItem table.
- **DeleteMenuItem**: Delete an existing menu item from the MenuItem table.
- **SearchMenuItemsByMenuItemTypeID**: Retrieve menu items based on a specific type ID.

Menu Item Type Management System

- **InsertMenuItemType**: Insert a new menu item type into the MenuItemType table.
- **UpdateMenuItemType**: Update an existing menu item type in the MenuItemType table.
- **DeleteMenuItemType**: Delete an existing menu item type from the MenuItemType table.
- **SearchMenuItemTypes**: Retrieve menu item types based on the type name.

Order Detail Management System

- **InsertOrderDetail**: Insert a new order detail into the OrderDetail table.
- **UpdateOrderDetail**: Update an existing order detail in the OrderDetail table.
- **DeleteOrderDetail**: Delete an existing order detail from the OrderDetail table.
- **SearchOrderDetailsByOrderID**: Retrieve order details based on a specific order ID.

Payment Management System

- **InsertPayment**: Insert a new payment into the Payment table.
- **UpdatePayment**: Update an existing payment in the Payment table.
- **DeletePayment**: Delete an existing payment from the Payment table.
- **SearchPaymentsByOrderID**: Retrieve payments based on a specific order ID.

Rating Management System

- **InsertRating**: Insert a new rating into the Rating table.
- **UpdateRating**: Update an existing rating in the Rating table.
- **DeleteRating**: Delete an existing rating from the Rating table.
- **SearchRatingsByMenuItemID**: Retrieve ratings based on a specific menu item ID.
- **SearchRatingsByCustomerID**: Retrieve ratings based on a specific customer ID.

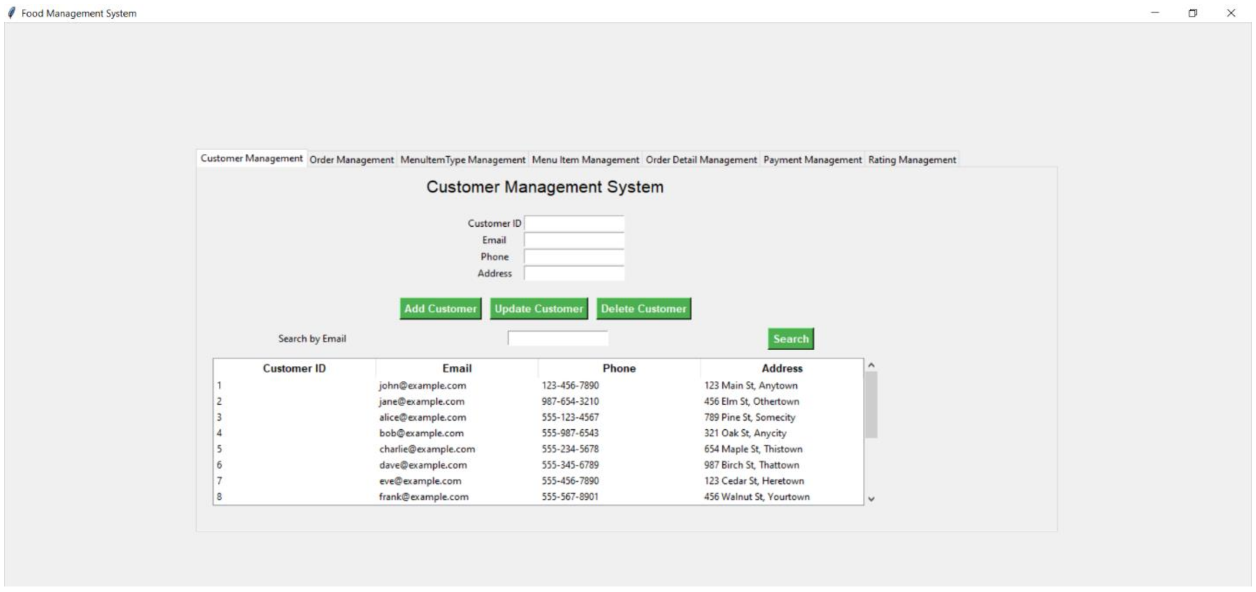
STORED PROCEDURES CODE

```sql
DELIMITER //
CREATE PROCEDURE InsertCustomer(IN p_email VARCHAR(50), IN p_phone VARCHAR(20),
IN p_address VARCHAR(100)) BEGIN INSERT INTO Customer (Email, Phone, Address)
VALUES (p_email, p_phone, p_address); END //
CREATE PROCEDURE UpdateCustomer(IN p_customer_id INT, IN p_email VARCHAR(50), IN
p_phone VARCHAR(20), IN p_address VARCHAR(100)) BEGIN UPDATE Customer SET Email =
p_email, Phone = p_phone, Address = p_address WHERE CustomerID = p_customer_id;
END //
CREATE PROCEDURE DeleteCustomer(IN p_customer_id INT) BEGIN DELETE FROM Customer
WHERE CustomerID = p_customer_id; END //
CREATE PROCEDURE SearchCustomersByEmail(IN p_email VARCHAR(50)) BEGIN SELECT *
FROM Customer WHERE Email LIKE CONCAT('%', p_email, '%'); END //
```

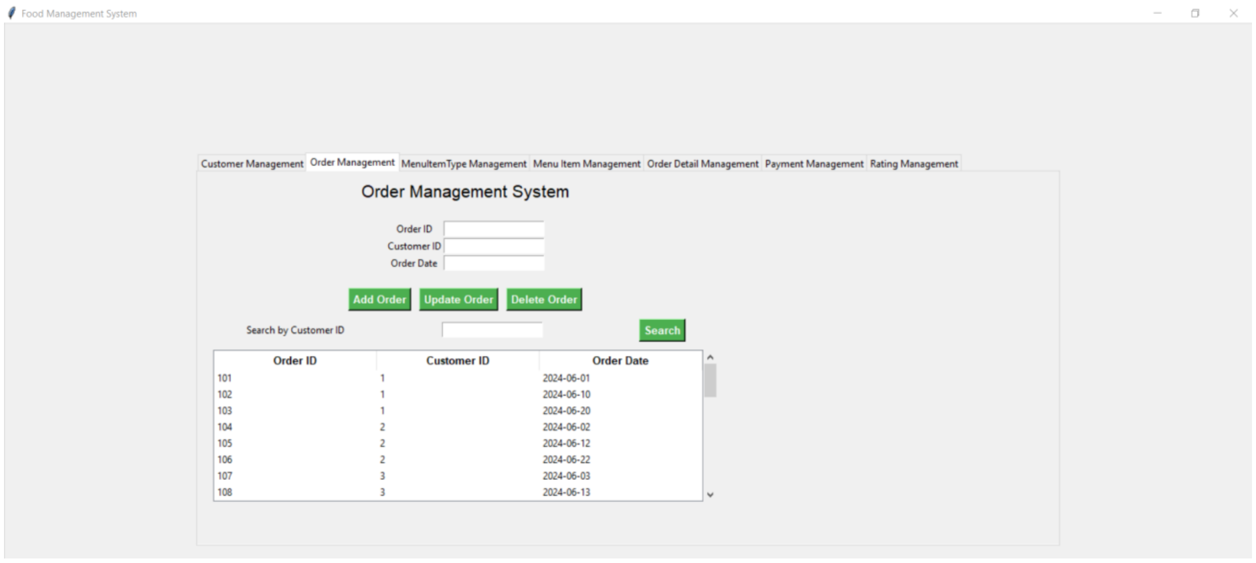
```
CREATE PROCEDURE InsertOrder(IN p_CustomerID INT, IN p_OrderDate DATE) BEGIN
INSERT INTO Orders (CustomerID, OrderDate) VALUES (p_CustomerID, p_OrderDate);
END //
CREATE PROCEDURE UpdateOrder(IN p_OrderID INT, IN p_CustomerID INT, IN
p_OrderDate DATE) BEGIN UPDATE Orders SET CustomerID = p_CustomerID, OrderDate =
p_OrderDate WHERE OrderID = p_OrderID; END //
CREATE PROCEDURE DeleteOrder(IN p_OrderID INT) BEGIN DELETE FROM Orders WHERE
OrderID = p_OrderID; END //
CREATE PROCEDURE SearchOrdersByCustomerID(IN p_CustomerID INT) BEGIN SELECT *
FROM Orders WHERE CustomerID = p_CustomerID; END //

CREATE PROCEDURE InsertMenuItemType(IN type_name VARCHAR(50)) BEGIN INSERT INTO
MenuItemType (TypeName) VALUES (type_name); END //
CREATE PROCEDURE UpdateMenuItemType(IN type_id INT, IN type_name VARCHAR(50))
BEGIN UPDATE MenuItemType SET TypeName = type_name WHERE MenuItemTypeID =
type_id; END //
CREATE PROCEDURE DeleteMenuItemType(IN type_id INT) BEGIN DELETE FROM
MenuItemType WHERE MenuItemTypeID = type_id; END //
CREATE PROCEDURE SearchMenuItemTypes() BEGIN SELECT * FROM MenuItemType; END //

CREATE PROCEDURE InsertMenuItem(IN menu_name VARCHAR(50), IN price DECIMAL(10,
2), IN description VARCHAR(255), IN type_id INT) BEGIN INSERT INTO MenuItem
(MenuName, Price, Description, MenuItemTypeID) VALUES (menu_name, price,
description, type_id); END //
CREATE PROCEDURE UpdateMenuItem(IN item_id INT, IN menu_name VARCHAR(50), IN
price DECIMAL(10, 2), IN description VARCHAR(255), IN type_id INT) BEGIN UPDATE
MenuItem SET MenuName = menu_name, Price = price, Description = description,
MenuItemTypeID = type_id WHERE MenuItemID = item_id; END //
CREATE PROCEDURE DeleteMenuItem(IN item_id INT) BEGIN DELETE FROM MenuItem WHERE
MenuItemID = item_id; END //
CREATE PROCEDURE SearchMenuItemsByMenuItemTypeID(IN type_id INT) BEGIN SELECT *
FROM MenuItem WHERE MenuItemTypeID = type_id; END //

CREATE PROCEDURE InsertOrderDetail(IN order_id INT, IN item_id INT, IN quantity
INT, IN instructions VARCHAR(255)) BEGIN INSERT INTO OrderDetail (OrderID,
MenuItemID, Quantity, SpecialInstructions) VALUES (order_id, item_id, quantity,
instructions); END //
CREATE PROCEDURE UpdateOrderDetail(IN detail_id INT, IN order_id INT, IN item_id
INT, IN quantity INT, IN instructions VARCHAR(255)) BEGIN UPDATE OrderDetail SET
OrderID = order_id, MenuItemID = item_id, Quantity = quantity,
SpecialInstructions = instructions WHERE OrderDetailID = detail_id; END //
CREATE PROCEDURE DeleteOrderDetail(IN detail_id INT) BEGIN DELETE FROM
OrderDetail WHERE OrderDetailID = detail_id; END //
CREATE PROCEDURE SearchOrderDetailsByOrderID(IN order_id INT) BEGIN SELECT * FROM
OrderDetail WHERE OrderID = order_id; END //
```

```
CREATE PROCEDURE InsertPayment(IN order_id INT, IN amount DECIMAL(10, 2), IN pay_date DATE, IN method VARCHAR(50)) BEGIN INSERT INTO Payment (OrderID, PaymentAmount, PaymentDate, PaymentMethod) VALUES (order_id, amount, pay_date, method); END //
CREATE PROCEDURE UpdatePayment(IN payment_id INT, IN order_id INT, IN amount DECIMAL(10, 2), IN pay_date DATE, IN method VARCHAR(50)) BEGIN UPDATE Payment SET PaymentAmount = amount, PaymentDate = pay_date, PaymentMethod = method WHERE PaymentID = payment_id AND orderID = order_id; END //
CREATE PROCEDURE DeletePayment(IN payment_id INT) BEGIN DELETE FROM Payment WHERE PaymentID = payment_id; END //
CREATE PROCEDURE SearchPaymentsByOrderID(IN order_id INT) BEGIN SELECT * FROM Payment WHERE OrderID = order_id; END //

CREATE PROCEDURE InsertRating(IN item_id INT, IN rating_value INT, IN review VARCHAR(255), IN cust_id INT) BEGIN INSERT INTO Rating (MenuItemID, RatingValue, Review, CustomerID) VALUES (item_id, rating_value, review, cust_id); END //
CREATE PROCEDURE UpdateRating(IN rating_id INT, IN menu_item_id INT, IN rating_value INT, IN review VARCHAR(255), IN customer_id INT) BEGIN UPDATE Rating SET MenuItemID = menu_item_id, RatingValue = rating_value, Review = review, CustomerID = customer_id WHERE RatingID = rating_id; END //
CREATE PROCEDURE DeleteRating(IN rating_id INT) BEGIN DELETE FROM Rating WHERE RatingID = rating_id; END //
CREATE PROCEDURE SearchRatingsByMenuItemID(IN item_id INT) BEGIN SELECT * FROM Rating WHERE MenuItemID = item_id; END //
DELIMITER;
```

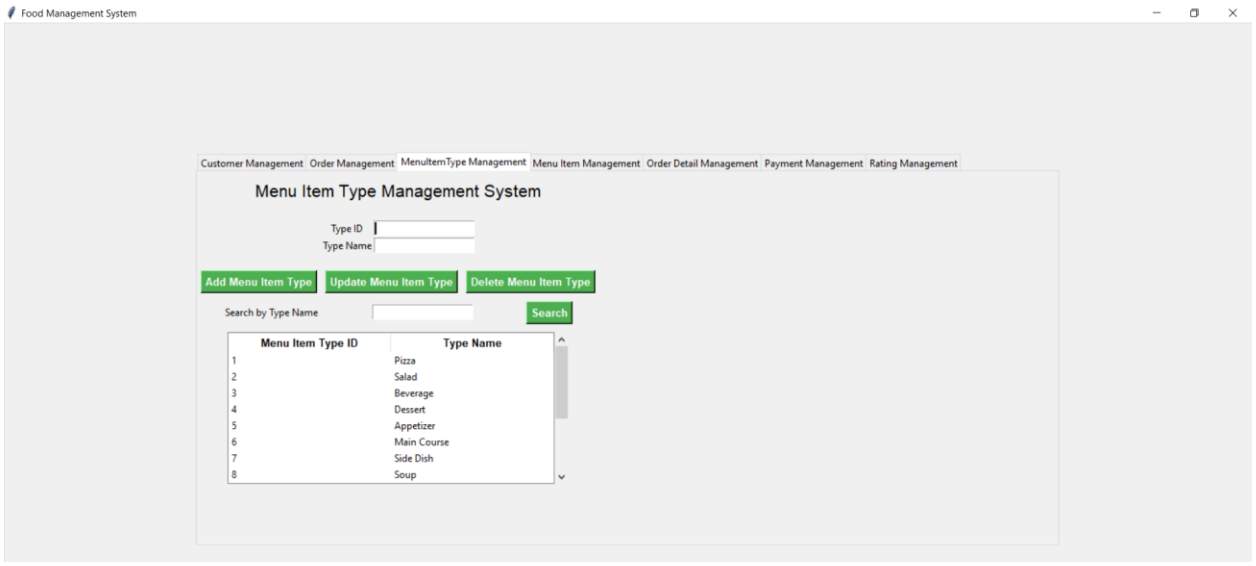
5.4. INTERFACES:



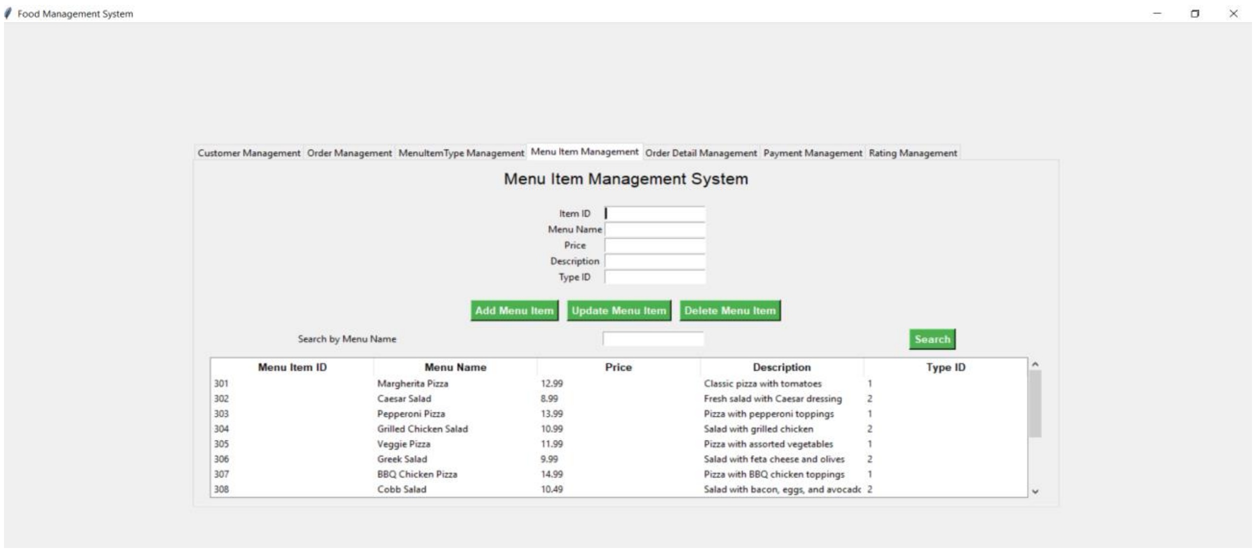
Customer ID	Email	Phone	Address
1	john@example.com	123-456-7890	123 Main St, Anytown
2	jane@example.com	987-654-3210	456 Elm St, Othertown
3	alice@example.com	555-123-4567	789 Pine St, Somecity
4	bob@example.com	555-987-6543	321 Oak St, Anycity
5	charlie@example.com	555-234-5678	654 Maple St, Thistown
6	dave@example.com	555-345-6789	987 Birch St, Thattown
7	eve@example.com	555-456-7890	123 Cedar St, Heretown
8	frank@example.com	555-567-8901	456 Walnut St, Yourtown



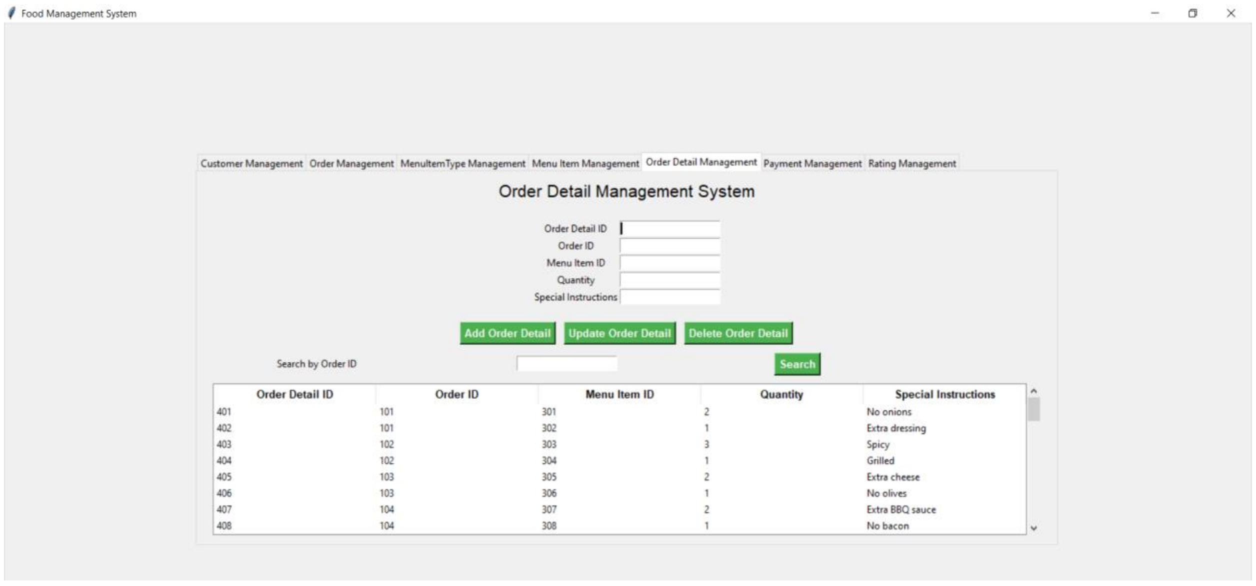
Order ID	Customer ID	Order Date
101	1	2024-06-01
102	1	2024-06-10
103	1	2024-06-20
104	2	2024-06-02
105	2	2024-06-12
106	2	2024-06-22
107	3	2024-06-03
108	3	2024-06-13



Menu Item Type ID	Type Name
1	Pizza
2	Salad
3	Beverage
4	Dessert
5	Appetizer
6	Main Course
7	Side Dish
8	Soup



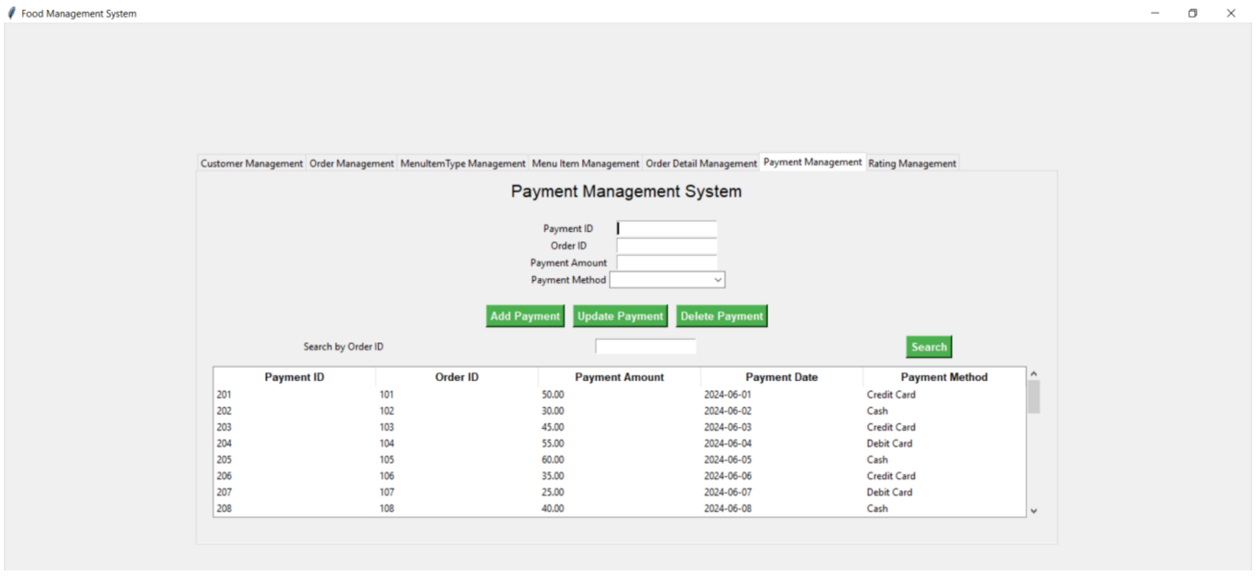
Menu Item ID	Menu Name	Price	Description	Type ID
301	Margherita Pizza	12.99	Classic pizza with tomatoes	1
302	Caesar Salad	8.99	Fresh salad with Caesar dressing	2
303	Pepperoni Pizza	13.99	Pizza with pepperoni toppings	1
304	Grilled Chicken Salad	10.99	Salad with grilled chicken	2
305	Veggie Pizza	11.99	Pizza with assorted vegetables	1
306	Greek Salad	9.99	Salad with feta cheese and olives	2
307	BBQ Chicken Pizza	14.99	Pizza with BBQ chicken toppings	1
308	Cobb Salad	10.49	Salad with bacon, eggs, and avocadc	2



Order Detail Management System

Order Detail ID
Order ID
Menu Item ID
Quantity
Special Instructions

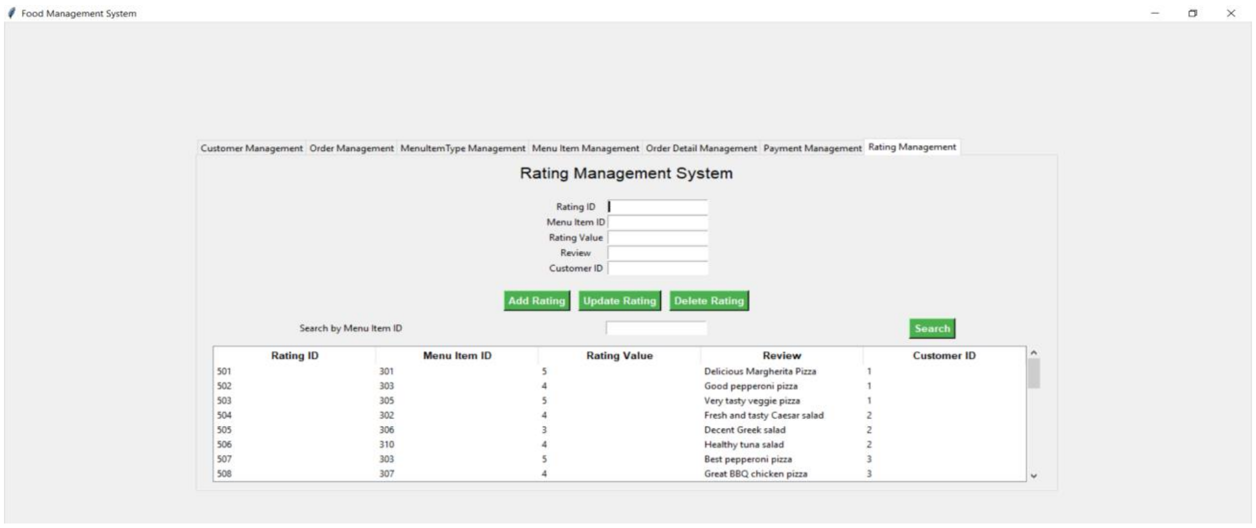
Order Detail ID	Order ID	Menu Item ID	Quantity	Special Instructions
401	101	301	2	No onions
402	101	302	1	Extra dressing
403	102	303	3	Spicy
404	102	304	1	Grilled
405	103	305	2	Extra cheese
406	103	306	1	No olives
407	104	307	2	Extra BBQ sauce
408	104	308	1	No bacon



Payment Management System

Payment ID
Order ID
Payment Amount
Payment Method

Payment ID	Order ID	Payment Amount	Payment Date	Payment Method
201	101	50.00	2024-06-01	Credit Card
202	102	30.00	2024-06-02	Cash
203	103	45.00	2024-06-03	Credit Card
204	104	55.00	2024-06-04	Debit Card
205	105	60.00	2024-06-05	Cash
206	106	35.00	2024-06-06	Credit Card
207	107	25.00	2024-06-07	Debit Card
208	108	40.00	2024-06-08	Cash



Rating Management System

Rating ID
Menu Item ID
Rating Value
Review
Customer ID

Rating ID	Menu Item ID	Rating Value	Review	Customer ID
501	301	5	Delicious Margherita Pizza	1
502	303	4	Good pepperoni pizza	1
503	305	5	Very tasty veggie pizza	1
504	302	4	Fresh and tasty Caesar salad	2
505	306	3	Decent Greek salad	2
506	310	4	Healthy tuna salad	2
507	303	5	Best pepperoni pizza	3
508	307	4	Great BBQ chicken pizza	3

CHAPTER 6 : CONCLUSION

6.1. Lessons Learned

Technical Skills:

Database Management: Gained a solid understanding of MySQL, including stored procedures, database creation, data insertion, MySQL queries.
UI Development: Improved skills with Tkinter to build user-friendly Python interfaces.
Error Handling: Enhanced ability to manage exceptions in Python and MySQL, making the application more dependable.

Project Management:

Time Management: Learned to effectively divide time between development, testing, and documentation.
Documentation: Recognized the importance of keeping detailed and up-to-date documentation for smooth project flow and handover.

Collaboration:

Teamwork: Benefited from effective communication and teamwork, leading to better problem-solving and creativity.
Feedback: Understood the importance of incorporating feedback from team members and users to improve the project.

6.2. Challenges and Solutions

Design Challenges:

Database Schema: Ensuring a complete and normalized database schema was challenging, but iterative design and team reviews helped.

Implementation Challenges:

Stored Procedures: Overcoming the complexity of stored procedures was achieved through extensive study and testing.

Testing Challenges:

Integration Testing: Ensuring all components worked together smoothly required both automated scripts and thorough manual testing.

6.3. Future Work and Improvements

Additional Features:

User **Roles and Permissions:** Implement a user roles and permissions system to manage various levels of access (e.g., admin, manager, customer).
Order **History:** Maintain a detailed order history for customers, allowing them to view past orders and reorder easily.

Optimizations:

Performance Tuning: Improve database query and stored procedure performance for large datasets.
UI Enhancements: Make the GUI more responsive and user-friendly, using modern frameworks like VS Code with Tkinter.

Broader Applications:

Mobile App: Create a mobile version for managing ratings on the go.
Data Analytics: Integrate analytics to provide insights and trends, aiding food management decisions.

6.4. Final Thoughts

Personal Insights:

Growth: This project was a major learning experience in both technical skills and project management.
Real-World Impact: Building an application that addresses a real need was incredibly rewarding.

Overall Impact:

User Impact: The system will make managing ratings and reviews more efficient, boosting customer satisfaction.
Professional Development: This project has significantly contributed to our professional growth, preparing us for future challenges.

Acknowledgments:

- **Team Members:** Abubakar, Appi Zunaira Akbar, Danish Abdullah Khan
- **Instructor:** Mam Asiya Batool
- **Special Thanks:** ChatGPT and Google for their help in creating Python GUIs.

REFERENCES

MySQL Documentation: *https://dev.mysql.com/doc/*

Tkinter Documentation: *https://docs.python.org/3/library/tkinter.html*