NATIONAL INSTITUTE OF TECHNOLOGY,

DURGAPUR

Department of Computer Science Engineering

Durgapur, West Bengal

Mini Project Report

Title:Online FoodOrdering System

Submitted in partial fulfillment for the requirement of

Data Base Management Systems in

the 4th Semester 2025

Submitted by:

Name:Lingampalli Sirija

Roll No:23CS8097

Under the guidance of

Prasenjit Choudhury Associate Professor https://nitdgp.ac.in/department/computer-science engineering/faculty-1/prasenjit-choudhury

SUBJECT : DBMS Lab SUBJECT CODE :- CSS 453

Acknowledgement

We express our heartfelt gratitude to everyone who has supported us throughout the development of this mini project, *Online Food Ordering System*, submitted in partial fulfillment of the requirements for the course *Database Management Systems* at the National Institute of Technology, Durgapur.

First and foremost, we extend our deepest appreciation to our project guide, **Dr. Prasenjit Choudhury**, Associate Professor, Department of Computer Science and Engineering, for his invaluable guidance, encouragement, and expertise. His insightful suggestions and constant motivation have been instrumental in shaping this project and enhancing our understanding of database systems.

We are also grateful to the **Department of Computer Science and Engineering** at NIT Durgapur for providing us with the resources and academic environment necessary to undertake this endeavor. The knowledge imparted through lectures and practical sessions has laid a strong foundation for this work.

We would also like to acknowledge the contributions of online resources such as *W3Schools*, *GeeksforGeeks*, and the *MySQL Documentation*, which served as valuable references for mastering SQL and database concepts during the project's development. Finally, we extend our gratitude to our families and friends for their unwavering support, patience, and encouragement throughout this academic journey. Their belief in us has been a constant source of inspiration.

This project would not have been possible without the contributions of all these individuals and resources, and for that, we are truly thankful.

Abstract

In today's fast-paced world, the *Online Food Ordering System* emerges as a game-changer, blending efficiency with innovation! This mini project unveils a **robust SQL-based backend database** that seamlessly connects customers, restaurants, and their favorite meals. Crafted with precision, it captures critical data—customer details, restaurant profiles, food items, and orders—through a relational design powered by **foreign keys**, **joins**, and **normalization**. From tracking top-selling dishes to calculating revenue with slick SQL queries, this system delivers **data-driven insights** like never before. Built under the guidance of **Dr. Prasenjit Choudhury** at NIT Durgapur, it's a scalable foundation ready to fuel real-world food ordering apps. Hungry for efficiency? This project's got the recipe!

Contents

Introduction

The Online Food Ordering System is designed to provide a robust backend database that streamlines food orders between customers and restaurants. This system captures essential information related to customers, restaurants, orders, and food items. The system is built entirely with SQL and focuses solely on backend logic without using any frontend interfaces. All operations and analytics are performed within a virtual database environment using SQL queries.

Objective

 Efficient Database Design: Creating a scalable SQL-based relational backend to manage customers, restaurants, food items, and orders with robust organization and integrity
 Proficient SQL Operations: Demonstrating expertise in insertion, updates, aggregation, filtering, and joins for real-world scenarios
 Entity Relationship Modeling: Designing a clear Entity Relationship Diagram (ERD) for structured system mapping Data-Driven Insights: Utilizing SQL queries for analytics like customer patterns, restaurant order volumes, and popular items. **Practical Application:** Bridging database theory and practice while preparing for future enhancements like frontend integration and API development.

This project showcases technical skills, teamwork, and innovation to meet academic and real-world demands.

Scope of the Project

The scope of your Online Food Ordering System project is focused on the backend database system and includes the following:

- Database design and schema definition.
- Use of foreign keys and relational integrity.
- Execution of real-world SQL queries.
- Analytical reporting using SQL statements.
- Data normalization and entity relationships

Problem Statement

In an era where convenience reigns supreme, the food ordering landscape faces a critical challenge: inefficient management of data between customers, restaurants, and orders. Traditional systems—whether manual or poorly digitized—struggle with disorganized records, slow processing, and limited scalability, leaving both customers and restaurant owners hungry for a better solution. The absence of a streamlined, centralized database system often leads to errors in order tracking, delays in delivery, and missed opportunities for data-driven insights like customer preferences or revenue trends. For students and developers learning database management, this real-world issue poses an exciting yet complex problem to solve.

The core problem lies in designing a backend system that can: 1.
Handle diverse entities—customers, restaurants, food items, and orders—while maintaining relational integrity.

2. Execute real-time operations such as inserting new orders, updating customer details, or querying restaurant performance, all without a frontend crutch.
3. Provide actionable analytics—think total revenue, popular dishes, or repeat customers—using only SQL-based logic.
4. Scale effortlessly to accommodate growing data without compromising efficiency or accuracy.

Without such a system, businesses risk data redundancy, inconsistent records, and poor decision-making, while learners miss a chance to master practical database skills. This project, undertaken as part of the *Database Management Systems* course at NIT Durgapur, tackles these hurdles head-on. By building a robust SQL-driven backend, we aim to bridge the gap between theoretical database concepts and their application in a dynamic, food-ordering context—delivering a solution that's as functional as it is educational.

Objective of the Project

The Online Food Ordering System is more than just a database—it's a mission to master data management with purpose! Crafted as part of the Database Management Systems course at NIT Durgapur, this project sets out to achieve the following objectives:

Design an Efficient Database: Build a scalable, relational backend using SQL to seamlessly manage customers, restaurants, food items, and orders, ensuring robust data organization and integrity through foreign keys and normalization.

Master SQL Operations: Demonstrate proficiency in a wide range of SQL techniques—insertion, updates, aggregation, filtering, and joins—to handle real-world food ordering scenarios with precision and speed. Model Entity Relationships: Create a clear and logical Entity-Relationship Diagram (ERD) to map the connections between entities, providing a blueprint for a structured and interconnected system.

Deliver Data-Driven Insights: Leverage SQL queries to extract meaningful analytics, such as customer spending patterns, restaurant order volumes, and popular food items, empowering decision-making with actionable results. Bridge Theory and Practice: Apply core database concepts learned in the classroom to a practical, industry-relevant problem, preparing the groundwork for future enhancements like frontend integration or API development.

Under the guidance of Dr. Prasenjit Choudhury, our goal is to craft a system that's not just functional but exemplary—a testament to technical skill and teamwork. By meeting these objectives, we aim to serve up a solution that's as educational as it is innovative, ready to satisfy both academic rigor and real world demands!

Project Tasks :

Our project primarily involves the following tasks:

Database Design :

- Designing the database schema, which includes defining the tables, columns, data types, primary keys, and foreign keys.

- Creating an Entity Relationship Diagram (ERD) to visually represent the relationships between entities in the database.

SQL Implementation :

- Writing SQL statements to create the database and tables.
- Implementing SQL queries to perform various operations on the database, such as:

  • Inserting data into tables.

- Updating data in tables.
- Retrieving data from tables using SELECT statements.
- Using aggregate functions, grouping, filtering, and subqueries to analyze and report on data.

Reporting :

- Generating reports using SQL queries to provide insights into the food ordering system.

Project Overview

The "Online Food Ordering System" project focuses on designing and implementing the backend database for an online food ordering platform.

- **Key aspects of the project overview:**

  - The primary goal is to create a robust and efficient database system to manage food orders between customers and restaurants.
  - The system captures essential data related to customers, restaurants, food items, and orders.

  - The project emphasizes the use of SQL for all backend operations and analytics, without involving any frontend interfaces.
  - The project aims to demonstrate various SQL functionalities and their application in a real-world scenario.
  - The system is designed to be scalable and ready for future integration with a frontend or API.

Entity Relationship Diagram (ERD)

The ER diagram represents the logical structure of the

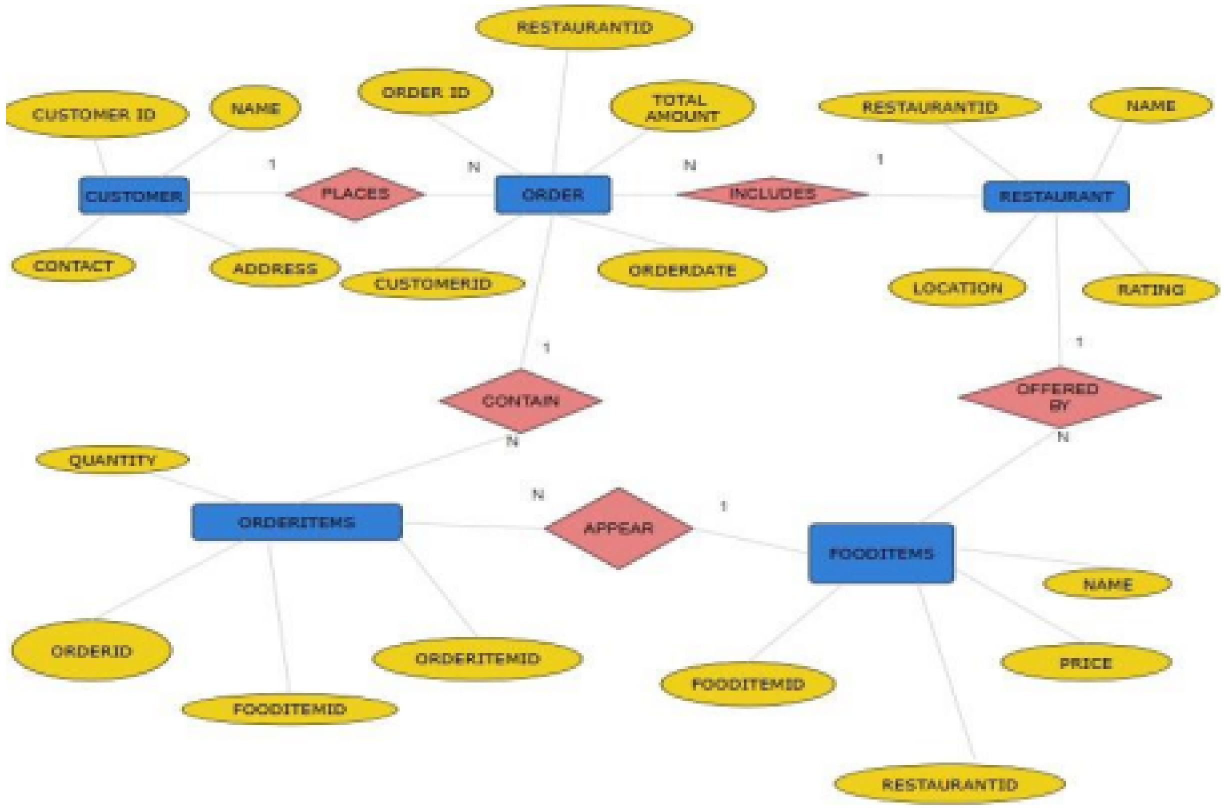
database. · **Entities:**

  - Customers
  - Restaurants

- FoodItems
- Orders
- OrderItems

· **Key Relationships:**

- A **Customer** can place multiple **Orders**.
- A **Restaurant** can receive many **Orders**
- An **Order** can contain multiple **OrderItems**
- A **Restaurant** can offer many **FoodItems**.
- A **FoodItems** can appear in many **OrderItems**



ER Diagram

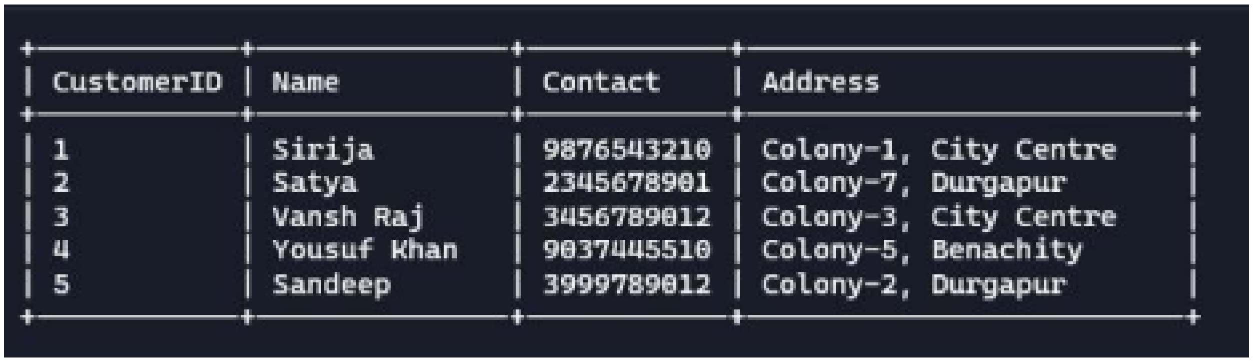
Table 1: Customers

Code :

```
CREATE TABLE Customers (
  CustomerID INT PRIMARY KEY,
  Name VARCHAR(100),
  Contact VARCHAR(20),
  Address VARCHAR(200)
);

INSERT INTO Customers VALUES
(1, 'Sirija', '9876543210', 'Colony-1, City Centre'),
(2, 'Satya', '2345678901', 'Colony-7, Durgapur'),
(3, 'Vansh Raj', '3456789012', 'Colony-3,City Centre'),
(4, 'Yousuf Khan', '9037445510', 'Colony-5,Benachity'),
(5, 'Sandeep', '3999789012', 'Colony-2,Durgapur');
```

Result :



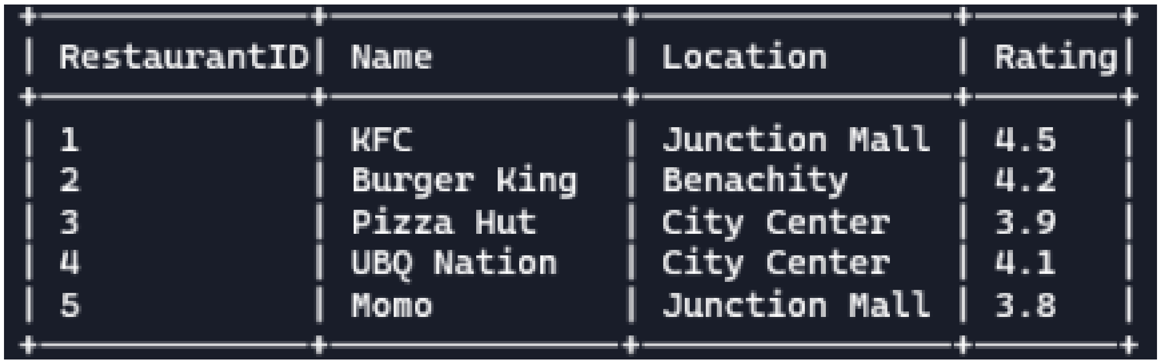
```
+------------+---------------+------------+-----------------------+
| CustomerID | Name          | Contact    | Address               |
+------------+---------------+------------+-----------------------+
| 1          | Sirija        | 9876543210 | Colony-1, City Centre |
| 2          | Satya         | 2345678901 | Colony-7, Durgapur    |
| 3          | Vansh Raj     | 3456789012 | Colony-3, City Centre |
| 4          | Yousuf Khan   | 9037445510 | Colony-5, Benachity   |
| 5          | Sandeep       | 3999789012 | Colony-2, Durgapur    |
+------------+---------------+------------+-----------------------+
```

Table 2: Restaurants

```
CREATE TABLE Restaurants (
 RestaurantID INT PRIMARY KEY,
 Name VARCHAR(100),
  Location VARCHAR(100),
  Rating FLOAT
);

INSERT INTO Restaurants VALUES
(1, 'KFC', 'Junction Mall', 4.5),
(2, 'Burger King', 'Benachity', 4.2),
(3, 'PizzaHut', 'City Center', 3.9),
(4, 'UBQNation', 'City Center', 4.1),
(5, 'Momo', 'Junction Mall', 3.8);
```

Result :



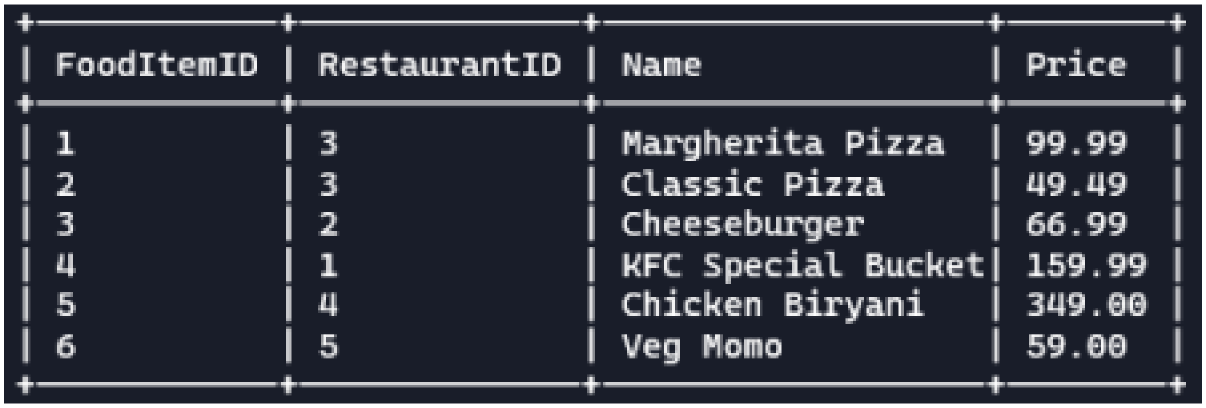
```
+-------------+------------+--------------+--------+
| RestaurantID| Name       | Location     | Rating|
+-------------+------------+--------------+--------+
| 1           | KFC        | Junction Mall | 4.5   |
| 2           | Burger King | Benachity    | 4.2   |
| 3           | Pizza Hut  | City Center  | 3.9   |
| 4           | UBQ Nation | City Center  | 4.1   |
| 5           | Momo       | Junction Mall | 3.8   |
+-------------+------------+--------------+--------+
```

Table 3 : FoodItems

```
CREATE TABLE FoodItems (
FoodItemID INT PRIMARY KEY,
RestaurantID INT,
Name VARCHAR(100),
Price DECIMAL(6,2),
FOREIGN KEY (RestaurantID) REFERENCES Restaurants(RestaurantID)
);
```

```
INSERT INTO FoodItems VALUES
(1, 3, 'Margherita Pizza', 99.99),
(2, 3, 'Classic Pizza', 49.49),
(3, 2, 'Cheeseburger', 66.99),
(4, 1, 'KFC Soecial Bucket', 159.99),
(5, 4, 'Chicken Biryani', 349.00),
(6, 5, 'Veg Momo', 59.00);
```

Result :



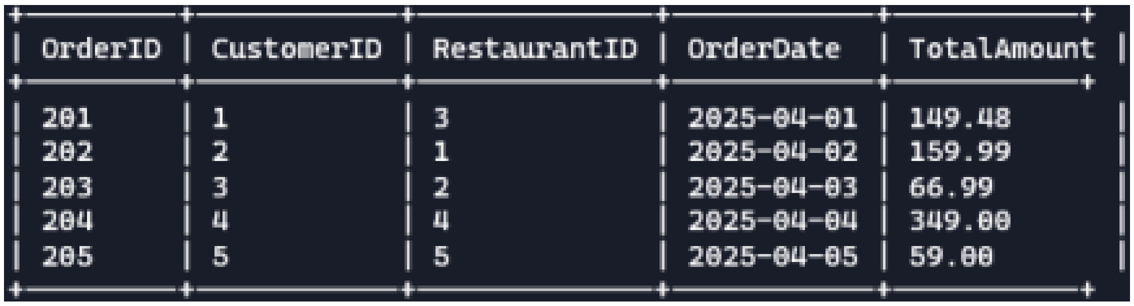
```
+------------+--------------+---------------------+--------+
| FoodItemID | RestaurantID | Name                | Price  |
+------------+--------------+---------------------+--------+
| 1          | 3            | Margherita Pizza    | 99.99  |
| 2          | 3            | Classic Pizza       | 49.49  |
| 3          | 2            | Cheeseburger        | 66.99  |
| 4          | 1            | KFC Special Bucket| 159.99 |
| 5          | 4            | Chicken Biryani     | 349.00 |
| 6          | 5            | Veg Momo            | 59.00  |
+------------+--------------+---------------------+--------+
```

Table 4 : Orders

```
CREATE TABLE Orders (
OrderID INT PRIMARY KEY,
CustomerID INT,
RestaurantID INT,
OrderDate DATE,
TotalAmountDECIMAL(7,2),
FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID),
FOREIGN KEY (RestaurantID) REFERENCES Restaurants(RestaurantID)
);

INSERT INTO Orders VALUES
(201, 1, 3, '2025-04-01', 149.48),
(202, 2, 1, '2025-04-02', 159.99),
(203, 3, 2, '2025-04-03', 66.99),
(204, 4, 4, '2025-04-04', 349.00),
(205, 5, 5, '2025-04-05', 59.00);
```

Results :



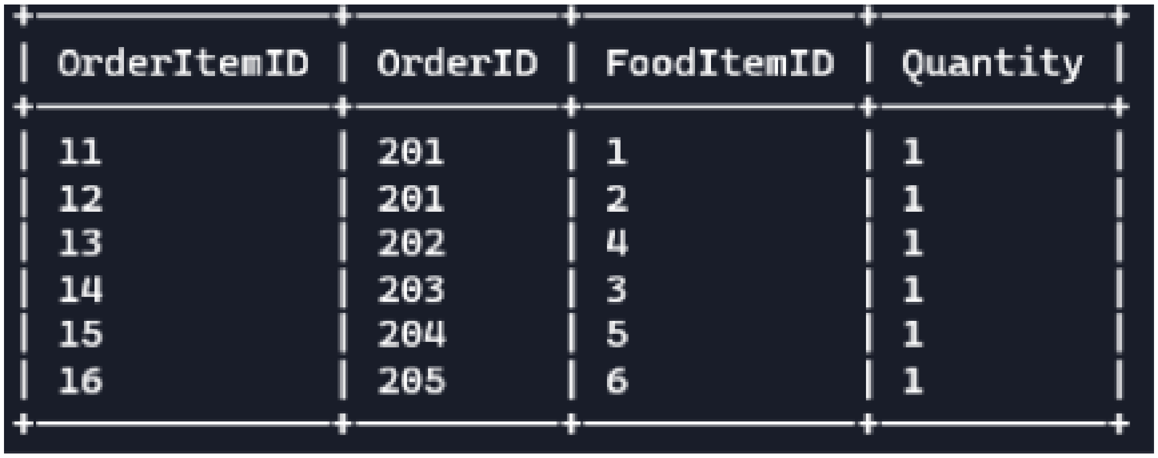
```
+---------+------------+--------------+------------+-------------+
| OrderID | CustomerID | RestaurantID | OrderDate  | TotalAmount |
+---------+------------+--------------+------------+-------------+
| 201     | 1          | 3            | 2025-04-01 | 149.48      |
| 202     | 2          | 1            | 2025-04-02 | 159.99      |
| 203     | 3          | 2            | 2025-04-03 | 66.99       |
| 204     | 4          | 4            | 2025-04-04 | 349.00      |
| 205     | 5          | 5            | 2025-04-05 | 59.00       |
+---------+------------+--------------+------------+-------------+
```

Table 5: OrderItems

```
CREATE TABLE OrderItems (
OrderItemID INT PRIMARY KEY,
OrderID INT,
FoodItemID INT,
Quantity INT,
FOREIGN KEY (OrderID) REFERENCES Orders(OrderID),
FOREIGN KEY (FoodItemID) REFERENCES FoodItems(FoodItemID)
);
```

```
INSERT INTO OrderItems VALUES
(11, 201, 1, 1),
(12, 201, 2, 1),
(13, 202, 4, 1),
(14, 203, 3, 1),
(15, 204, 5, 1),
(16, 205, 6, 1);
```

Result :



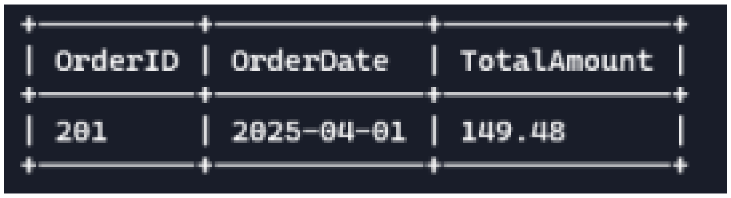
```
+-------------+---------+------------+----------+
| OrderItemID | OrderID | FoodItemID | Quantity |
+-------------+---------+------------+----------+
| 11          | 201     | 1          | 1        |
| 12          | 201     | 2          | 1        |
| 13          | 202     | 4          | 1        |
| 14          | 203     | 3          | 1        |
| 15          | 204     | 5          | 1        |
| 16          | 205     | 6          | 1        |
+-------------+---------+------------+----------+
```

Queries

Query 1: Orders by a Specific Customer

```
SELECT
  OrderID,
OrderDate,
TotalAmount
FROM Orders
WHERE
  CustomerID = 1;
```

Output :

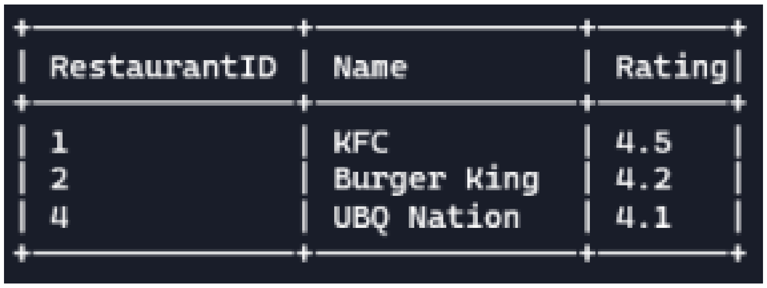


```
+---------+------------+-------------+
| OrderID | OrderDate  | TotalAmount |
+---------+------------+-------------+
| 201     | 2025-04-01 | 149.48      |
+---------+------------+-------------+
```

Query 2: Restaurants with Rating Above 4.0

```
SELECT
  RestaurantID,
  Name,
  Rating
FROM Restaurants
WHERE
  Rating > 4.0;
```

Output :

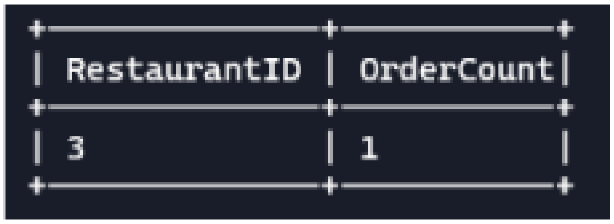


```
+--------------+-------------+-------+
| RestaurantID | Name        | Rating|
+--------------+-------------+-------+
| 1            | KFC         | 4.5   |
| 2            | Burger King | 4.2   |
| 4            | UBQ Nation  | 4.1   |
+--------------+-------------+-------+
```

Query 3: Most Popular Restaurant

```
SELECT
  RestaurantID,
  COUNT(*) AS OrderCount
FROM Orders
GROUP BY RestaurantID
ORDER BY OrderCount DESC
LIMIT 1;
```

Ouput :



```
+---------------+------------+
| RestaurantID  | OrderCount |
+---------------+------------+
| 3             | 1          |
+---------------+------------+
```

Query 4: Update Customer Address

```
UPDATE Customers
SET
  Address = 'New Colony, City Centre'
WHERE
  CustomerID = 1;
```

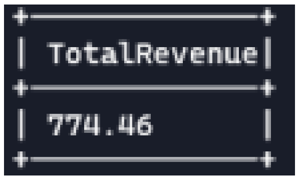
Output:

(**Note**: This query modifies data, it doesn't return a table in the same way as SELECT. To show the change, you'd typically re-select from the table)

Query 5: Total Revenue

```
SELECT
  SUM(TotalAmount) AS TotalRevenue
FROM Orders;
```

Output :

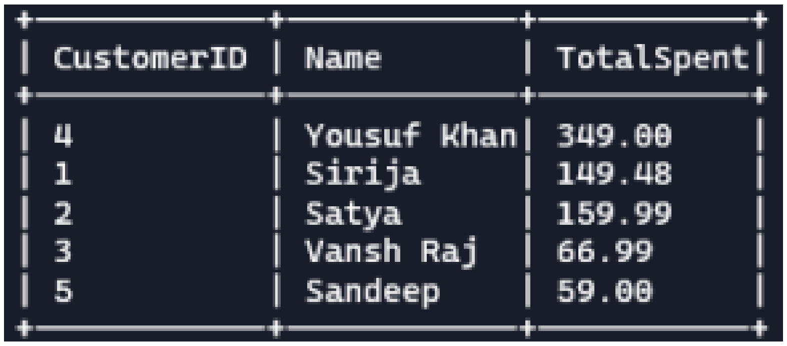


```
+--------------+
| TotalRevenue|
+--------------+
| 774.46      |
+--------------+
```

Query 6: Top 5 Highest Spending Customers

```
SELECT
  c.CustomerID,
  c.Name,
  SUM(o.TotalAmount) AS TotalSpent
FROM Customers c
JOIN Orders o ON c.CustomerID = o.CustomerID
GROUP BY c.CustomerID, c.Name
ORDER BY TotalSpent DESC
LIMIT 5;
```

Output :



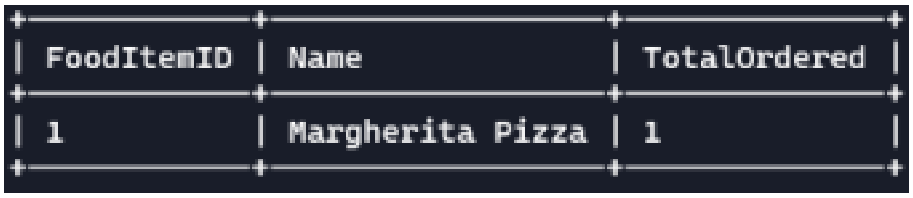
```
+------------+-------------+------------+
| CustomerID | Name        | TotalSpent|
+------------+-------------+------------+
| 4          | Yousuf Khan| 349.00     |
| 1          | Sirija      | 149.48     |
| 2          | Satya       | 159.99     |
| 3          | Vansh Raj   | 66.99      |
| 5          | Sandeep     | 59.00      |
+------------+-------------+------------+
```

Query 7: Most Ordered Food Item

```
SELECT
  f.FoodItemID,
  f.Name,
  SUM(oi.Quantity) AS TotalOrdered
FROM FoodItems f
JOIN OrderItems oi ON f.FoodItemID = oi.FoodItemID
GROUP BY f.FoodItemID, f.Name
ORDER BY TotalOrdered DESC
```

LIMIT 1;

Output :

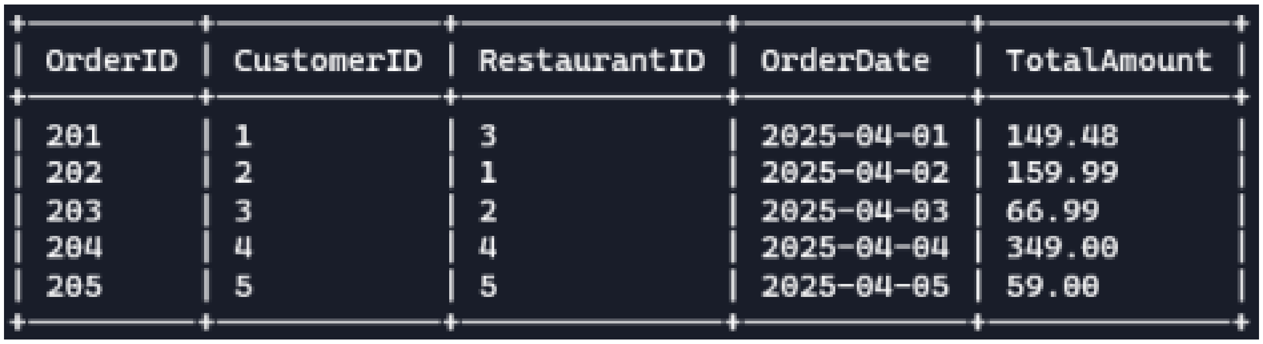


```
+------------+-------------------+--------------+
| FoodItemID | Name              | TotalOrdered |
+------------+-------------------+--------------+
| 1          | Margherita Pizza | 1            |
+------------+-------------------+--------------+
```

Query 8: Orders Placed in the Last Week

```
SELECT
  * FROM Orders
WHERE
  OrderDate BETWEEN CURDATE() - INTERVAL 7 DAY AND CURDATE();
```

Output :

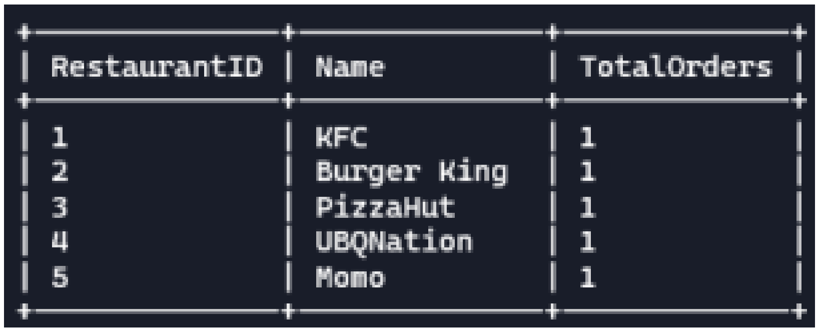


```
+---------+------------+--------------+------------+-------------+
| OrderID | CustomerID | RestaurantID | OrderDate  | TotalAmount |
+---------+------------+--------------+------------+-------------+
| 201     | 1          | 3            | 2025-04-01 | 149.48      |
| 202     | 2          | 1            | 2025-04-02 | 159.99      |
| 203     | 3          | 2            | 2025-04-03 | 66.99       |
| 204     | 4          | 4            | 2025-04-04 | 349.00      |
| 205     | 5          | 5            | 2025-04-05 | 59.00       |
+---------+------------+--------------+------------+-------------+
```

Query 9: Total Orders by Restaurant

```
SELECT
  r.RestaurantID,
  r.Name,
  COUNT(o.OrderID) AS TotalOrders
FROM Restaurants r
LEFT JOIN Orders o ON r.RestaurantID = o.RestaurantID
GROUP BY r.RestaurantID, r.Name;
```

Output :



```
+--------------+-------------+-------------+
| RestaurantID | Name        | TotalOrders |
+--------------+-------------+-------------+
| 1            | KFC         | 1           |
| 2            | Burger King | 1           |
| 3            | PizzaHut    | 1           |
| 4            | UBQNation   | 1           |
| 5            | Momo        | 1           |
+--------------+-------------+-------------+
```

Query 10: Restaurants with Highest Customer Retention Rate

```
SELECT
  r.RestaurantID,
  r.Name,
  COUNT(DISTINCT o.CustomerID) AS RepeatedCustomers
FROM Orders o
JOIN Restaurants r ON o.RestaurantID = r.RestaurantID
WHERE o.CustomerID IN (SELECT CustomerID FROM Orders GROUP BY CustomerID
HAVING COUNT(OrderID) > 1)
GROUP BY r.RestaurantID, r.Name
ORDER BY RepeatedCustomers DESC;
```

Output :

Conclusion:

This project successfully delivered a relational database system for online food ordering. The database is designed to manage key entities like customers, restaurants, food items, and orders, using SQL for all operations. Key SQL functions such as joins, aggregation, grouping, filtering, and subqueries were implemented. The system is designed to be scalable and can be integrated with front-end interfaces or REST APIs.

Real-life Impact:

Such a system directly addresses the inefficiencies in traditional food ordering systems. By digitizing and centralizing data, it can reduce errors, speed up order processing, and improve delivery times. It also provides restaurants with valuable data-driven insights, enabling better understanding of customer preferences and streamlining business operations.

Future Advancements:

The database provides a solid foundation. Future advancements would likely focus on building a user-friendly front-end, integrating real-time order tracking, implementing automated delivery management systems, and incorporating AI for personalized recommendations and demand forecasting.

Bibliography

- **Elmasri, R., & Navathe, S. B. (2017).** ***Fundamentals of Database Systems*** **(7th ed.). Pearson Education.**

- **Silberschatz, A., Korth, H. F., & Sudarshan, S.** (2019). *Database System Concepts* (7th ed.). McGraw-Hill Education.

- **GeeksforGeeks** (n.d.). *Database Management Systems (DBMS) Guide*. Retrieved April 6, 2025, from https://www.geeksforgeeks.org/dbms/

- **Oracle Corporation** (2024). *MySQL Documentation*. Retrieved April 6, 2025, from https://dev.mysql.com/doc/

- **W3Schools (n.d.).** ***SQL Tutorial*****. Retrieved April 6, 2025, from https://www.w3schools.com/sql/**

- **Choudhury, P.** (2025). *Lecture Notes on Database Management Systems*. National Institute of Technology, Durgapur