

**UNIVERSITY of ASIA PACIFIC**

***Department of Computer Science & Engineering (CSE)***

**Course Title:** Database Systems Lab

**Course Code:**  CSE - 212

**Project Name:** Restaurant Management System

|  |  |  |
| --- | --- | --- |
| **Submitted By-** | | **Submitted To-** |
| **Name** | : Sharmin Sultana Shila,  Md. Junaid Al Mahir,  Uday Majumder | **Alif Ruslan**  Lecturer, Dept. of CSE, University of Asia Pacific |
| **Registration** | : 23101070,  23101058,  23101082 |
| **Semester** | : 2nd Year 2nd Semester |
| **Section** | : B1 |

***Table Of Content***

**1. Project Description 3**

**2. Entities (Tables) 3**

**3. Table of Work 4**

1. **Relationship Between Entities 7**

**i. Customer 7**

**ii. Table\_Number 7**

**iii. Reserve\_Table 7**

**iv. Item\_List 7**

**v. Food 7**

**vi. Item\_Price 8**

**vii. Is\_Exchangble 8**

**viii. Customer\_Order 8**

**ix.Payment 8**

**x.Waiters 8**

**5. ER- Diagram 9**

**6. Schema- Diagram 10**

1. **SQL Queries 11**

**i. Arithmatic Operation based 11**

**ii. Conditional Operation based 13**

**iii. Table Modification 16**

**iv. Order\_By & Group\_By 18**

**v. Join Operaton 19**

**8. CEP Mapping 22**

**8. Conclusion 27**

**1.Project Description**

The Restaurant Management System is a relational database project developed to efficiently manage the operations of a restaurant. Restaurants typically deal with a large volume of dynamic data including customer reservations, food menu management, order tracking, payment processing, and staff assignments. Managing this data manually or through flat-file systems often results in inefficiencies, errors, and delays in service.

This project leverages MySQL to implement a normalized relational database that supports these core operations. The system includes multiple interconnected entities such as Customer, Table, Food, Order, Payment, and Waiter. Each table has been carefully designed using primary and foreign key constraints to maintain referential integrity and ensure accurate data relationships.

The primary goal is to offer a comprehensive backend data solution that can:

1. Record and track customer details and reservations.
2. Maintain an up-to-date menu with flexible ingredient substitutions.
3. Process customer orders and calculate total bills.
4. Provide reports on popular dishes, high-spending customers and table availability.
5. Streamline waiter assignments and staff salary analysis.

By implementing a robust SQL-based backend, this system allows restaurant managers to efficiently manage day-to-day operations and make informed decisions using real-time data insights. The database is scalable and can be integrated with a web or mobile frontend to build a complete restaurant management software in the future.

**2.Entities(Tables)**

1. **Customer:** (Customer\_ID, Customer\_Name, City, Contact\_No)
2. **Table\_Number:** (Table\_ID, Capacity)
3. **Reserve\_Table:(** Reserve\_ID, Table\_ID (FK), Customer\_ID (FK), Is\_Available)
4. **Item\_List:** (Item\_ID, Item\_Type)
5. **Food:** (Food\_ID, Food\_Name, Item\_ID(FK), Food\_Size, Food\_Quantity, Price)
6. **Item\_Price:** ( Exchangable\_Item, Price )
7. **IsExchangble:** ( Food\_ID(FK), Exchangable\_Item(FK), Quantity\_Of\_Item )
8. **Customer\_Order: (** Order\_ID, Customer\_ID(FK), Item\_ID(FK), Food\_ID(FK), Quantity **)**
9. **Payment: (**Payment\_ID**,** Order\_ID(FK)**,** Method, Total\_Bill **)**
10. **Waiters : (** Waiter\_ID (FK), Waiter\_Name, Age, Salary  **)**

**3.Table of Work**

**1. Customer :**

* 1. **Customer\_ID (Primary Key)**
  2. **Customer\_Name (Not Null)**
  3. **City**
  4. **Contact\_No**

This table is created to store information about customers who gives order.

1. **Table\_Number:**
   1. **Table\_ID (Primary Key)**
   2. **Capacity**

This table is designed to keep track each tables with their capacity.

1. **Reserve\_Table:**
   1. **Reserve\_ID (Primary Key)**
   2. **Table\_ID (Foreign Key)**
   3. **Customer\_ID (Foreign Key)**
   4. **Is\_Available**

This table is designed to keep track which tables are reserved by a customer. Here, Table\_ID and Customer\_ID both are foreign key referred from Table\_Number and Customer table respectively.

1. **Item\_List:** 
   1. **Item\_ID (Primary Key)**
   2. **Item\_Type**

This table describes the menu for the restaurent, their number and corresponding type.

1. **Food:** 
   1. **Food\_ID (Primary Key)**
   2. **Food\_Name (Primary Key)**
   3. **Item\_ID (Foreign Key)**
   4. **Food\_Size**
   5. **Food\_Quantity**
   6. **Price**

This table illustrates the meals in each segment of item, their size, quantity and corresponding price. Here, Item\_ID is the foreign key which is referred from Item\_List table.

1. **Item\_Price:**
   1. **Exchangable\_Item (Primary Key)**
   2. **Price**

This table distinguishes the price of each items which are used in foods.

1. **IsExchangble:**
   1. **Food\_ID (Foreign Key)**
   2. **Exchangable\_Item (Foreign Key)**
   3. **Quantity\_Of\_Item**

This table illustrates that a food can be replaced by a particular or many items. The quantity of that item and its related info’s are described in this table. Here, Food\_ID and Exchangable\_Item both are foreign key referred from Food and Item\_Price table respectively.

1. **Customer\_Order:**
   1. **Order\_ID (Primary Key)**
   2. **Customer\_ID (Foreign Key)**
   3. **Item\_ID (Foreign Key)**
   4. **Food\_ID (Foreign Key)**
   5. **Quantity**

This table shows the order details placed by customer, their ordered meal and quantity. Here, Food\_ID, Item\_ID and Customer\_ID both are foreign key referred from Food , Item\_List and Customer table respectively.

1. **Payment:** 
   1. **Payment\_ID (Primary Key)**
   2. **Order\_Id (Foreign Key)**
   3. **Method**
   4. **Total\_Bill**

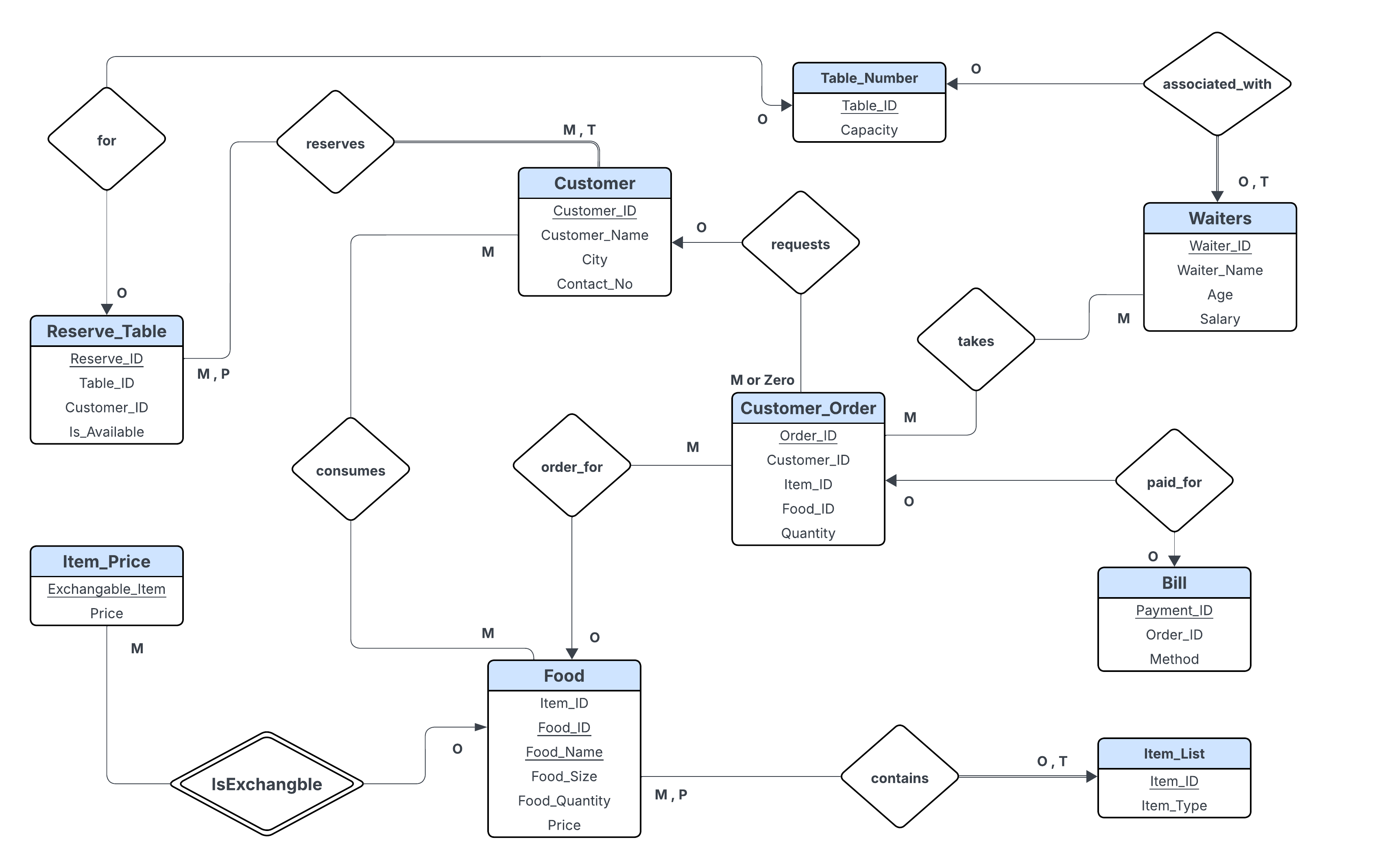
This table tells the payment method according to customer order.

1. **Waiters:**
   1. **Waiter\_ID (Primary Key) (Foreign Key)**
   2. **Waiter\_Name**
   3. **Age**
   4. **Salary**

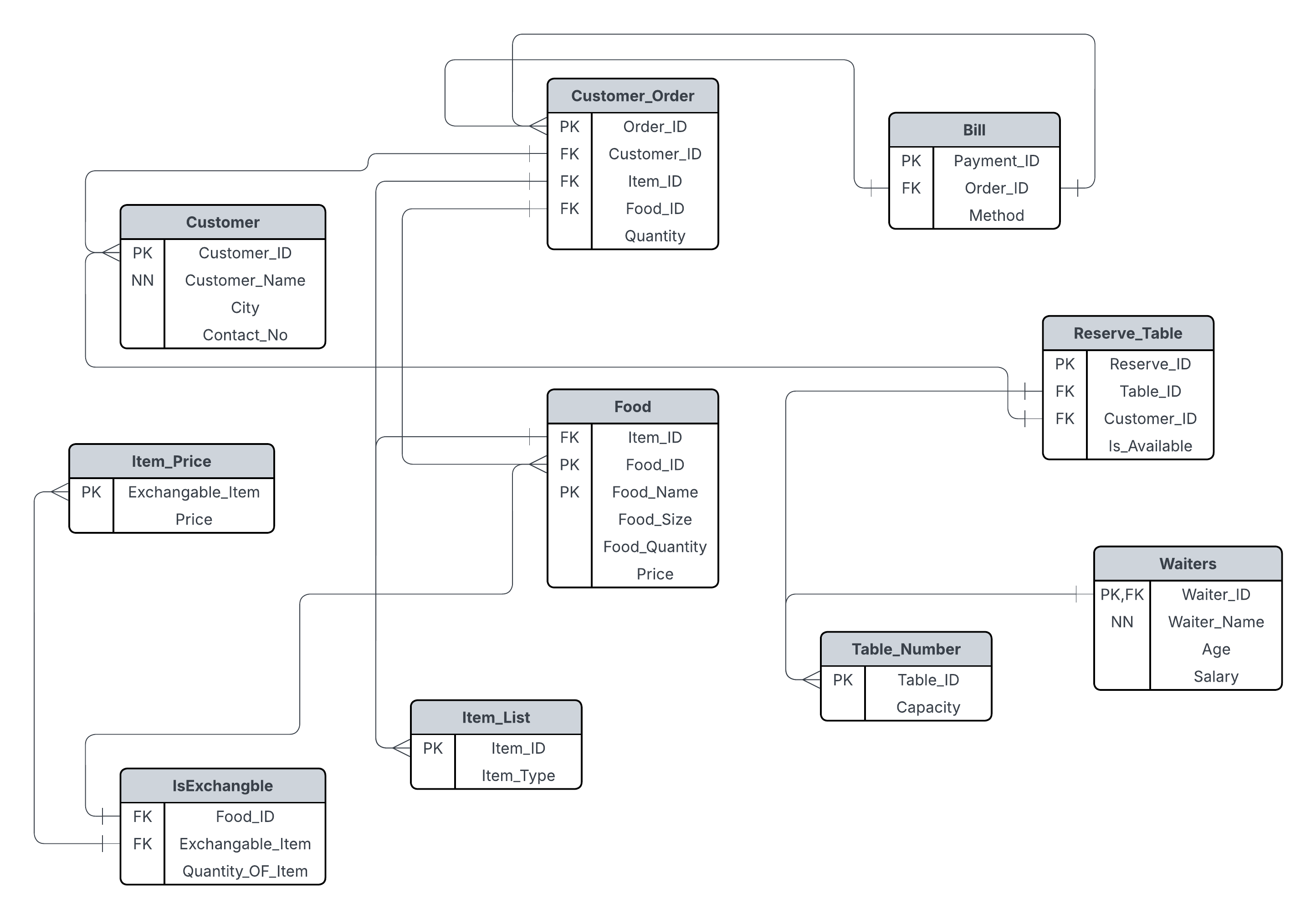
This table holds the details of a waiter. Here, Waiter\_ID is both Primary and Foreign key references from Table\_Number.

1. **Relationship Between Entities**
2. **Customer ↔ Reserve\_Table**  
   **Relationship:** One-to-Many  
   **Explanation:**  
   A single customer can reserve multiple tables over time. The Reserve\_Table entity links each reservation to a specific customer and table. This relationship enables the system to track which customers have reserved which tables and their availability status.
3. **Table\_Number ↔ Reserve\_Table**  
   **Relationship:** One-to-Many  
   **Explanation:**  
   Each table can be reserved multiple times. The Reserve\_Table holds the mapping between Table\_Number and reservation details, helping the restaurant manage table usage efficiently.
4. **Table\_Number ↔ Waiters**  
   **Relationship:** One-to-One  
   **Explanation:**  
   Each waiter is assigned to one specific table. The Waiter\_ID is also used as a foreign key referencing Table\_ID, creating a one-to-one relationship. This helps track which waiter is responsible for each table.
5. **Item\_List ↔ Food**  
   **Relationship:** One-to-Many  
   **Explanation:**  
   Every food item is categorized by its type (e.g., Starter, Main Course, etc.) through the Item\_List table. Each Item\_ID in Item\_List can be linked to multiple food items in the Food table.
6. **Food ↔ Customer\_Order**  
   **Relationship:** One-to-Many  
   **Explanation:**  
   A food item may appear in multiple customer orders. The Customer\_Order table captures all food items that customers have ordered, their quantities, and associated customer and item details.
7. **Customer ↔ Customer\_Order**  
   **Relationship:** One-to-Many  
   **Explanation:**  
   A customer can place multiple orders, each recorded in the Customer\_Order table. This allows the system to track complete order history per customer.
8. **Customer\_Order ↔ Payment**  
   **Relationship:** One-to-One  
   **Explanation:**  
   Every order placed by a customer has one associated payment record. The Payment table stores the payment method and the total amount, linked to the corresponding order via Order\_ID.
9. **Food ↔ IsExchangble**  
   **Relationship:** One-to-Many  
   **Explanation:**  
   Some food items can have their ingredients replaced. The IsExchangble table links food items to their optional replacement ingredients, allowing flexible dish customization.
10. **IsExchangble ↔ Item\_price**  
    **Relationship:** Many-to-One  
    **Explanation:**  
    Each exchangeable item is listed with a specific price in the Item\_price table. The relationship helps calculate new prices when ingredients in food items are substituted.

**5.ER Diagram**

****

**6. Schema Diagram**

****

1. **SQl Queries**

***# Simple:***

**1. Find the waiters details whose name starts with 'Don'.**

**~ SELECT \* FROM Waiters WHERE Waiter\_Name LIKE "Don%";**

A screenshot of a computer

AI-generated content may be incorrect.

**2. Find the food list of menu 11.**

**~ SELECT \* FROM Food WHERE  Item\_ID = 11;**

A screenshot of a computer

AI-generated content may be incorrect.

**3. Find the Waiter details whose age is between 20-23.**

**~ SELECT \* FROM Waiters WHERE Age BETWEEN 20 AND 23 ;**

A screenshot of a computer

AI-generated content may be incorrect.

**4. Find the cities that customer belongs from.**

**~ SELECT DISTINCT City AS City\_Of\_Customers FROM Customer LIMIT 5;**

A screenshot of a computer

AI-generated content may be incorrect.

**5. Select the total number of tables in that restaurent .**

**~ SELECT COUNT(Table\_ID) AS Number\_Of\_Tables FROM Table\_Number where Capacity >= 08;**

A screenshot of a computer

AI-generated content may be incorrect.

**6. Annual salary of the Waiters.**

**~ SELECT Waiter\_ID,Waiter\_Name,12\*Salary AS Annual\_Salary FROM Waiters ;**

A screenshot of a computer

AI-generated content may be incorrect.

***# Moderate:***

**7. Concatenate the list of food and name corresponding to their price.**

**~ SELECT CONCAT(Food\_ID, " - ",Food\_name, " = ", Price) AS Details\_Of\_Dish FROM Food;**

A screenshot of a menu

AI-generated content may be incorrect.

**8. Change the name of "Bill" Table.**

**~ ALTER TABLE Bill RENAME TO Payment;**

**~ SELECT \* FROM Payment;**

A screenshot of a computer

AI-generated content may be incorrect.

**9.Add a new column in existing table "Payment".**

**~ ALTER TABLE Payment ADD (Total\_Bill DOUBLE(6,2));**

**~ SELECT \* FROM Payment;**

A screenshot of a computer

AI-generated content may be incorrect.

**10. Make a reserved seat availabe for other customers.**

**~ UPDATE Reserve\_Table SET Is\_Available = "True" WHERE table\_ID = 05;**

**~ SELECT \* FROM Reserve\_Table;**

A screenshot of a computer

AI-generated content may be incorrect.

**11. Find the waiter details of table no 05.**

**~ SELECT \* FROM Waiters WHERE Waiter\_ID IN**

**(SELECT Table\_ID FROM Table\_Number WHERE Table\_ID = 05  );**

A screenshot of a computer

AI-generated content may be incorrect.

**12.  Find availabe seats in that Restaurant.**

**~ SELECT Table\_ID, Capacity FROM Table\_Number WHERE Table\_ID Not IN ( SELECT Table\_ID FROM Reserve\_Table );**

A screenshot of a computer

AI-generated content may be incorrect.

**13. Show the list of food names in descending order.**

**~ SELECT Food\_ID,Food\_Name,Price FROM Food Order By Food\_name DESC;**

A screenshot of a computer

AI-generated content may be incorrect.

**14.  Change the column name of Food Table.**

**~ ALTER TABLE Food CHANGE COLUMN Food\_Size Size VARCHAR(15) NOT NULL ;**

**~ SELECT \* FROM Food;**

A screenshot of a computer

AI-generated content may be incorrect.

***# Hard:***

**15. Count the number of customers from each city.**

**~ SELECT City, COUNT(Customer\_ID) As No\_Of\_Customers FROM Customer GROUP BY City;**

A screenshot of a computer

AI-generated content may be incorrect.

**16.  Show the list of foods that are exchangable by other items.**

**~ SELECT F.Food\_ID,F.Food\_Name, E.Exchangable\_Item FROM Food as F, IsExchangble as E WHERE F.Food\_ID = E.Food\_ID ;**

A screenshot of a computer

AI-generated content may be incorrect.

**17. Fill up the Total\_Bill column of Payment table according to the customers order.**

**~ UPDATE Payment P**

**JOIN Customer\_Order CO ON P.Order\_ID = CO.Order\_ID**

**JOIN Food F ON CO.Food\_ID = F.Food\_ID**

**SET P.Total\_Bill = CO.Quantity \* F.Price;**

**SELECT \* FROM Payment;**

A screenshot of a computer

AI-generated content may be incorrect.

**18. Replace vegetable from "Veg Biryani" by Mutton.**

**~ SELECT**

**F.Food\_Name,**

**F.Price AS Original\_Price,**

**IP1.Price AS Old\_Ingredient\_Price,**

**IP2.Price AS New\_Ingredient\_Price,**

**(F.Price - IP1.Price + IP2.Price) AS Total\_Price**

**FROM Food AS F**

**JOIN IsExchangble E1 ON F.Food\_ID = E1.Food\_ID AND E1.Exchangable\_Item = 'Vegetable'**

**JOIN IsExchangble E2 ON F.Food\_ID = E2.Food\_ID AND E2.Exchangable\_Item = 'Mutton'**

**JOIN Item\_Price IP1 ON E1.Exchangable\_Item = IP1.Exchangable\_Item**

**JOIN Item\_Price IP2 ON E2.Exchangable\_Item = IP2.Exchangable\_Item**

**WHERE F.Food\_Name = 'Veg Biryani';**

A screenshot of a computer

AI-generated content may be incorrect.

**19. count the number of customers who purchased more than 1000 BDT in each city in ascending order.**

**~ SELECT C.Customer\_ID, C.Customer\_Name, C.City, SUM(O.Quantity \* F.Price) AS Total\_Spent**

**FROM Customer C**

**JOIN Customer\_Order O ON C.Customer\_ID = O.Customer\_ID**

**JOIN Food F ON O.Food\_ID = F.Food\_ID**

**GROUP BY C.Customer\_ID, C.Customer\_Name, C.City HAVING SUM(O.Quantity \* F.Price) > 1000;**

A screenshot of a computer

AI-generated content may be incorrect.

**20. Show the top 5 most famous selling item in this restaurant.**

**~ SELECT F.Item\_ID, F.Food\_ID, F.Food\_Name, SUM(O.Quantity) AS Total\_Quantity\_Sold**

**FROM Food F**

**JOIN Customer\_Order O ON F.Food\_ID = O.Food\_ID**

**GROUP BY F.Food\_ID, F.Food\_Name ORDER BY Total\_Quantity\_Sold DESC LIMIT 5;**

A screenshot of a computer

AI-generated content may be incorrect.

**8.CEP Mapping:**

**1. Knowledge Profiles(K’s) are addressed through the project and mapping among K’s, CO’s, and PO’s:-**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **K’s** | **Attributes** | **How K’s are Addressed in this project** | **CO** | **PO** |
| K2 | Mathematics | Conceptual mathematics and numerical analysis are used in calculating number of Items, availability of tables, all kinds of foods etc. | CO2,  CO3,  CO4 | PO1,  PO3,  PO6,  PO7,  PO8 |
| K3 | Engineering Fundamentals | Fundamental database principles are applied to design and operate the restaurant database, including formulation of relationships, normalization and integrity constraints to ensure proper data management. | CO1,  CO2,  CO3 | PO1 |
| K5 | Engineering Design | The project involves designing a structured ER diagram and schema diagram, effectively representing the data flow between entities like orders, customers, payments and reservations. | CO3,  CO4 | PO3,  PO5 |
| K6 | Engineering Practice | We’ve implemented our design in MySQL. We created tables, inserted data with DDL and ran queries with DML. We’ve used MySQL workbench software as a tool for implementing our design. |  |  |

**2. Complex Engineering Problem solving (P’s) are addressed through the project and mapping among K’s, CO’s, and PO’s:-**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **P’s** | **Attributes** | **How P’s are addressed through this project** | **CO’s** | **PO’s** |
| P1 | Depth of knowledge required | Our project requires us to study one of the fundamentals of engineering which is DBMS design (K3). It also requires designing ER & Schema Diagram(K5) and implementing the designs (K6) | CO3,  CO4,  CO5 | PO1,  PO2,  PO3,  PO7 |
| P3 | Depth of analysis required | Functional queries and schema optimization are developed using abstract thinking and in-depth analysis to handle customer. | CO3,  CO5 | PO3 |
| P6 | Extent of stake | The system is designed to accommodate the unique data requirements of stakeholders, such as customers, servers and kitchen staff members, maintaining smooth cooperation and satisfaction. | CO3,  CO5 | PO2,  PO7 |
| P7 | Interdependence | High levels of interdependence are displayed by components like IsExchangble, ItemPrice, OrderItem, TableNo, CustomerID; these interations guarantee proper data connection and reporting | CO3,  CO4,  CO5 | PO2,  PO7 |

**3. Complex Engineering Activity(A’s) are addressed through the project and mapping among K’s, CO’s, and PO’s:-**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A’s** | **Attributes** | **How A’s are addressed through our project** | **CO’s** | **PO’s** |
| A1 | Range of resources | The Project utilizes resources such as customer data, payment information, food orders and staff records to manage restaurant operations effectively. | CO1,  CO2,  CO8 | PO1,  PO3,  PO5,  PO6,  PO7,  PO8 |
| A2 | Level of interaction | High interaction between stakeholders Is modeled using relationship like Customer can replace an Item by IsExchangible entity, Waiter taking orders etc. | CO7,  CO8,  CO9 | PO1,  PO3,  PO5,  PO6 |
| A3 | Innovation | Innovations include query optimization and database triggers to automate order updates, managing table reservation and payment process, ensuring real time accuracy. | CO3,  CO5,  CO6,  CO7 | PO3,  PO6,  PO7,  PO8 |
| A5 | Familiarity | The project builds on previously established database concepts such as relational models and normalization and applies them to specific context of restaurant management. It addresses real-world challenges like minimizing data redundancy, resolving reservation conflicts and ensuring accurate financial record-keeping, showcasing familiarity with the practical application of database systems. | CO5,  CO6,  CO7 | PO10,  PO11,  PO12 |

**9.Conclusion:**

The Restaurant Management System developed in this project effectively demonstrates how relational databases can be used to model, manage, and optimize real-world restaurant operations. Through the use of structured SQL tables, well-defined relationships, and meaningful queries, the system supports essential functionalities such as handling customer data, managing reservations, processing orders, tracking inventory, managing payments, and organizing staff information.

#### ****Advantages:****

1. **Efficient Data Handling:** All data is organized in a structured format with well-defined relationships, making it easier to retrieve and update information.
2. **Customizable Orders:** The system supports ingredient substitutions using the IsExchangble table, allowing flexibility for customer preferences.
3. **Data Accuracy & Integrity:** Use of primary and foreign keys ensures no invalid or orphaned records exist in the database.
4. **Insightful Analysis:** SQL queries enable analysis of top-selling dishes, high-spending customers, and table availability.

#### ****Limitations:****

1. **No User Interface:** The project is purely backend; it lacks a GUI or frontend for practical usage by staff or customers.
2. **Limited Date/Time Handling:** Reservations and orders do not include timestamps, which are essential for real-world scheduling and tracking.
3. **Fixed Waiter Assignment:** Waiters are linked to only one table; in reality, a waiter may serve multiple tables or rotate shifts.
4. **No Authentication or Roles:** There’s no role-based access or login system for customers, admins, or waiters.

[**Drive Link:**](https://docs.google.com/document/d/1cwOOPbzpnjcuC0CTO0fXOW83co_rQQDu/edit?usp=drive_link&ouid=108986274194644082135&rtpof=true&sd=true)