DATABASE AND MANAGEMENT SYSTEM (ICCSC09)

PROJECT WORK

CLOUD KITCHEN MANAGEMENT SYSTEM

SUBMITTED TO-MR. ANAND GUPTA

ICE-2

GROUP MEMBERS:	ROLL NO
RHEA SONI	2022UIC3601
KUNAL KASHYAP	2022UIC3602
RITIK JAIN	2022UIC3603
GIRISH PORWAL	2022UIC3604

PROBLEM STATEMENT

Cloud Kitchen Management System

- **Inadequate Customer Information Management**:

The current system lacks a comprehensive database to efficiently capture and manage customer details, leading to potential errors and delays in order processing.

- **Manual Delivery Personnel Assignment:**

Without automated tools for assigning delivery personnel and optimizing delivery routes, there are inefficiencies in coordinating deliveries, resulting in delays and customer dissatisfaction.

- **Limited Menu Planning Support:**

Managing a diverse menu with various food items and sizes without proper linkage to chefs for preparation can lead to challenges in menu planning and timely order fulfillment.

- **Manual Payment Management:**

Challenges in monitoring payment statuses and processing payments manually can result in discrepancies, late payments, and potential revenue loss for the cloud kitchen.

- **Untapped Customer Insights:**

Inability to analyze customer order history and preferences due to limited data tracking inhibits the ability to tailor offerings and marketing strategies, missing out on opportunities to enhance customer experience and loyalty.

- **Underutilized Data Analytics Potential**:

Despite data availability, the current system lacks robust analytics capabilities to extract actionable insights, limiting opportunities for optimizing operations and decision-making.

.Without insights from data analytics, resources such as chefs and delivery personnel may not be allocated optimally, resulting in increased operational costs and decreased profitability.

OBJECTIVE

The objective of this project is to develop a comprehensive CKMS that automates and optimizes various aspects of cloud kitchen management, ultimately improving customer satisfaction and business profitability.

Features of CKMS

- **Customer Management:**

Capture customer details such as name, email, and mobile number.Maintain a database of customer orders and order history.

- **Delivery Coordination:**

Assign delivery boys to orders and track their departure and arrival times.Optimize delivery routes to ensure timely and efficient deliveries.

- **Chef Scheduling:**

Manage chef details including name, salary, and contact information.Schedule chefs for food preparation based on demand and availability.

- **Food Menu Planning:**

Maintain a dynamic menu with various food items, sizes, and prices.Link food items to respective chefs for efficient preparation.

- **Order Management:**

Generate unique order IDs and ensure the smooth delivery.Monitor payment status, customer reviews, and delivery status in real-time.

- **Proposed Solution:**

The CKMS will be implemented using a relational database management system (RDBMS) to store and manage data efficiently. The system will feature a user-friendly interface accessible to administrators, chefs, delivery personnel, and customers. Key functionalities will include order management, analytics dashboards, and reporting tools.

REQUIREMENTS ANALYSIS

Performance:

•Ensure fast response times for order processing, menu updates, and payment transactions.
•Handle concurrent user requests efficiently to prevent system slowdowns or downtime during peak hours.

Scalability:

•Design the system to accommodate future growth in terms of users, orders, and menu items.
•Implement scalable infrastructure and database solutions to handle increased load without performance degradation.

Reliability:

•Minimize system failures and errors to ensure continuous operation.
•Implement backup and recovery mechanisms to protect against data loss and system downtime.

Usability:

•Design an intuitive user interface for customers, chefs, and delivery boys to facilitate easy navigation and task completion.
•Provide clear instructions and feedback messages to guide users through the ordering and delivery process.

Accessibility:

Ensure that the system is accessible to users with disabilities, adhering to accessblity standards & guidelines.

By addressing these requirements, the cloud kitchen management system can efficiently manage orders, streamline operations, and deliver a seamless experience for customers, chefs, and delivery personnel.

ER MODEL

The Entity-Relationship (ER) model in Database Management Systems (DBMS) is a conceptual data model used to represent the structure of a database in a visual manner. Here's a short note on the ER model:
The Entity-Relationship (ER) model is a graphical representation used to design and conceptualize databases. It allows database designers to identify and define the entities, attributes, and relationships within a system.

•**Entities:** Entities represent real-world objects, such as people, places, or things, about which data needs to be stored in the database. Entities are depicted as rectangles in an ER diagram.

•**Attributes:** Attributes are properties or characteristics of entities that describe the data being stored. They are depicted as ovals connected to their respective entities in an ER diagram.

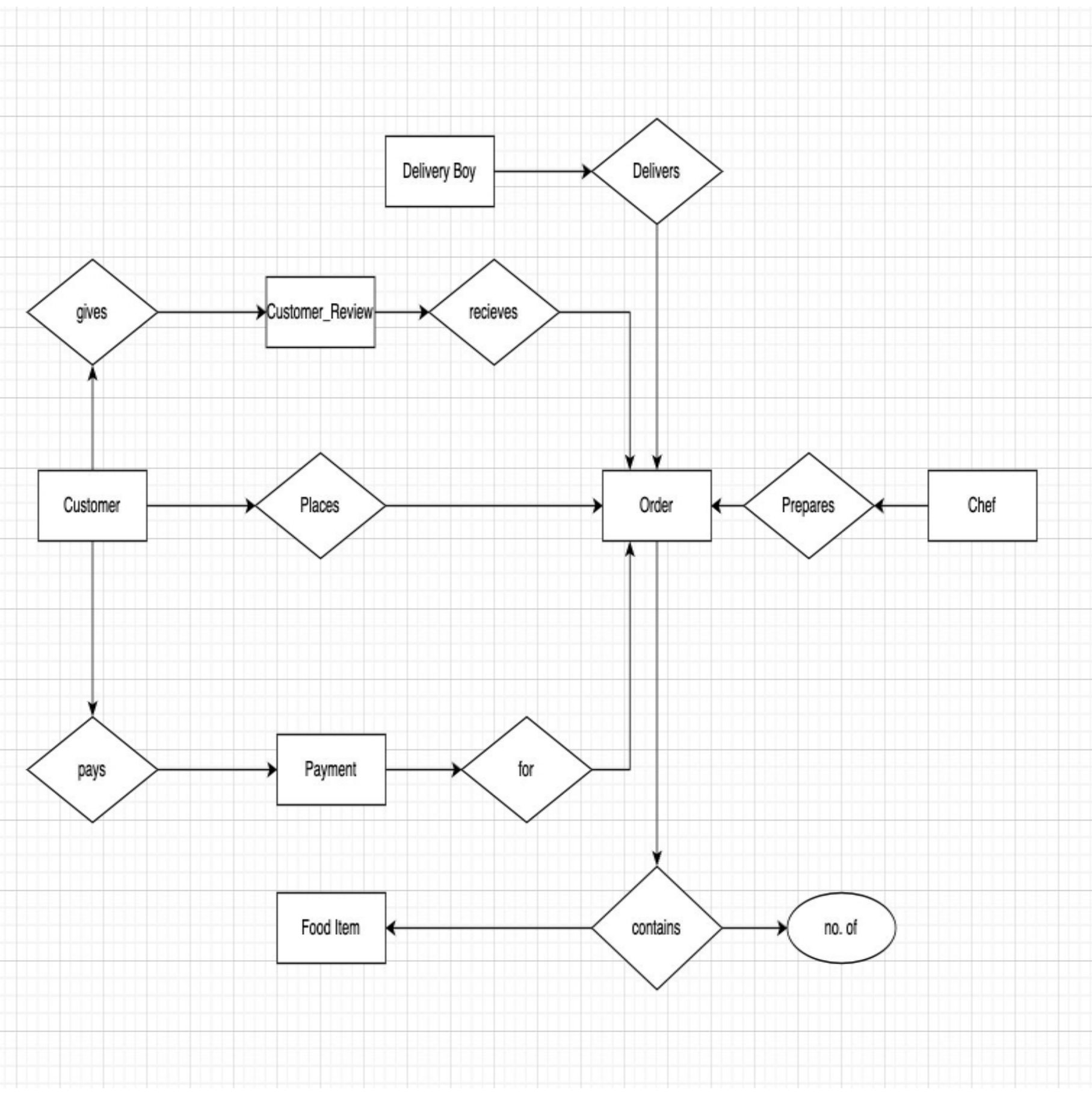
•**Relationships:** Relationships describe how entities are connected or related to each other within the database. They represent associations or interactions between entities and are depicted as lines connecting the participating entities in an ER diagram.
•**Cardinality:** Cardinality defines the number of instances of one entity that can be associated with a single instance of another entity through a relationship. It describes the relationship's multiplicity or how many entities are involved on each side of the relationship.

•**Normalization:** ER modeling helps in identifying and eliminating data redundancy and anomalies by applying normalization techniques. It ensures that the database design is optimized for efficient data storage and retrieval.

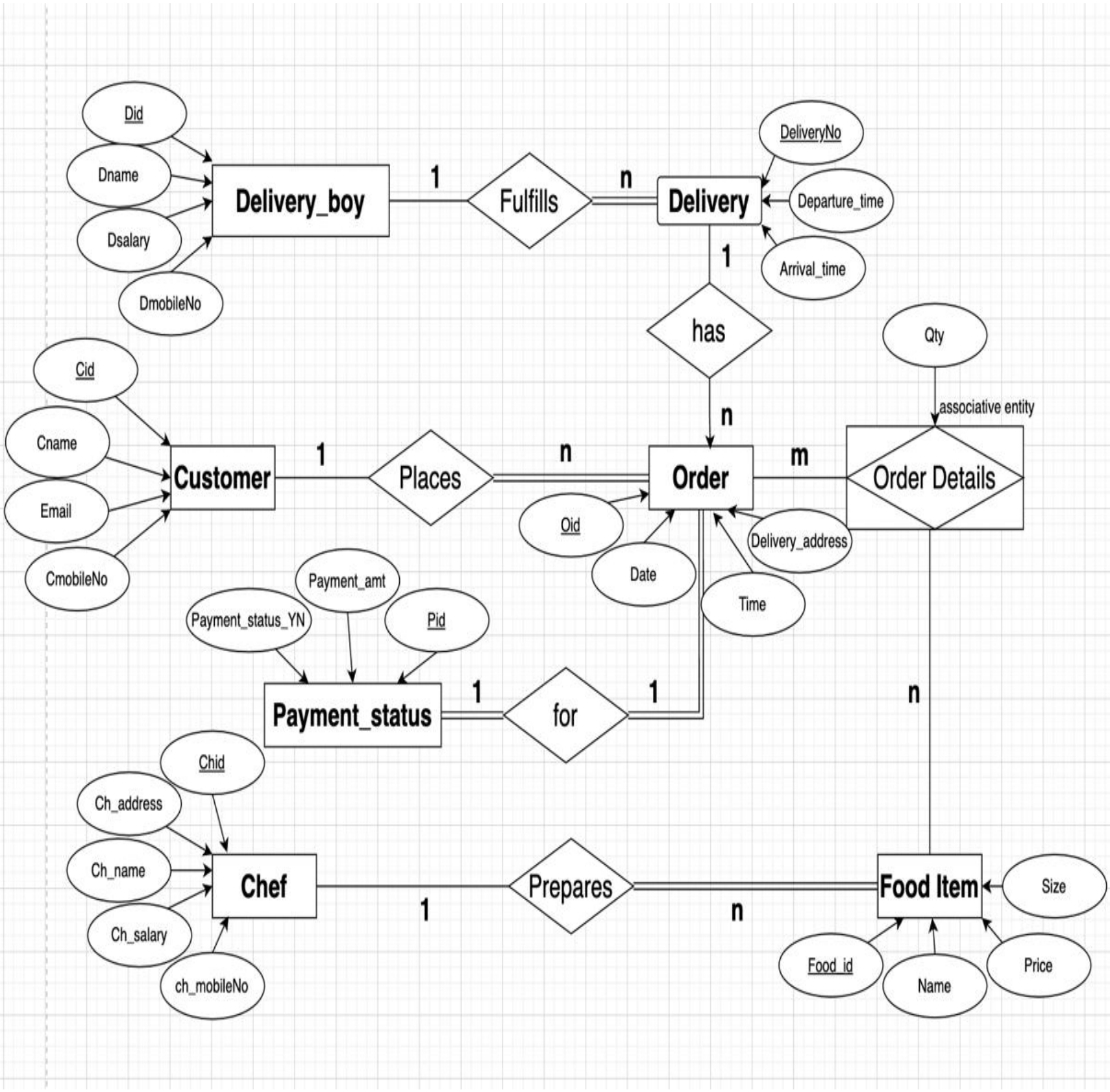
•**Denormalization:** In some cases, ER modeling may also involve denormalization, where redundancy is intentionally introduced to improve performance or simplify queries in a database system.

The ER model provides a visual representation of the database structure, making it easier for stakeholders to understand and communicate complex data relationships. It serves as a blueprint for creating the actual database schema and forms the basis for database implementation and maintenance.

BRAINSTORMING ER MODEL



FINAL ER MODEL



ER DIAGRAM TO RELATIONAL SCHEMA

1. Customer Table (C)

- cid
- cName
- email
- cMobileNo

2. Chef Table (Ch)

- **c**hid,
- chName
- chSalary
- chMobileNo

3. Food Item Chef Table (F)

- foodId
- fName
- price
- Size
- chid

4. Order Table (O)

- oid
- cid (foreign key references C.cid)
- date_time
- cust_review
- pid
- deliveryNo
- paymentAmout
- paymentStatus_YN

5. Delivery_boy Table (Delboy)

- dName
- did
- dSalary
- dMobileNo

6. Delivery Table (D)

deliveryNo (foreign key references O.deliveryNo)
did (foreign key references delBoy.did)
departureTime
arrivalTime

7. Order Details Table (OD)

oid(foreign key reference O.oid)
foodId
amount

FUNCTIONAL DEPENDENCIES

1. **customer Table**

   (C) [cid, cName, email, cMobileNo, Delivery_address**]**

   cid -> cName, email, cMobileNo

2. delivery_boy Table

(D) [deliveryNo, oid (foreign key references O.oid), did(foreign key references Delboy.dName), departureTime, arrivalTime**]**

did -> dName, dSalary, dMobileNo

dMobileNo -> did, dName, dSalary

3. chef Table

(Ch) [chid, chName, chSalary, chMobileNo**]**

chid -> chName, chSalary, chMobileNo

chMobileNo -> chid, chName, chSalary

4. delBoy_delivery Table

(Delboy) [dName, dSalary, dMobileNo**]**

deliveryNo -> departureTime, arrivalTime, did

5. foodItems Table

(F) [foodId, fName, price, size**]**

foodId -> fName, price, size, chid

size -> price, foodId

name -> chid

6. orders Table

(O) [oid, cid (foreign key references C.Cid), date_time, paymentAmount, paymentStatus_YN**]**

oid -> date_time, delivery_add, pid, cid

pid -> paymentAmount, paymentStatus_YN, oid

7. orderDetails Table

(OD)[oid (foreign key references O.oid), foodId (foreign key references F.foodId), amount**]**

foodId, oid -> amount

NORMALISATION

First Normal Form

- Ø Customer Table is in 1NF, as each column contains atomic values and there are no repeating groups
- Ø Chef Table is in 1NF, as each column contains atomic values and there are no repeating groups
- Ø delBoy Delivery Table is in 1NF, as each column contains atomic values and there are no repeating groups
- Ø Orders Table is in 1NF, as each column contains atomic values and there are no repeating groups
- Ø Food Items Table is in 1NF, as each column contains atomic values and there are no repeating groups
- Ø Order Details Table is in 1NF, as each column contains atomic values and there are no repeating groups
- Ø Delivery Boy Table is in 1NF, as each column contains atomic values and there are no repeating groups

Second Normal Form

- Ø Customer Table is in 2NF, as there are no combination candidate keys thus there is no partial dependency
- Ø Chef Table is in 2NF, as there are no combination candidate keys thus there is no partial dependency
- Ø delBoy Delivery Table is in 2NF, as there are no combination candidate keys thus there is no partial dependency
- Ø Orders Table is in 2NF, as there are no combination candidate keys thus there is no partial dependency
- Ø Food Table is not in 2NF, as there is partial dependency of chid on fName. Hence we decompose it into two tables i.e.
  foodItems [foodId, fName, size, price]
  foodChef [fName, chid]
  foodItems and foodChef are in 2NF, , as there are is no partial dependency.
- Ø Orders Table is in 2NF, as there are no combination candidate keys thus there is no partial dependency
- Ø Delivery Table is in 2NF, as there are no combination candidate keys thus there is no partial dependency

Three Normal Form

- Ø Customer Table is in 3NF, as there are no transistive dependencies.
- Ø Chef Table is in 3NF, as there are no transistive dependencies.
- Ø delBoy Delivery Table is in 3NF, as there are no transistive dependencies.
- Ø Orders Table is in 3NF, as there are no transistive dependencies. Food Items Table is in 3NF, as there are no transistive dependencies.
- Ø Order Details Table is in 3NF, as there are no transistive dependencies.
- Ø Delivery Boy Table is in 3NF, as there are no transistive dependencies.

BCNF Normal Form

- Ø Customer Table is in BCNF, as all elements on left hand site of every functional dependency is super key
- Ø Chef Table is in BCNF, as all elements on left hand site of every functional dependency is super key
- Ø delBoy Delivery Table is in BCNF, as all elements on left hand site of every functional dependency is super key
- Ø Orders Table is in BCNF, as all elements on left hand site of every functional dependency is super key
- Ø Food Items Table is in BCNF, as all elements on left hand site of every functional dependency is super key
- Ø Order Details Table is in BCNF, as all elements on left hand site of every functional dependency is super key
- Ø Delivery Boy Table is in BCNF, as all elements on left hand site of every functional dependency is super key

4NF Normal Form

- Ø Customer Table is in 4NF, as no multivalued functional dependencies are present
- Ø Chef Table is in 4NF, as no multivalued functional dependencies are present
- Ø delBoy Delivery Table is in 4NF, as no multivalued functional dependencies are present
- Ø Orders Table is in 4NF, as no multivalued functional dependencies are present
- Ø Food Items Table is in 4NF, as no multivalued functional dependencies are present
- Ø Order Details Table is in 4NF, as no multivalued functional dependencies are present
- Ø Delivery Boy Table is in 4NF, as no multivalued functional dependencies are present

5NF Normal Form

- Ø Customer Table is in 5NF, as no non implied joint functional dependencies are present
- Ø Chef Table is in 5NF, as no non implied joint functional dependencies are present
- Ø delBoy Delivery Table is in 5NF, as no non implied joint functional dependencies are present
- Ø Orders Table is in 5NF, as no non implied joint functional dependencies are present
- Ø Food Items Table is in 5NF, as no non implied joint functional dependencies are present
- Ø Order Details Table is in 5NF, as no non implied joint functional dependencies are present
- Ø Delivery Boy Table is in 5NF, as no non implied joint functional dependencies are present

FINAL
RELATIONAL SCHEMA

1. Customer Table (C)

- cid
- cName
- email
- cMobileNo

2. Chef Table (Ch)

- **c**hid,
- chName
- chSalary
- chMobileNo**]**

3. Food Item Table (F)

- foodId
- fName
- price
- size

4. Order Table (O)

- oid
- cid (foreign key references C.cid)
- date_time
- cust_review
- pid
- deliveryNo
- paymentAmout
- paymentStatus_YN**]**

5. Delivery_boy Table (Delboy)

- dName
- dSalary
- dMobileNo

6. Delivery Table (D)

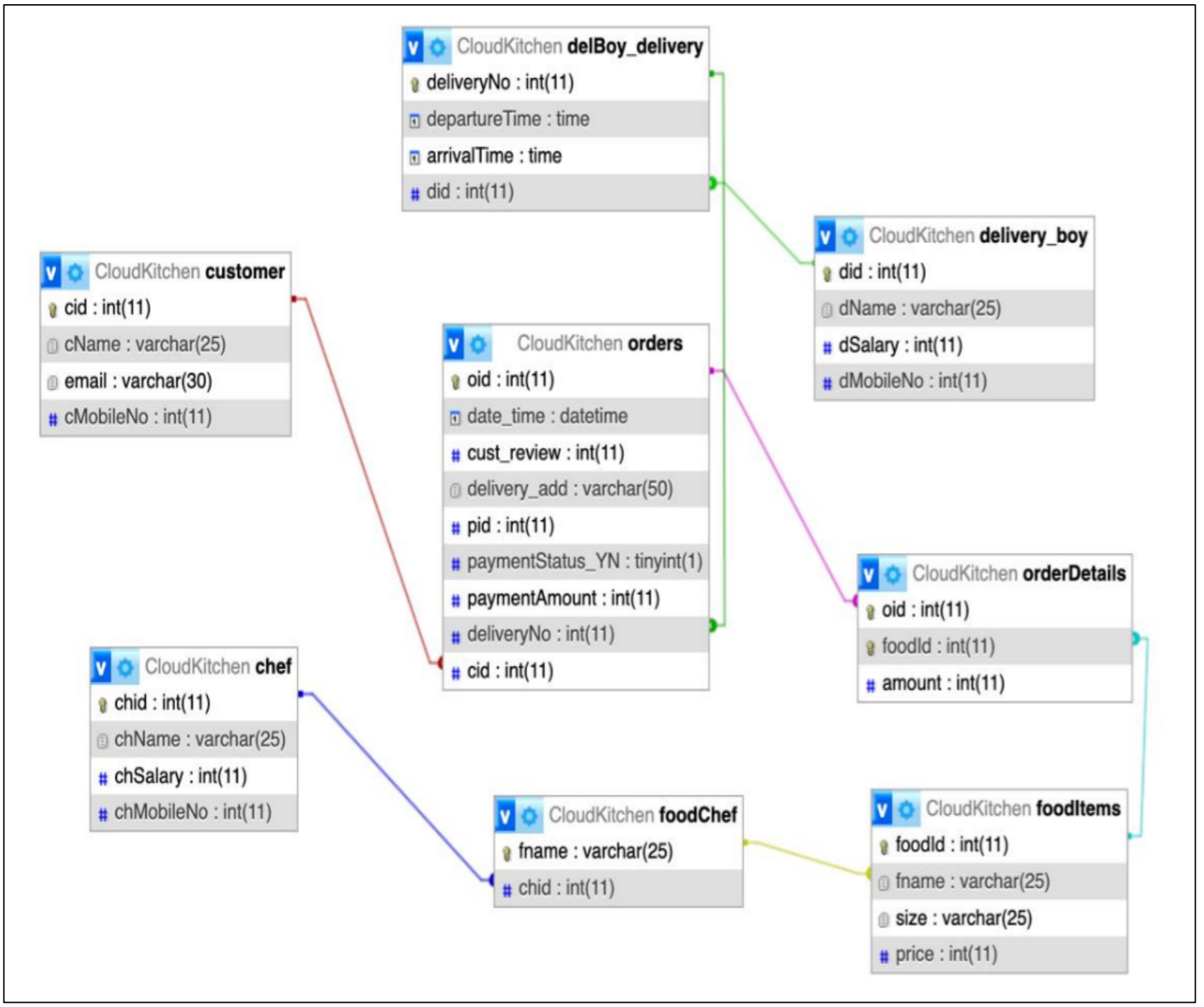
- deliveryNo (foreign key references O.deliveryNo)
- did (foreign key references delBoy.did)
- departureTime
- arrivalTime

7. Order Details Table (OD)

oid(foreign key reference O.oid)
foodId
amount

RELATIONAL SCHEMA

A **relational schema** is a set of relational tables and associated items that are related to one another. All of the base tables, views, indexes, domains, user roles, stored modules, and other items that a user creates to fulfill the data needs of a particular enterprise or set of applications belong to one schema.



CREATION OF DATABASE

Server: localhost

Databases | SQL | Status | User accounts | Export | Import

Run SQL query/queries on server "localhost":

```
1 CREATE DATABASE cloudKitchen;
2
```

CREATION OF TABLES

CUSTOMER TABLE

Run SQL query/queries on database cloudKitchen:

```
1 CREATE TABLE customer(
2 cid INT(11) NOT NULL PRIMARY KEY AUTO_INCREMENT,
3 cName VARCHAR(25),
4 email VARCHAR(30),
5 cMobileNo VARCHAR(11)
6 );
```

CHEF TABLE

Run SQL query/queries on database cloudKitchen:

```
1 CREATE TABLE chef(
2 chid INT(11)  PRIMARY KEY AUTO_INCREMENT,
3 chName VARCHAR(25),
4 chSalary INT(11),
5 chMobileNo VARCHAR(11)
6 );
```

DELIVERY BOY TABLE

Run SQL query/queries on database cloudKitchen:

```
CREATE TABLE delivery_boy(
did INT(11)  PRIMARY KEY AUTO_INCREMENT,
dName VARCHAR(25),
dSalary INT(11),
dMobileNo VARCHAR(11)
);
```

FOOD CHEF TABLE

Run SQL query/queries on database cloudKitchen:

```
CREATE TABLE foodChef(
fname VARCHAR(25) PRIMARY KEY,
chid INT(11),
FOREIGN KEY (chid) REFERENCES chef(chid)
);
```

FOOD ITEMS TABLE

un SQL query/queries on database cloudKitchen:

```
CREATE TABLE foodItems(
foodId int(11) PRIMARY KEY AUTO_INCREMENT,
fname VARCHAR (25) NOT NULL,
size VARCHAR(25) NOT NULL DEFAULT("regular"),
price INT(11),
FOREIGN KEY (fname) REFERENCES foodChef(fname)
);
```

DELBOY DELIVERY TABLE

Run SQL query/queries on database cloudKitchen:

```
CREATE TABLE delBoy_delivery (
deliveryNo INT (11) PRIMARY KEY AUTO_INCREMENT,
departureTime TIME,
arrivalTime TIME,
did INT(11),
FOREIGN KEY (did) REFERENCES delivery_boy(did)
);
```

ORDER DETAILS TABLE

Run SQL query/queries on database cloudKitchen:

```
CREATE TABLE orderDetails(
oid INT(11),
foodId INT(11),
amount INT(11),
CONSTRAINT primaryKey PRIMARY KEY(oid, foodId),
FOREIGN KEY (oid) REFERENCES orders(oid),
FOREIGN KEY (foodId) REFERENCES foodItems(foodId)
);
```

ORDERS TABLE

SQL query/queries on database cloudKitchen:

```
CREATE TABLE orders(
oid INT(11) PRIMARY KEY AUTO_INCREMENT,
date_time DATETIME ,
cust_review INT(11),
delivery_add VARCHAR(50),
pid VARCHAR(25) UNIQUE,
paymentStatus_YN BOOL,
paymentAmount INT (11),
deliveryNo INT(11),
cid INT (11),
FOREIGN KEY (cid) REFERENCES customer(cid),
FOREIGN KEY (deliveryNo) REFERENCES delBoy_delivery(deliveryNo),
constraint reviewLessThanTen check (cust_review <10)
 );
```

INSERTION OF DATA INTO TABLES

CHEF TABLE

Run SQL query/queries on database cloudKitchen:

```sql
INSERT INTO chef (chName, chSalary, chMobileNo) VALUES
('Vikram Singh', 35000, 9876543210),
('Ranjit Kaur', 32000, 8765432109),
('Amit Sharma', 38000, 7654321098),
('Preeti Verma', 36000, 6543210987),
('Rajesh Kumar', 34000, 5432109876);
```

DELIVERY BOY TABLE

Run SQL query/queries on database cloudKitchen:

```sql
INSERT INTO delivery_boy (dName, dSalary, dMobileNo) VALUES
('Rahul Verma', 25000, '9876543210'),
('Suresh Kumar', 24000, '8765432109'),
('Amit Singh', 26000, '7654321098'),
('Priya Sharma', 23000, '6543210987'),
('Anil Yadav', 24500, '5432109876');
```

CUSTOMER TABLE

Run SQL query/queries on database cloudKitchen:

```sql
INSERT INTO customer (cName, email, cMobileNo) VALUES
('Rahul Kumar', 'rahul@example.com', '9876543210'),
('Sneha Gupta', 'sneha@example.com', '8765432109'),
('Amit Singh', 'amit@example.com', '7654321098'),
('Pooja Sharma', 'pooja@example.com', '6543210987'),
('Nitin Verma', 'nitin@example.com', '5432109876'),
('Kavita Yadav', 'kavita@example.com', '4321098765'),
('Vikram Singh', 'vikram@example.com', '3210987654'),
('Anjali Reddy', 'anjali@example.com', '2109876543'),
('Alok Patel', 'alok@example.com', '1098765432'),
('Priya Kapoor', 'priya@example.com', '9876543211'),
('Rajesh Kumar', 'rajesh@example.com', '8765432101'),
('Neha Gupta', 'neha@example.com', '7654321092'),
('Sumit Sharma', 'sumit@example.com', '6543210983'),
('Riya Verma', 'riya@example.com', '5432109874'),
('Sachin Yadav', 'sachin@example.com', '4321098765'),
```

and more . . . .

FOODCHEF TABLE

```sql
INSERT INTO foodChef (fname, chid) VALUES
('Paneer Tikka', 1),
('Vegetable Biryani', 1),
('Palak Paneer', 4),
('Chole Bhature', 5),
('Samosa', 5),
('Rajma Chawal', 3),
('Aloo Paratha', 2),
('Matar Paneer', 3),
('Vegetable Pulao', 3),
('Pav Bhaji', 1),
('Tandoori Roti', 2),
('Naan', 1),
('Aloo Tikki', 4),
```

and more . . . .

FOOD ITEMS TABLE

Run SQL query/queries on database cloudKitchen:

```sql
INSERT INTO foodItems (fname, size, price) VALUES
('Paneer Tikka', 'regular', 300),
('Paneer Tikka', 'large', 350),
('Vegetable Biryani', 'regular', 250),
('Palak Paneer', 'regular', 200),
('Rajma Chawal', 'regular', 200),
('Aloo Paratha', 'regular', 40),
('Matar Paneer', 'regular', 250),
('Vegetable Pulao', 'regular', 180),
('Pav Bhaji', 'regular', 120),
('Tandoori Roti', 'regular', 15),
('Tandoori Roti', 'butter', 20),
('Naan', 'regular', 20),
('Naan', 'butter', 25),
('Aloo Tikki', 'regular', 40),
('Chana Masala', 'regular', 150),
```

and more . . . .

ORDERS TABLE

Run SQL query/queries on database cloudKitchen:

```sql
INSERT INTO orders (date_time, cust_review, delivery_add, pid, paymentStatus_YN, cid)
VALUES
('2024-03-01 10:00:00', 8, '123 Main St, Delhi', 'ICI2203011000123',1, 1),
('2024-03-02 11:00:00', 9, '456 Elm St, Delhi', 'HDF2203021100456',1, 2),
('2024-03-03 12:00:00', 7, '789 Oak St, Delhi', 'AXI2203031200789',1, 3),
('2024-03-04 13:00:00', 6, '101 Pine St, Delhi', 'KOT2203041300101',1, 4),
('2024-03-05 14:00:00', 9, '222 Maple St, Delhi', 'BOI2203051400222',1, 5),
('2024-03-06 15:00:00', 8, '333 Birch St, Delhi', 'IDB2203061500333',1, 6),
('2024-03-07 16:00:00', 7, '444 Cedar St, Delhi', 'UBI2203071600444',1, 7),
('2024-03-08 17:00:00', 9, '555 Walnut St, Delhi', 'PNB2203081700555',1, 8),
('2024-03-09 18:00:00', 8, '666 Spruce St, Delhi', 'SBI2203091800666',1, 9),
('2024-03-10 19:00:00', 7, '777 Ash St, Delhi', 'BOB2203101900777',1, 10),
('2024-03-11 20:00:00', 6, '888 Oak St, Delhi', 'ICI2203112000888',1, 11),
('2024-03-12 21:00:00', 9, '999 Elm St, Delhi', 'HDF2203122100999',1, 12),
('2024-03-13 10:30:00', 8, '111 Pine St, Delhi', 'AXI2203131030111',1, 13),
('2024-03-14 11:30:00', 7, '222 Maple St, Delhi', 'KOT2203141130222',1, 14),
```

and more . . . .

ORDER DETAILSTABLE

Run SQL query/queries on database cloudKitchen:

```sql
INSERT INTO orderDetails (oid, foodId, amount) VALUES
(1, 1, 3),
(1, 12, 2),
(1, 19, 3),
(1, 8, 2),
(2, 14, 3),
(2, 5, 1),
(2, 18, 4),
(2, 22, 1),
(3, 10, 3),
(3, 16, 1),
(3, 4, 4),
(3, 20, 3),
(4, 7, 3),
(4, 15, 2),
(4, 23, 4),
```

and more . . . .

ORDERS TABLE

Run SQL query/queries on database cloudKitchen:

```
1  INSERT INTO delBoy_delivery (departureTime, arrivalTime, did) VALUES
2  ('10:07:00', '10:18:00',1),
3  ('11:08:00', '11:19:00',2),
4  ('12:10:00', '12:20:00',3),
5  ('13:10:00', '13:21:00',1),
6  ('14:11:00', '14:22:00',3),
7  ('15:10:00', '15:21:00',5),
8  ('16:05:00', '16:16:00',4),
9  ('17:09:00', '17:20:00',3),
10 ('18:07:00', '18:17:00',2),
11 ('19:12:00', '19:23:00',5),
12 ('20:09:00', '20:20:00',3),
13 ('21:14:00', '21:25:00',1),
14 ('10:37:00', '10:48:00',1),
15 ('11:32:00', '11:43:00',2),
16 ('12:38:00', '12:49:00',2),
```

and more . . . .

TABLES INTO DATABASE

CHEF TABLE

Showing rows 0 - 4 (5 total, Query took 0.0005 seconds.)

```
SELECT * FROM `chef`
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

Extra options

	chid	chName	chSalary	chMobileNo
Edit Copy Delete	1	Vikram Singh	35000	9876543210
Edit Copy Delete	2	Ranjit Kaur	32000	8765432109
Edit Copy Delete	3	Amit Sharma	38000	7654321098
Edit Copy Delete	4	Preeti Verma	36000	6543210987
Edit Copy Delete	5	Rajesh Kumar	34000	5432109876

CUSTOMER TABLE

Showing rows 0 - 24 (29 total, Query took 0.0004 seconds.)

```
SELECT * FROM `customer`
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

1 > >> | Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

Extra options

	cid	cName	email	cMobileNo
Edit Copy Delete	1	Rahul Kumar	rahul@example.com	9876543210
Edit Copy Delete	2	Sneha Gupta	sneha@example.com	8765432109
Edit Copy Delete	3	Amit Singh	amit@example.com	7654321098
Edit Copy Delete	4	Pooja Sharma	pooja@example.com	6543210987
Edit Copy Delete	5	Nitin Verma	nitin@example.com	5432109876
Edit Copy Delete	6	Kavita Yadav	kavita@example.com	4321098765
Edit Copy Delete	7	Vikram Singh	vikram@example.com	3210987654
Edit Copy Delete	8	Anjali Reddy	anjali@example.com	2109876543
Edit Copy Delete	9	Alok Patel	alok@example.com	1098765432
Edit Copy Delete	10	Priya Kapoor	priya@example.com	9876543211
Edit Copy Delete	11	Rajesh Kumar	rajesh@example.com	8765432101
Edit Copy Delete	12	Neha Gupta	neha@example.com	7654321092
Edit Copy Delete	13	Sumit Sharma	sumit@example.com	6543210983
Edit Copy Delete	14	Riya Verma	riya@example.com	5432109874
Edit Copy Delete	15	Sachin Yadav	sachin@example.com	4321098765
Edit Copy Delete	16	Preeti Singh	preeti@example.com	3210987656
Edit Copy Delete	17	Arjun Reddy	arjun@example.com	2109876547

DELBOY DELIVERY TABLE

Showing rows 0 - 39 (40 total, Query took 0.0002 seconds.)

```
SELECT * FROM `delBoy_delivery`
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 100 Filter rows: Search this table Sort by key: None

Extra options

	deliveryNo	departureTime	arrivalTime	did
Edit Copy Delete	1	10:07:00	10:18:00	1
Edit Copy Delete	2	11:08:00	11:19:00	2
Edit Copy Delete	3	12:10:00	12:20:00	3
Edit Copy Delete	4	13:10:00	13:21:00	1
Edit Copy Delete	5	14:11:00	14:22:00	3
Edit Copy Delete	6	15:10:00	15:21:00	5
Edit Copy Delete	7	16:05:00	16:16:00	4
Edit Copy Delete	8	17:09:00	17:20:00	3
Edit Copy Delete	9	18:07:00	18:17:00	2
Edit Copy Delete	10	19:12:00	19:23:00	5
Edit Copy Delete	11	20:09:00	20:20:00	3
Edit Copy Delete	12	21:14:00	21:25:00	1
Edit Copy Delete	13	10:37:00	10:48:00	1
Edit Copy Delete	14	11:32:00	11:43:00	2
Edit Copy Delete	15	12:38:00	12:49:00	2
Edit Copy Delete	16	13:38:00	13:49:00	4
Edit Copy Delete	17	14:37:00	14:48:00	4
Edit Copy Delete	18	15:32:00	15:43:00	4

Console

DELIVERY BOY TABLE

Showing rows 0 - 4 (5 total, Query took 0.0002 seconds.)

```
SELECT * FROM `delivery_boy`
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

Extra options

	did	dName	dSalary	dMobileNo
Edit Copy Delete	1	Rahul Verma	25000	9876543210
Edit Copy Delete	2	Suresh Kumar	24000	8765432109
Edit Copy Delete	3	Amit Singh	26000	7654321098
Edit Copy Delete	4	Priya Sharma	23000	6543210987
Edit Copy Delete	5	Anil Yadav	24500	5432109876

Check all With selected: Edit Copy Delete Export

FOODCHEF TABLE

Showing rows 0 - 14 (15 total, Query took 0.0002 seconds.)

```
SELECT * FROM `foodChef`
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

Extra options

	fname	chid
Edit Copy Delete	Naan	1
Edit Copy Delete	Paneer Tikka	1
Edit Copy Delete	Pav Bhaji	1
Edit Copy Delete	Vegetable Biryani	1
Edit Copy Delete	Aloo Paratha	2
Edit Copy Delete	Chana Masala	2
Edit Copy Delete	Tandoori Roti	2
Edit Copy Delete	Matar Paneer	3
Edit Copy Delete	Rajma Chawal	3
Edit Copy Delete	Vegetable Pulao	3
Edit Copy Delete	Aloo Tikki	4
Edit Copy Delete	Baingan Bharta	4
Edit Copy Delete	Palak Paneer	4
Edit Copy Delete	Chole Bhature	5
Edit Copy Delete	Samosa	5

FOODITEMS TABLE

Showing rows 0 - 23 (24 total, Query took 0.0002 seconds.)

```
SELECT * FROM `foodItems`
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table Sort by

Extra options

	foodId	fname	size	price
Edit Copy Delete	1	Paneer Tikka	regular	300
Edit Copy Delete	2	Paneer Tikka	large	350
Edit Copy Delete	3	Vegetable Biryani	regular	250
Edit Copy Delete	4	Palak Paneer	regular	200
Edit Copy Delete	5	Rajma Chawal	regular	200
Edit Copy Delete	6	Aloo Paratha	regular	40
Edit Copy Delete	7	Matar Paneer	regular	250
Edit Copy Delete	8	Vegetable Pulao	regular	180
Edit Copy Delete	9	Pav Bhaji	regular	120
Edit Copy Delete	10	Tandoori Roti	regular	15
Edit Copy Delete	11	Tandoori Roti	butter	20
Edit Copy Delete	12	Naan	regular	20
Edit Copy Delete	13	Naan	butter	25
Edit Copy Delete	14	Aloo Tikki	regular	40
Edit Copy Delete	15	Chana Masala	regular	150
Edit Copy Delete	16	Baingan Bharta	regular	180
Edit Copy Delete	17	Chole Bhature	regular	170
Edit Copy Delete	18	Samosa	regular	25

Console

ORDER DETAILS TABLE

Showing rows 0 - 159 (160 total, Query took 0.0003 seconds.)

```
SELECT * FROM `orderDetails`
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 250 Filter rows: Search this table Sort by key: None

Extra options

	oid	foodId	amount
Edit Copy Delete	1	1	3
Edit Copy Delete	1	8	2
Edit Copy Delete	1	12	2
Edit Copy Delete	1	19	3
Edit Copy Delete	2	5	1
Edit Copy Delete	2	14	3
Edit Copy Delete	2	18	4
Edit Copy Delete	2	22	1
Edit Copy Delete	3	4	4
Edit Copy Delete	3	10	3
Edit Copy Delete	3	16	1
Edit Copy Delete	3	20	3
Edit Copy Delete	4	2	2
Edit Copy Delete	4	7	3
Edit Copy Delete	4	15	2
Edit Copy Delete	4	23	4
Edit Copy Delete	5	3	1

Console

ORDERS TABLE

Showing rows 0 - 39 (40 total, Query took 0.0002 seconds.)

```
SELECT * FROM `orders`
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 50 Filter rows: Search this table Sort by key: None

Extra options

	oid	date_time	cust_review	delivery_add	pid	paymentStatus_YN	paymentAmount	deliveryNo	cid
Edit Copy Delete	1	2024-03-01 10:00:00	8	123 Main St, Delhi	ICI2203011000123	1	1480	1	1
Edit Copy Delete	2	2024-03-02 11:00:00	9	456 Elm St, Delhi	HDF2203021100456	1	570	2	2
Edit Copy Delete	3	2024-03-03 12:00:00	7	789 Oak St, Delhi	AXI2203031200789	1	1925	3	3
Edit Copy Delete	4	2024-03-04 13:00:00	6	101 Pine St, Delhi	KOT2203041300101	1	1950	4	4
Edit Copy Delete	5	2024-03-05 14:00:00	9	222 Maple St, Delhi	BOI2203051400222	1	1510	5	5
Edit Copy Delete	6	2024-03-06 15:00:00	8	333 Birch St, Delhi	IDB2203061500333	1	525	6	6
Edit Copy Delete	7	2024-03-07 16:00:00	7	444 Cedar St, Delhi	UBI2203071600444	1	1460	7	7
Edit Copy Delete	8	2024-03-08 17:00:00	9	555 Walnut St, Delhi	PNB2203081700555	1	540	8	8
Edit Copy Delete	9	2024-03-09 18:00:00	8	666 Spruce St, Delhi	SBI2203091800666	1	1060	9	9
Edit Copy Delete	10	2024-03-10 19:00:00	7	777 Ash St, Delhi	BOB2203101900777	1	2290	10	10
Edit Copy Delete	11	2024-03-11 20:00:00	6	888 Oak St, Delhi	ICI2203112000888	1	2140	11	11
Edit Copy Delete	12	2024-03-12 21:00:00	9	999 Elm St, Delhi	HDF2203122100999	1	945	12	12
Edit Copy Delete	13	2024-03-13 10:30:00	8	111 Pine St, Delhi	AXI2203131030111	1	1240	13	13
Edit Copy Delete	14	2024-03-14 11:30:00	7	222 Maple St, Delhi	KOT2203141130222	1	970	14	14
Edit Copy Delete	15	2024-03-15 12:30:00	9	333 Birch St, Delhi	BOI2203151230333	1	745	15	15
Edit Copy Delete	16	2024-03-16 13:30:00	8	444 Cedar St, Delhi	IDB2203161330444	1	1925	16	16
Edit Copy Delete	17	2024-03-17 14:30:00	7	555 Walnut St, Delhi	UBI2203171430555	1	810	17	17
Edit Copy Delete	18	2024-03-18 15:30:00	6	666 Spruce St, Delhi	PNB2203181530666	1	1045	18	18

Console

SQL QUERIES

Some basic queries covered in the project:

- No. of orders placed between date.
- Display food menu (name,price,rating)
- Delivery count by boys.
- Food item prepared by chef.
- Customer order history.

Implementation using SQL:

```sql
-- count of orders between two dates
select * from orders;
SELECT COUNT(*) AS num_orders
FROM orders
WHERE date_time BETWEEN '2024:03:15' AND '2024:04:01';
-- menu
SELECT f.fname, f.price, AVG(o.cust_review) AS avg_review
FROM foodItems f
LEFT JOIN orderDetails od ON f.foodId = od.foodId
LEFT JOIN orders o ON od.oid = o.oid
GROUP BY f.fname, f.price;
-- count of delivery by delivery boys
SELECT d.did, d.dName, COUNT(*) AS delivery_count
FROM delBoy_delivery dd
INNER JOIN delivery_boy d ON dd.did = d.did
GROUP BY d.did, d.dName;
-- food items prepared by a chef
SELECT fc.fname
FROM foodChef fc
INNER JOIN chef c ON fc.chid = c.chid
WHERE c.chid = 5 ;
-- customer order history
SELECT fi.fname, COUNT(*) AS frequency
FROM orders o
INNER JOIN orderDetails od ON o.oid = od.oid
INNER JOIN foodItems fi ON od.foodId = fi.foodId
WHERE o.cid = 1 -- customer id
GROUP BY fi.fname;
```

Implementation using gui in python is done further

FRONTEND USING PYTHON

CONNECTION ESTABLISHED

```python
#Connection to Xampp Database
import mysql.connector

mydb = mysql.connector.connect(host='localhost',
                               user='root',
                               password='',
                               database='phpmyadmin',
                               port='3307')
if mydb.is_connected():
    print("connected")
```

```
/Users/girishporwal/Desktop/CodingRelated/python/pythonProject/.venv/bin/python /Users/girishporwal/Desktop/CodingRelated/pyt
connected

Process finished with exit code 0
```

PYTHON CODE

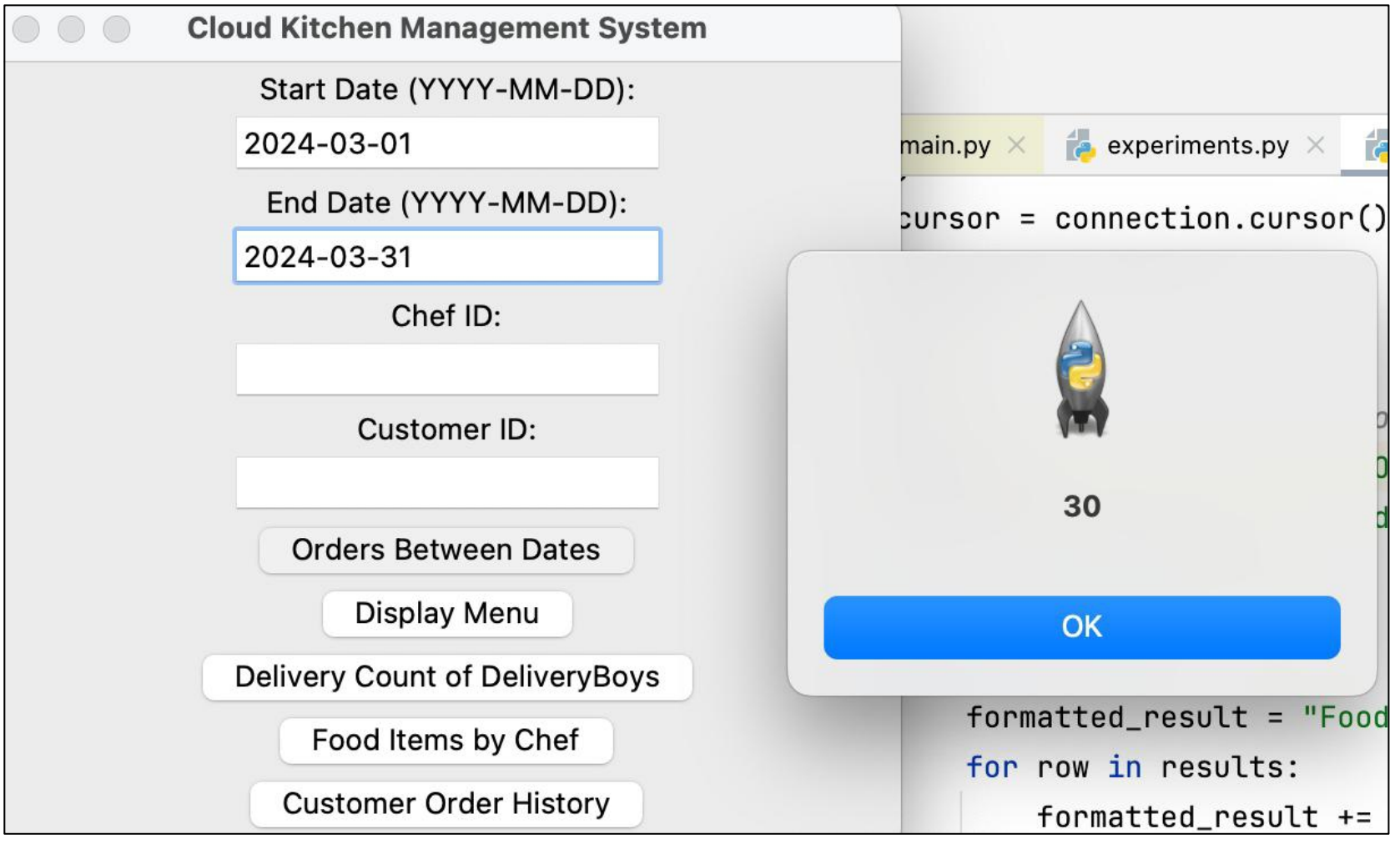
```python
import mysql.connector
from tkinter import *
from tkinter import messagebox

# Function to execute query and display result in a messagebox
5 usages
def execute_query(query, query_type=None):
    cursor = None
    try:
        connection = mysql.connector.connect(
            host='localhost',
            user='root',
            password='',
            database='cloudKitchen',
            port='3307'
        )
        cursor = connection.cursor()
        cursor.execute(query)
        results = cursor.fetchall()

        # Format the results based on the query type
        if query_type == "Customer Order History":
            formatted_result = "Food Item\t\tFrequency\n"
            for row in results:
                formatted_result += f"{row[0]:<30}{row[1]}\n"
        elif query_type == "Display Menu":
            formatted_result = "Food Item\t\tPrice\t\tAverage Review\n"
            for row in results:
                formatted_result += f"{row[0]:<30}{row[1]:<15}{row[2]}\n"
        else:
            formatted_result = ""
            for row in results:
                formatted_result += ", ".join(map(str, row)) + "\n"

```

```
            # Display messagebox with formatted result
            if formatted_result.strip():
                messagebox.showinfo( title: "Query Result", formatted_result)
            else:
                messagebox.showinfo( title: "Query Result",  message: "No records found.")

        except mysql.connector.Error as e:
            messagebox.showerror( title: "Error",  message: f"Error executing query: {e}")
        finally:
            if cursor:
                cursor.close()
            if connection:
                connection.close()

    # Function to create GUI and execute queries
    1 usage
    def create_gui():
        root = Tk()
        root.title("Cloud Kitchen Management System")
        root.geometry("400x350")

        # Define functions for each query
        def query_orders_between_dates():
            start_date = entry_start_date.get()
            end_date = entry_end_date.get()
            query = f"SELECT COUNT(*) FROM orders WHERE date_time BETWEEN '{start_date}' AND '{end_date}'"
            execute_query(query)

        def query_display_menu():
            query = """SELECT f.fname, f.price, AVG(o.cust_review) AS avg_review
                       FROM foodItems f
                       LEFT JOIN orderDetails od ON f.foodId = od.foodId
                       LEFT JOIN orders o ON od.oid = o.oid
                       GROUP BY f.fname, f.price"""
            execute_query(query,  query_type: "Display Menu")

        def query_delivery_count_by_boy():
            query = """SELECT d.did, d.dName, COUNT(*) AS delivery_count
                       FROM delBoy_delivery dd
                       INNER JOIN delivery_boy d ON dd.did = d.did
                       GROUP BY d.did, d.dName"""
            execute_query(query)

        def query_food_items_by_chef():
            ch_id = entry_chef_id.get()
            query = f"SELECT fname FROM foodChef WHERE chid = {ch_id}"
            execute_query(query)

        def query_customer_order_history():
            cust_id = entry_cust_id.get()
            query = f"""SELECT fi.fname, COUNT(*) AS frequency
                       FROM orders o
                       INNER JOIN orderDetails od ON o.oid = od.oid
                       INNER JOIN foodItems fi ON od.foodId = fi.foodId
                       WHERE o.cid = {cust_id}
                       GROUP BY fi.fname"""
            execute_query(query,  query_type: "Customer Order History")

        # Create labels and entry fields for user input
        lbl_start_date = Label(root, text="Start Date (YYYY-MM-DD):")
        lbl_start_date.pack()
        entry_start_date = Entry(root)
        entry_start_date.pack()

        lbl_end_date = Label(root, text="End Date (YYYY-MM-DD):")
        lbl_end_date.pack()
        entry_end_date = Entry(root)
        entry_end_date.pack()

        lbl_chef_id = Label(root, text="Chef ID:")
        lbl_chef_id.pack()
        entry_chef_id = Entry(root)
        entry_chef_id.pack()
```

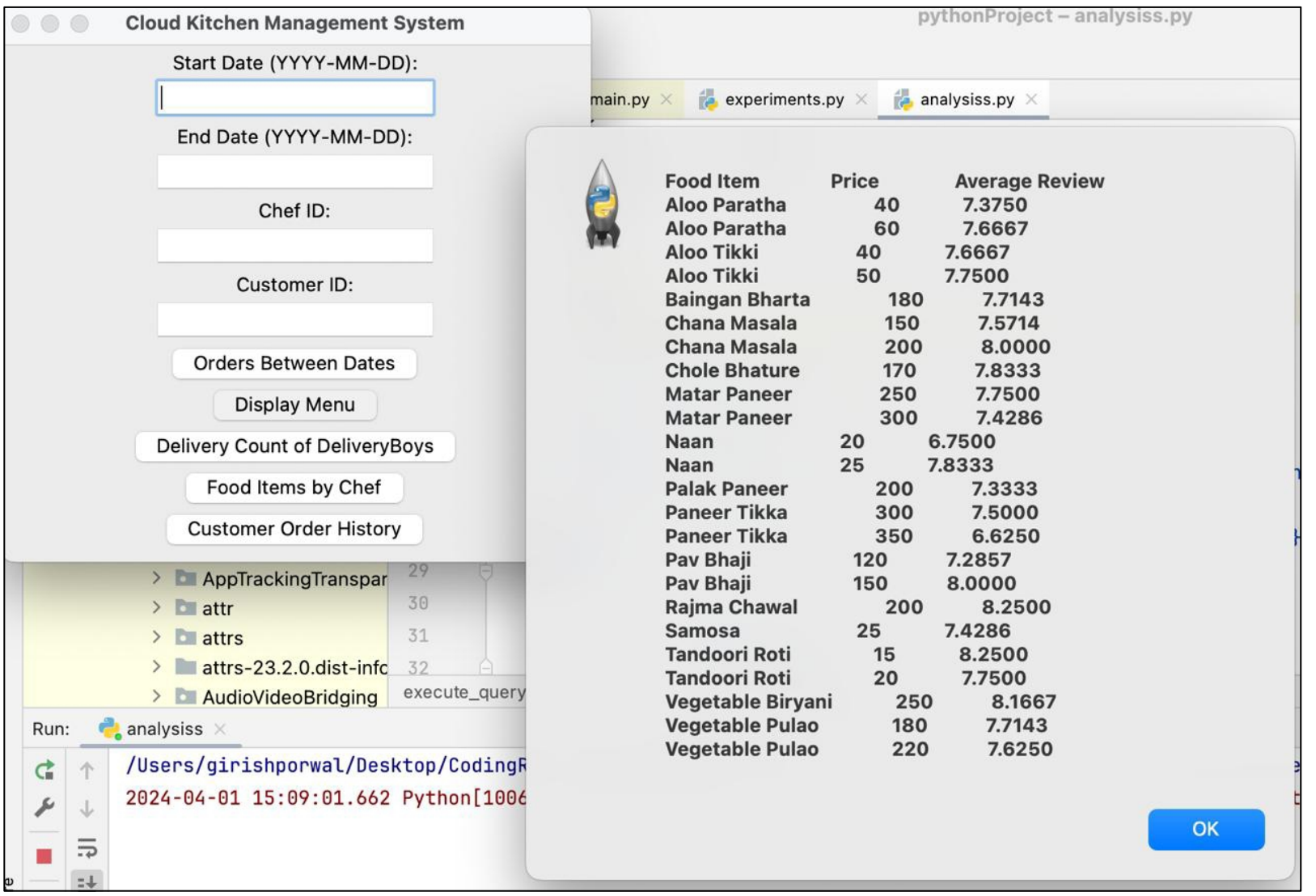
```
                    entry_chef_id.pack()

                    lbl_cust_id = Label(root, text="Customer ID:")
                    lbl_cust_id.pack()
                    entry_cust_id = Entry(root)
                    entry_cust_id.pack()

                    # Create buttons for each query
                    btn_orders_between_dates = Button(root, text="Orders Between Dates", command=query_orders_between_dates)
                    btn_orders_between_dates.pack()

                    btn_display_menu = Button(root, text="Display Menu", command=query_display_menu)
                    btn_display_menu.pack()

                    btn_delivery_count_by_boy = Button(root, text="Delivery Count by Boy", command=query_delivery_count_by_boy)
                    btn_delivery_count_by_boy.pack()

                    btn_food_items_by_chef = Button(root, text="Food Items by Chef", command=query_food_items_by_chef)
                    btn_food_items_by_chef.pack()

                    btn_customer_order_history = Button(root, text="Customer Order History", command=query_customer_order_history)
                    btn_customer_order_history.pack()

                    root.mainloop()

                # Call the function to create GUI and execute queries
                create_gui()

```

OUTPUT

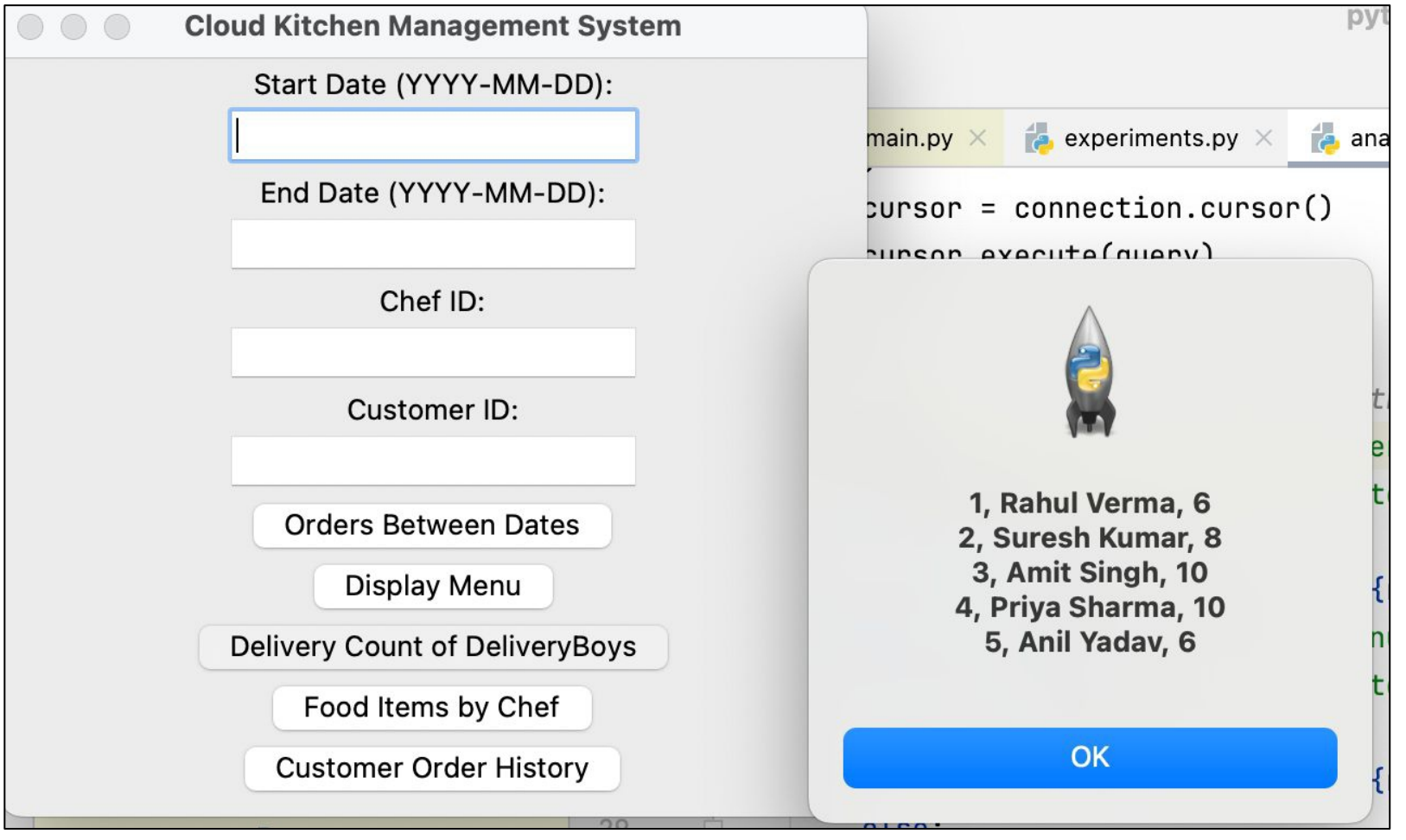
ORDERS BETWEEN DATE



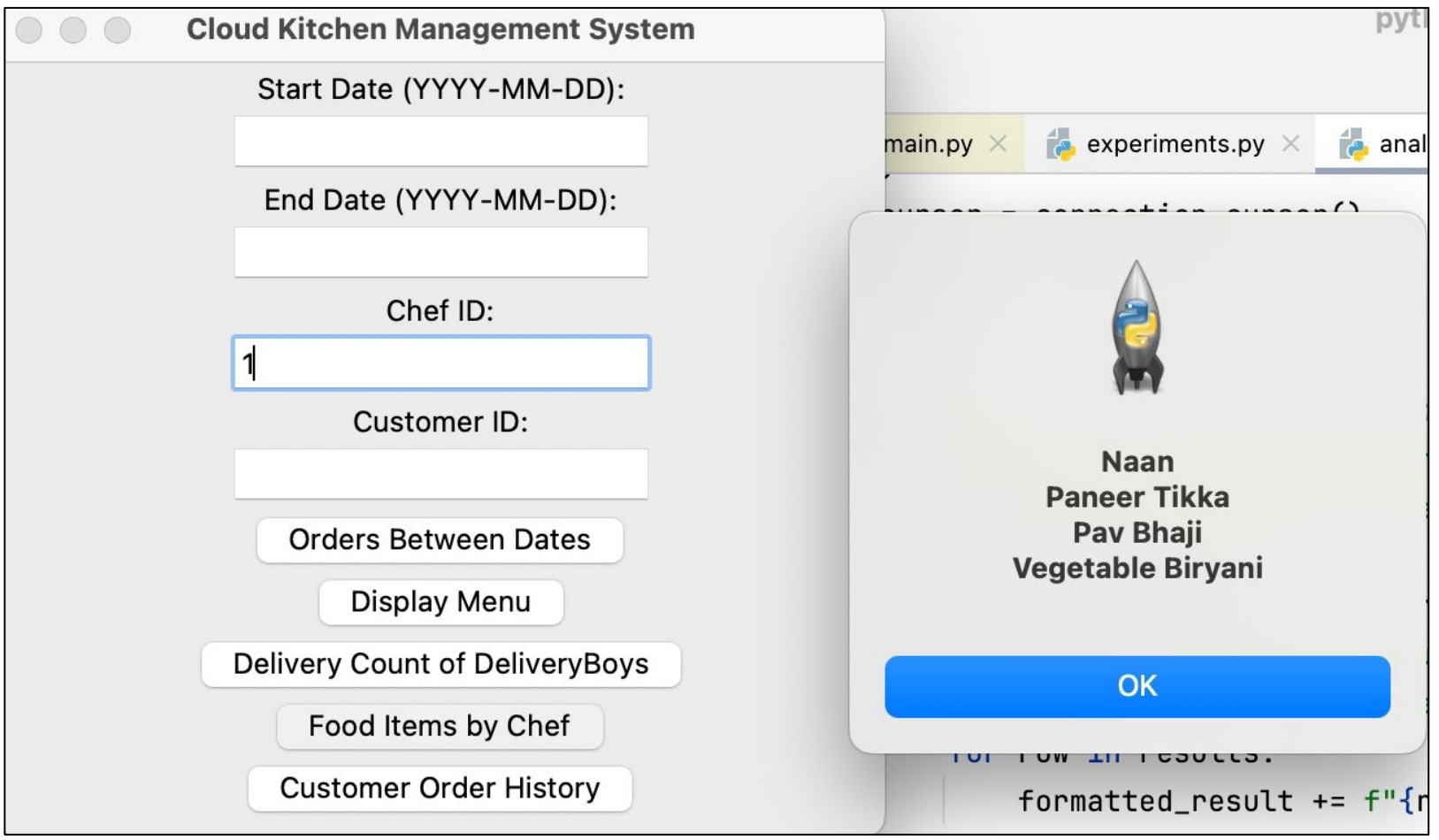
MENU WITH PRICE AND RATING



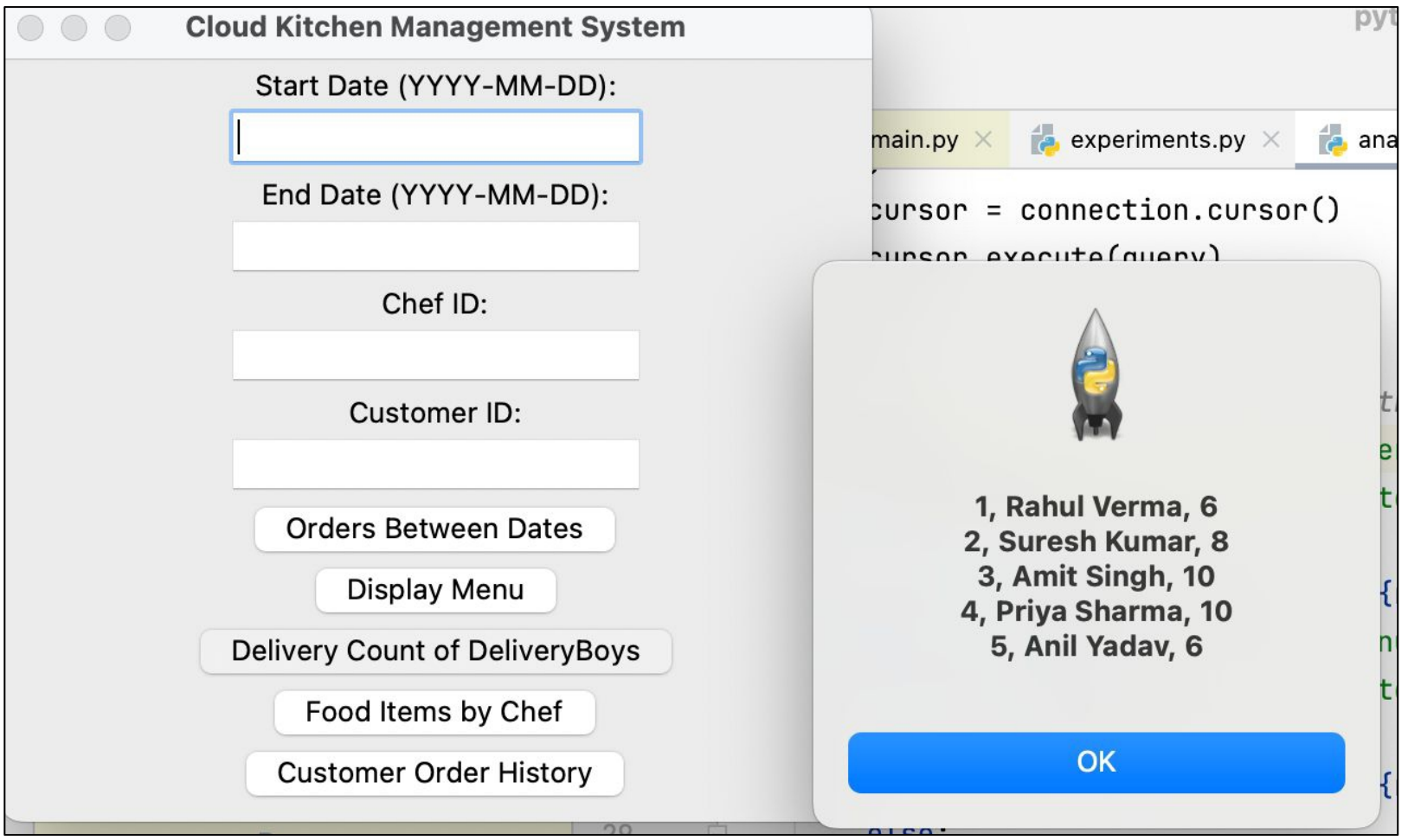
DELIVERY COUNT OF DELIVERY BOYS



FOOD ITEMS MADE BY CHEF



CUSTOMER ORDER HISTORY



SUMMARY

The cloud kitchen system project aims to provide an efficient platform for managing food orders, coordinating delivery services, and ensuring seamless operations within a virtual kitchen environment. The system facilitates the entire process from order placement to delivery, involving various entities such as food items, delivery personnel, chefs, payment status, and orders.

1.Database structure: Customers can place orders through the platform, which are then processed and managed within the system. Orders are assigned unique identifiers and include details such as order status, timestamp, total cost, and delivery address.

2.Menu Management: The system allows administrators to manage the menu by adding, updating, or removing food items. Each food item is associated with attributes like name, description, price, and availability status, ensuring that customers can make informed choices.

3.Delivery Coordination: Delivery personnel, or delivery boys, are assigned orders and provided with necessary information such as delivery address and estimated delivery time. The system tracks the delivery status and facilitates communication between customers, delivery personnel, and administrators.

4.Chef Coordination: Chefs receive notifications of new orders, prepare the required food items, and update the order status accordingly. Chefs may have specific specializations, and their availability status is tracked within the system.

5.Payment Processing: Customers can make payments securely through various payment methods supported by the platform. The system handles payment processing and updates the payment status of each order, ensuring transparency and accountability.

REFRENCES

- An introduction to database systems.
- Practical lab notes
- Class notes
- Sample projects