CIS 9340 - Principles of Database Management Systems

Final Group Project Deliverables

TOPIC - The Restaurant Management Database Project

GROUP MEMBERS –
Krishi Shah
Payal Surana
Sadhvi Grover
Siddarth Bhagirath

PROFESSOR - Howard Kline
Date Of Submission – 05/16/2024

Table of Contents

The Restaurant Management Database Project

The following material documents the design and development of a database application to support a Restaurant - Gourmet Garden. The project begins with a business description and proceeds with logical (Relational) modeling, normalization, and, finally, database application implementation.

I. Business Scenario

Our restaurant, Gourmet Garden, is a fine-dining Asian restaurant nestled in the heart of New York City. Currently, our restaurant utilizes a manual method to allot tables to customers and maintain other details like their contact details, number of diners, etc. We would like to streamline this process and replace this manual method with a database.

In our restaurant, Tables are allotted to Customers upon reaching the restaurant (regardless of whether it's a reservation or a walk-in). We do not have takeaway or online delivery options. Once seated at the allotted table, the customer orders food through a Service Staff member. Based on the order taken, our Chefs prepare the food ordered by the customer.

Our Service Staff and Chefs work different shifts to ensure we efficiently cover lunch and dinner timings, especially during the weekend rush. Once the customer finishes the meal, a bill will be generated for the order. Customers can pay the bill through three payment options: cash, Debit Card, or Credit Card. Our restaurant also provides a feedback form to the customer at the time of payment to help them understand their experience at the restaurant.

Commentary:
Based on the above description, we constructed a Relational Model that captures all the business data needs.

Relationship Sentences:

One **Customer** will be assigned one **table number**.
One **table number** will be assigned to one and only one **Customer**.

One **Service Staff** will take one or more **Orders**.
One **Order** must be taken by one and only one **Service Staff**.

One **Chef** will prepare one or more **Orders**.
One **Order** will be prepared by one or many **Chefs**.

One **Service Staff** will serve one or more **Customers**.
One **Customer** must be served by one and only one **Service Staff**.

One **Order** generates one and only one **Bill**.
One **Bill** will be generated for one **Order**.

One **Payment** will be paid for one and only one **Bill**.
One **Bill** will be paid by one and only one **Payment**.

One **Payment** will be made using one or more **Payment Methods** (Credit Card, Debit Card, and cash).
One **Payment Method** will be used for one or more **Payments**.

One **Customer** will provide one or more **Feedback**.
One **Feedback** is provided by one and only one **Customer**.

Commentary:
The relationship sentences should make sense. In this example, the verb phrases are underlined, and the entity names are in bold letters.

II. The Relational Model

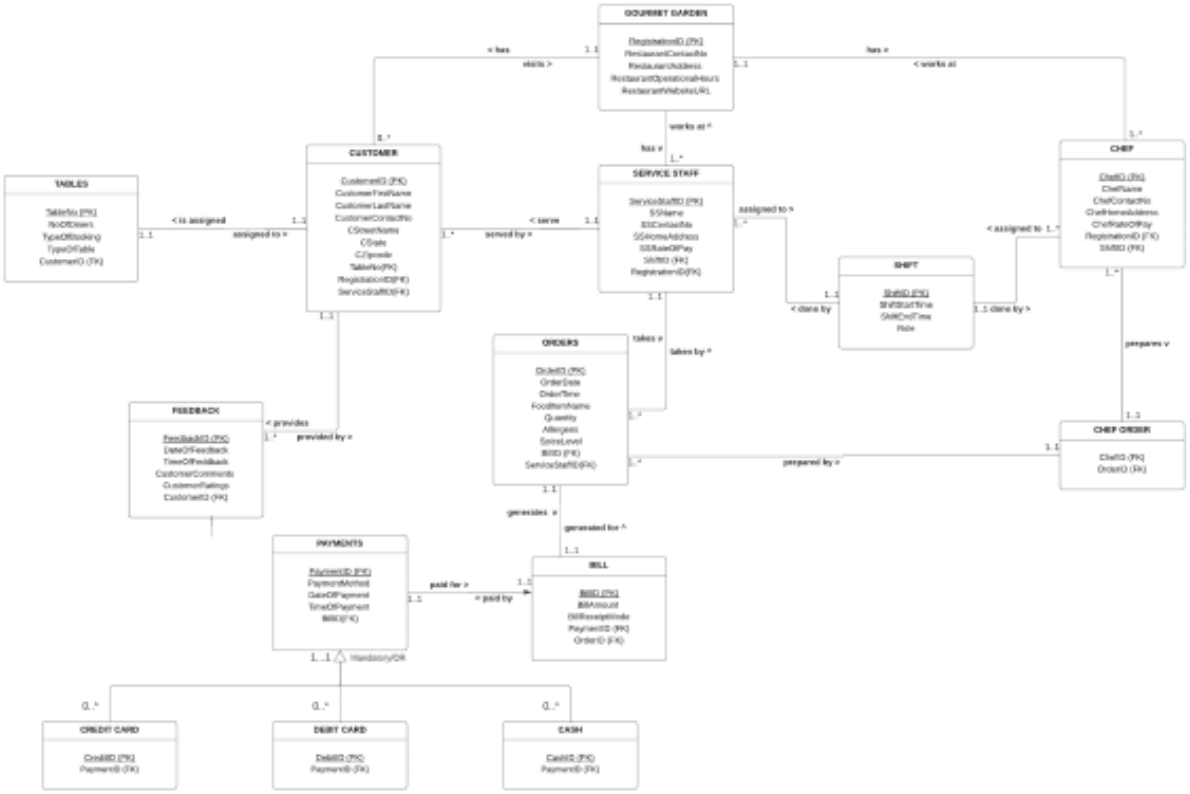
The next step is to Draw the Relational Model. During this step, identifiers in the entities become Key to the relations. One-to-many relationships result in a foreign key being copied from the One side to the Many side of the relationship.

Customer (CustomerID (PK), CustomerFirstName, CustomerLastName, CustomerContactNo, CStreetName, CState, CZipcode, TableNo (FK), RegistrationID (FK), ServiceStaff ID (FK))
GourmetGarden (RegistrationID (PK), RestaurantContactNo, RestaurantAddress, RestaurantOperationalHours, RestaurantWebsiteURL)
Tables (TableNo (PK), NoOfDiners, TypeOfBooking, TypeOfTable, CustomerID (FK))
Feedback (FeedbackID (PK), DateOfFeedback, TimeOfFeedback, CustomerComments, CustomerRatings, CustomerID (FK))
ServiceStaff (ServiceStaffID (PK), SSName, SSContactNo, SSHomeAddress, SSRateOfPay, ShiftID (FK), RegistrationID (FK))
Shift (ShiftID (PK), ShiftStartTime, ShiftEndTime, Role)
Chef (ChefID (PK), ChefName, ChefContactNo, ChefHomeAddress, ChefRateOfPay, RegistrationID (FK), ShiftID (FK))
Chef Order (ChefID (FK), OrderID (FK))
Orders (OrderID (PK), OrderDate, OrderTime, FoodItemName, Quantity, Allergens, SpiceLevel, BillID (FK), ServiceStaffID (FK))
Payments (PaymentID (PK), PaymentMethod, DateOfPayment, TimeOfPayment, BillID (FK))
Bill (BillID (PK), BillAmount, BillReceiptMode, PaymentID (FK), OrderID (FK))
Credit Card (CreditID (PK), PaymentID (FK))
Debit Card (DebitID (PK), PaymentID (FK))
Cash (CashID (PK), PaymentID (FK))

This is the “initial set of relations.”

Commentary:

Primary Keys are shown with the PK designation. Foreign keys are shown with the FK designation.



Screenshot: Our Relational Model Screenshot from Lucidchart

III. Normalization

The next step is to Normalize the Relations. We selected four tables on a random basis.

1. **Tables**

TABLES				
TableNo. (PK)	**NoOfDiners**	**TypeOfBooking**	**TypeOfTable**	**CustomerID (FK)**
T1	4	Reservation	Outdoor Seating	GG0001
T2	2	Reservation	Indoor Seating	GG0002
T3	6	Reservation	Outdoor Seating	GG0003
T4	3	Reservation	Rooftop Seating	GG0004
T5	5	Reservation	Indoor Seating	GG0005
T6	2	Walk-In	Rooftop Seating	GG0012
T7	1	Walk-In	Indoor Seating	GG0010

Primary Key: TableNo.
Solution: The table is in 3NF basis the below working:
1NF: Meets the definition of a relation
2NF: No partial Key dependencies and all non-key attributes are fully functional dependent on 'TableNo.'
3NF: No transitive dependencies

2. **Bill**

BILL				
BillID (PK)	**BillAmount**	**BillReceiptMode**	**PaymentID (FK)**	**OrderID (FK)**
BILL125	$50	Print	P2001	O1001
BILL126	$35	Email	P2002	OI002
BILL127	$20	Print	P2003	OI003
BILL128	$45	Print	P2004	OI004
BILL129	$28	Email	P2005	OI005
BILL121	$25	Print	P2008	OI115

Primary Key: Bill ID
Solution: The table is in 3NF basis the below working:
1NF: BILL(BillID (PK), BillAmount, BillReceiptMode, OrderID (FK), PaymentID (FK)). This meets the definition of a relation.
2NF: No partial key dependencies and all non-key attributes are fully functional dependent on 'Bill ID'.
3NF: No transitive dependencies.

3. Payments

PAYMENTS				
PaymentID (PK)	**PaymentMethod**	**DateOfPayment**	**TimeOfPayment**	**BillID (FK)**
P2001	Cash	2024-05-02	12:45 PM	BILL125
P2002	Debit Card	2024-05-02	1:30 PM	BILL126
P2003	Cash	2024-05-02	2:15 PM	BILL127
P2004	Credit Card	2024-05-02	7:30 PM	BILL128
P2005	Credit Card	2024-05-02	7:45 PM	BILL129
P2033	Debit Card	2024-05-04	8:00 PM	BILL551
P2056	Cash	2024-05-04	6:30 PM	BILL665

Primary Key: PaymentID
Solution: The table is in 3NF basis the below working:
1NF: This meets the definition of a relation because PaymentID is the unique identifier for each row.
2NF: There are no partial key dependencies because all attributes in the table are fully dependent on PaymentID.
3NF: There are no transitive dependencies.

4. Orders

ORDERS								
OrderID (PK)	**OrderDate**	**OrderTime**	**FoodItemName**	**Quantity**	**Allergens**	**SpiceLevel**	**BillID (FK)**	**ServiceStaffID (FK)**
OI001	2024-05-02	12:30 PM	Pad Thai	2	None	Medium	BILL125	SS1
OI002	2024-05-02	1:15 PM	Sushi Rolls	1	None	Mild	BILL126	SS2
OI003	2024-05-02	2:00 PM	Pho	1	None	None	BILL127	SS3
OI004	2024-05-02	3:00 PM	Dim Sum	2	Gluten	Spicy	BILL128	SS4
OI005	2024-05-02	4:30 PM	Ramen	1	None	Medium	BILL129	SS5
OI010	2024-05-03	3:00 PM	Sushi Rolls	2	None	Mild	BILL256	SS11
OI023	2024-05-03	5:00 PM	Miso Soup	3	Gluten	Medium	BILL290	SS10

Primary Key: OrderID
Solution: The table is in 3NF basis the below working:
1NF: Meets the definition of a relation
2NF: No partial Key dependencies and all non-key attributes are fully functionally dependent on 'OrderID'
3NF: No transitive dependencies

IV. DDL Statements - Structured Query Language (SQL) to Create the Schema

Create a table in the database for each of the relations in the final set of relations.

The following SQL code creates the tables and adds the PRIMARY KEY constraint to each one:

```
CREATE TABLE Tables(
TableNo Varchar(25) NOT NULL,
NoOfDiners Numeric(10) NOT NULL,
TypeOfBooking Varchar(30) NOT NULL,
TypeOfTable Varchar(40) NOT NULL,
CustomerID Varchar(25) NOT NULL,
CONSTRAINT Table_PK PRIMARY KEY (TableNo)
);

CREATE TABLE Shift (
ShiftID Varchar(25) NOT NULL,
ShiftStartTime Varchar(25) NOT NULL,
ShiftEndTime Varchar(25) NOT NULL,
Role Varchar(60) NOT NULL,
CONSTRAINT Shift1_PK PRIMARY KEY (ShiftID)

CREATE TABLE Payments(
PaymentID Varchar(25) NOT NULL,
PaymentMethod Varchar(25) NOT NULL,
DateOfPayment Varchar(30) NOT NULL,
TimeOfPayment Varchar(30) NOT NULL,
BillID Varchar(25) NOT NULL,
CONSTRAINT Payments_PK PRIMARY KEY (PaymentID)
);

CREATE TABLE GourmetGarden (
RegistrationID Varchar(10) NOT NULL,
RestaurantContactNo Numeric(10) NOT NULL,
RestaurantAddress Varchar(35) NOT NULL,
RestaurantOperationalHours Varchar(45) NOT NULL,
RestaurantWebsiteURL Varchar(45) NOT NULL,
CONSTRAINT Table_PK PRIMARY KEY (RegistrationID)
);

CREATE TABLE Bill (
BillID Varchar(25) NOT NULL,
BillAmount Varchar(25) NOT NULL,
```

```
BillReceiptMode Varchar(25) NOT NULL,
PaymentID Varchar(25) NOT NULL,
OrderID Varchar(25) NOT NULL,
CONSTRAINT Bill1_PK PRIMARY KEY (BillID)
);

CREATE TABLE Feedback(
FeedbackID Varchar(25) NOT NULL,
DateOfFeedback Varchar(25) NOT NULL,
TimeOfFeedback Varchar(25) NOT NULL,
CustomerComments Varchar(60) NOT NULL,
CustomerRatings Varchar(10) NOT NULL,
CustomerID Varchar(25) NOT NULL,
CONSTRAINT Feedback_PK PRIMARY KEY (FeedbackID),
CONSTRAINT Cus_FK FOREIGN KEY (CustomerID)
REFERENCES Customer (CustomerID)
ON DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE Orders(
OrderID Varchar(25) NOT NULL,
OrderDate Varchar(25) NOT NULL,
OrderTime Varchar(25) NOT NULL,
FoodItemName Varchar(25) NOT NULL,
Quantity Varchar(10) NOT NULL,
Allergens Varchar(45) NOT NULL,
SpiceLevel Varchar(25) NOT NULL,
BillID Varchar(25) NOT NULL,
ServiceStaffID Varchar(25) NOT NULL,
CONSTRAINT Order_PK PRIMARY KEY (OrderID),
CONSTRAINT Ser_FK FOREIGN KEY (ServiceStaffID)
REFERENCES ServiceStaff(ServiceStaffID)
ON DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE DebitCard (
DebitID Varchar(25) NOT NULL,
PaymentID Varchar(25) NOT NULL,
CONSTRAINT DC_PK PRIMARY KEY (DebitID),
CONSTRAINT Pay1_FK FOREIGN KEY (PaymentID)
REFERENCES Payments (PaymentID)
ON DELETE CASCADE ON UPDATE CASCADE
);
```

```
CREATE TABLE ServiceStaff (
ServiceStaffID Varchar(25) NOT NULL,
SSName Varchar(25) NOT NULL,
SSContactNo Numeric(10) NOT NULL,
SSHomeAddress Varchar(60) NOT NULL,
SSRateOfPay Varchar(10) NOT NULL,
ShiftID Varchar(25) NOT NULL,
RegistrationID Varchar(10) NOT NULL,
CONSTRAINT ServStaff_PK PRIMARY KEY (ServiceStaffID),
CONSTRAINT SS1_FK FOREIGN KEY (ShiftID)
REFERENCES SHIFT (ShiftID)
ON DELETE CASCADE ON UPDATE CASCADE,
CONSTRAINT SS2_FK FOREIGN KEY (RegistrationID)
REFERENCES GourmetGarden(RegistrationID)
ON DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE CreditCard (
CreditID Varchar(25) NOT NULL,
PaymentID Varchar(25) NOT NULL,
CONSTRAINT CC_PK PRIMARY KEY (CreditID),
CONSTRAINT Pay_FK FOREIGN KEY (PaymentID)
REFERENCES Payments (PaymentID)
ON DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE Cash (
CashID Varchar(25) NOT NULL,
PaymentID Varchar(25) NOT NULL,
CONSTRAINT Cash_PK PRIMARY KEY (CashID),
CONSTRAINT Pay2_FK FOREIGN KEY (PaymentID)
REFERENCES Payments (PaymentID)
ON DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE Customer(
CustomerID Varchar(25) NOT NULL,
CustomerFirstName Varchar(45) NOT NULL,
CustomerLastName Varchar(45) NOT NULL,
CustomerContactNo Numeric(10) NOT NULL,
CStreetName Varchar(45) NOT NULL,
CState Varchar(45) NOT NULL,
CZipcode Numeric(5) NOT NULL,
```

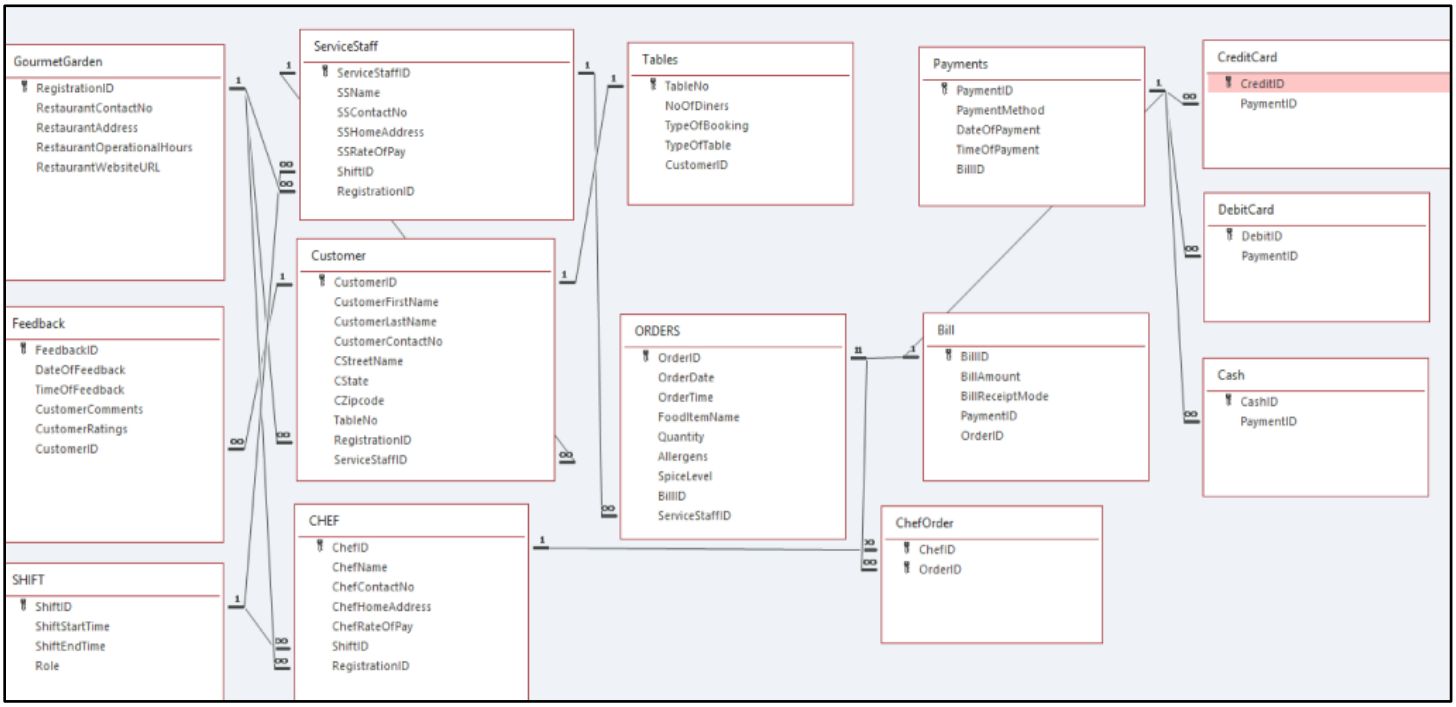
```
TableNo Varchar(25) NOT NULL,
RegistrationID  Varchar(10) NOT NULL,
ServiceStaffID Varchar(25) NOT NULL,
CONSTRAINT Cust_PK PRIMARY KEY (CustomerID),
CONSTRAINT Reg_FK FOREIGN KEY (RegistrationID)
REFERENCES GourmetGarden (RegistrationID)
ON DELETE CASCADE ON UPDATE CASCADE,
CONSTRAINT Ser1_FK FOREIGN KEY (ServiceStaffID)
REFERENCES ServiceStaff(ServiceStaffID)
ON DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE Chef(
ChefID Varchar(10) NOT NULL,
ChefName Varchar(45) NOT NULL,
ChefContactNo Numeric(10) NOT NULL,
ChefHomeAddress Varchar(45) NOT NULL,
ChefRateOfPay Varchar(15) NOT NULL,
ShiftID Varchar(25) NOT NULL,
RegistrationID Varchar(10) NOT NULL,
CONSTRAINT Cust_PK PRIMARY KEY (ChefID),
CONSTRAINT S_FK FOREIGN KEY (ShiftID)
REFERENCES SHIFT (ShiftID)
ON DELETE CASCADE ON UPDATE CASCADE,
CONSTRAINT S1_FK FOREIGN KEY (RegistrationID)
REFERENCES GourmetGarden(RegistrationID)
ON DELETE CASCADE ON UPDATE CASCADE
);

Create TABLE ChefOrder (
ChefID Varchar(10) NOT NULL,
OrderID Varchar(25) NOT NULL,
CONSTRAINT ChefOrder_PK PRIMARY KEY (ChefID,OrderID),
CONSTRAINT CO1_FK FOREIGN KEY (ChefID)
REFERENCES CHEF (ChefID)
ON DELETE CASCADE ON UPDATE CASCADE,
CONSTRAINT CO2_FK FOREIGN KEY (OrderID)
REFERENCES ORDERS (OrderID)
ON DELETE CASCADE ON UPDATE CASCADE
);

ALTER table Feedback DROP TimeOfFeedback

ALTER TABLE Shift ADD Notes Text(50) NOT NULL;
```



Screenshot: Relationship view of our Database

Commentary:

The relationship view above shows all the multiplicities and one to one relations as well. However, post using the DML queries in our database, since we had not mentioned the FK's in one to one relationships as discussed in the class with the Professor (as the data was not getting inserted otherwise), there is slight change in the relationship view of our database in MS Access i.e. one to one relationship is enforced without Referential Integrity.

V. DML Statements - Using DML Statements to interact with our database

INSERT INTO GourmetGarden (RegistrationID, RestaurantContactNo, RestaurantAddress, RestaurantOperationalHours, RestaurantWebsiteURL)
VALUES ('GG56743125', 1234567890, '123 Main St, Elm St., NY, 12345', '11:00 AM - 11:00 PM', 'https://www.gourmetgarden.com');

INSERT INTO Shift (ShiftID, ShiftStartTime, ShiftEndTime, Role, Notes)
VALUES ('CH1', '10:00AM', '5:00 PM', 'Morning Chef', 'None');

INSERT INTO Chef (ChefID, ChefName, ChefContactNo, ChefHomeAddress, ChefRateOfPay, ShiftID, RegistrationID)
VALUES ('C102', 'Aurora Stone', '9876543210', '456 Park Avenue, Brooklyn, NY 11201', '$30', 'CH2', 'GG56743125');

INSERT INTO Chef (ChefID, ChefName, ChefContactNo, ChefHomeAddress, ChefRateOfPay, ShiftID, RegistrationID)
VALUES ('C101', 'Xavier Blackwood', '5637897890', '123 Broadway, New York, NY 10001', '$25', 'CH1', 'GG56743125');

Query 1: Using SELECT and FROM Framework

```
SELECT RestaurantWebsiteURL
FROM GourmetGarden;
```

Query 2: Using SELECT, FROM and WHERE Framework

```
SELECT *
FROM Orders
WHERE FoodItemName='Dim Sum';
```

Query 3: Using SELECT Statement

```
SELECT FoodItemName, Quantity, Allergens, SpiceLevel
FROM Orders;
```

Query 4: Using DISTINCT

```
SELECT DISTINCT ServiceStaffID
FROM Orders;
```

Query 5: Using Calculated Fields

```
SELECT SSName, SSContactNo, SSRateOfPay, SSRateOfPay*6 AS SSPerDayRate
FROM ServiceStaff;
```

Query 6: Using Comparison Search Condition

```
SELECT CustomerFirstName, CustomerLastName, CustomerContactNo, CState, CZipcode
FROM  Customer
WHERE CState='NY' OR CState='PA';
```

Query 7: Using Range Search Condition

```
SELECT TableNo,NoOfDiners,CustomerID
FROM Tables
WHERE NoOfDiners>=5  AND   NoOfDiners<=8;
```

Query 8: Using Compound Comparison Search Condition

```
SELECT ShiftID, Role
FROM Shift
WHERE Role='Senior Service Staff' OR Role='Senior Chef';
```

Query 9: Using Pattern Matching

```
SELECT CustomerComments,CustomerRatings,CustomerID
FROM Feedback
WHERE CustomerComments LIKE '%excellent%';
```

Query 10: Using Ordering

```
SELECT BillID,BillAmount
FROM Bill
ORDER BY BillAmount DESC;
```

Query 11: Using Count

```
SELECT COUNT(* ) AS MyCount
FROM Shift
WHERE ShiftStartTime='11:00AM';
```

Query 12: Using Min, Max & Average

```
SELECT MIN(SSRateOfPay) AS MinRate,
MAX(SSRateOfPay) AS MaxRate,
AVG(SSRateOfPay) AS AvgRate
FROM ServiceStaff;
```

Query 13: Using Group By

```
SELECT BillReceiptMode, COUNT(*) AS NumberOfBills, SUM(BillAmount) AS TotalRevenue
FROM Bill
GROUP BY BillReceiptMode;
```

Query 14: Using Group By & Order By

```
SELECT TableNo, COUNT(TableNo) AS NumberOfTables
FROM Customer
GROUP BY TableNo
ORDER BY TableNo;
```

Query 15: Using Having

```
SELECT ServiceStaffID AS SSID, COUNT(*) AS NumberOfOrders
FROM Orders
GROUP BY ServiceStaffID
HAVING COUNT(*) > 1
ORDER BY ServiceStaffID;
```

Query 16: Using Join

```
SELECT Customer.CustomerID, Customer.ServiceStaffID, Customer.CustomerFirstName,
Customer.CustomerLastName
FROM Customer, ServiceStaff
WHERE Customer.ServiceStaffID = ServiceStaff.ServiceStaffID
AND Cstate='NJ';
```

Query 17: Using Subquery

```
SELECT BillID, BillAmount, BillReceiptMode, PaymentID
FROM Bill
WHERE BillID IN
(SELECT BillID
FROM Payments
WHERE PaymentID IN
(SELECT paymentID
FROM Bill
WHERE PaymentMethod ='Credit Card'
AND BillAmount >'30'));
```

Query 18: Using Update

```
UPDATE Bill SET Bill.BillAmount = "52"
WHERE ((Bill.BillAmount)="40");
```

Query 19: Using Subquery

```
SELECT Orders.OrderID, Orders.FoodItemName, Orders.Allergens, Orders.SpiceLevel
FROM Orders
WHERE OrderID IN
(SELECT OrderID
FROM Bill
WHERE SpiceLevel='Spicy');
```

Query 20: Using Join

```
SELECT Orders.ServiceStaffID, Orders.FoodItemName, Orders.Allergens, Orders.SpiceLevel
FROM Orders, ServiceStaff
WHERE Orders.ServiceStaffID=ServiceStaff.ServiceStaffID
AND FoodItemName='Dim Sum'
AND Allergens='None';
```

Query 21: Using Delete

```
DELETE *
FROM Customer
WHERE CustomerFirstName='Olivia';
```

VI. Group Meeting Log Sheets

Log Sheet 1:

Date: 17th - 21st February, 2024
Meeting Agenda: Finalizing a topic.
Participants: Krishi, Payal, Sadhvi, Sidharth

Accomplishments: We initiated a brainstorming session to determine the topic for our project. Each team member proposed two topics of interest using communication tools such as a WhatsApp group and a collaborative document on Google Drive. Some suggested topics included Library Management System, e-commerce website Management System, Learning Management System, Rental Finder, and Blood Donation Management System. After allowing a week for discussion and consideration, we collectively decided on the Restaurant Management System as our project's focus. This choice resonated with all team members and was highly relevant to the real world.

Challenges and Solutions: Initially, we opted for the Rental Finder Management System, but as we delved into research and began our work, we discovered its extensive scope exceeded our limited knowledge. Our progress on the topic was hindered, and we lacked enthusiasm. Following another meeting, we collectively agreed that developing a Restaurant Management System was a more appealing choice. We communicated our decision to our professor via email and discussed it with him. He approved the idea, provided valuable feedback, and suggested additional areas to consider in further developing our model.

Log Sheet 2:

Date: 21st March & 28th March, 2024
Meeting Agenda: Identifying broad entities and relations
Participants: Krishi, Payal, Sadhvi

Accomplishments: We collaborated to identify the main entities and attributes to include in our relational model. Additionally, we established several assumptions to simplify the understanding of multiplicities. We also determined the primary keys (PKs) and foreign keys (FKs) for our model. Simultaneously, we crafted a project description to provide clarity on the nature of our business. We constructed the entire relational model on LucidChart.

Challenges and Solutions: While working on the relational model, we encountered complex issues and opted to simplify it. This involved selecting only the most pertinent attributes and entities. Additionally, we ensured there were no duplicate attributes, and each was distinctly named. Following simplification, we revisited the drawing board to refine multiplicities, primary keys, foreign keys, and the overall diagram flow.

Log Sheet 3:

Date: 4th April & 25th April, 2024
Meeting Agenda: Normalization & DDL
Participants: Krishi, Payal, Sadhvi, Sidharth

Accomplishments: After consulting with Professor Kline, we determined four tables for normalization: Tables, Bill, Payments & Order. We normalized these tables to the first, second, and third normal forms (1NF, 2NF, and 3NF). Additionally, we drafted all Data Definition Language (DDL) statements and formulated SQL queries and tables to establish a relationship model.

Challenges and Solutions: Our initial Data Definition Language (DDL) model failed to incorporate the relationship of Gourmet Garden with Customers, Service Staff, and Chef, as outlined in our relational model. Consequently, we revised the code, including Gourmet Garden as a foreign key in the Customers, Service Staff, and Chef tables to represent the intended model accurately. We also faced issues while writing DDL statements for 1:1 relations. We discussed this during the lecture with the professor and other classmates from class and learned that we have to update that relation in the diagram flow.

Log Sheet 4:

Date: 13th May, 2024
Meeting Agenda: DML
Participants: Krishi, Payal, Sadhvi

Accomplishments: We continued working on our project by diving deeper into our DML statements and progressing with various queries. This involved refining our data manipulation techniques to ensure accurate and efficient data handling.

Challenges and Solutions: As we crafted these queries, we encountered some issues, such as missing attributes and incorrect usage of insert functions for foreign keys, which required us to redo our work. Despite these challenges, we persevered and successfully corrected our queries, allowing us to move forward with a more robust and well-structured database.

Log Sheet 5:

Date: 14th and 15th May, 2024
Meeting Agenda: DML
Participants: Krishi, Payal, Sadhvi

Accomplishments: We completed sub-queries, join functions, and the formatting of our final document.