RESTAURANT MANAGEMENT DATABASE PROJECT

BY

SHIWANI RAJAGOPALAN

Restaurant Management Database project overview:

This project focusses on all the information that is collected in a restaurant from ordering till billing and maintaining a check on stock of ingredients. Storing of these information will make each and every data organized and will be helpful for the restaurant employees to pull data about anything easily.

This project has been done individually by SHIWANI RAJAGOPALAN

Purpose:

The purpose of this project is to organize data to help people working in a restaurant to be able to track anything in a given time. For example, information like number of orders that have been given out and payments made can benefit the employees in situation where they have to calculate the revenue made by the restaurant and compare revenues with other days to be able to get an idea of whether if there has been profit or not.

End User:

Employees of the restaurant who administer this work.

Data:

There are existing data to populate in the table but I will be manually populating certain entries using insert statements.

RDBMS Software:

MySQL

Step 1: Scenario and Database Requirement

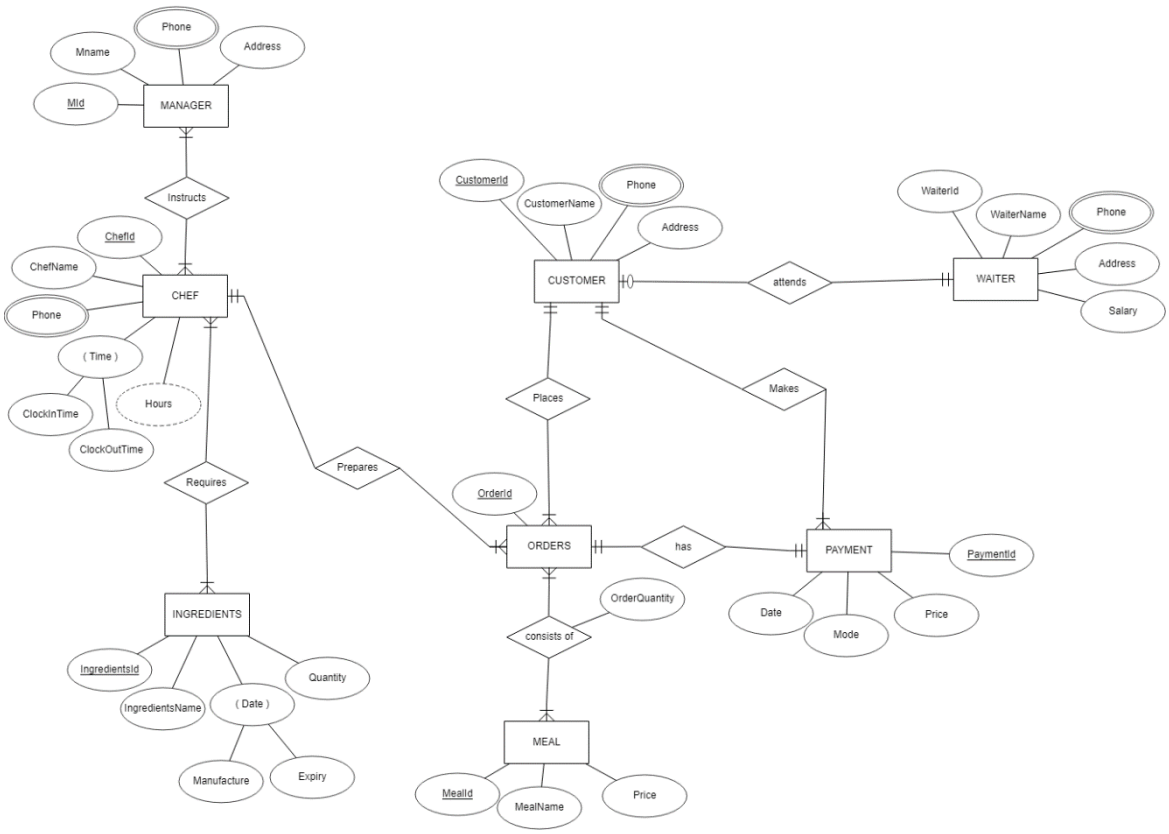
Restaurant Management System Database project focusses on maintaining records of information related customers who are ordering a meal in the restaurant, order details of the customer, transaction information, employee related like chef, waiters and managers who are working in the restaurant. In addition to all of this, record of ingredients required to make dishes are also kept track of so as to know which ingredients are going out of stock.

The main purpose of creating this database system is to help the restaurant keep track of which meals are available at any particular time, to check if the ingredients are available or not, and to have information related to the customer who has placed the order so as to map the order to the customer who has asked for it.

The Restaurant Management System will keep data of the following:

1) Every Manager has Unique manager id, name, phone number and address.
2) Each chef in the restaurant has a unique chef id, name, phone number, clock in and clock out times, and a derived attribute hours calculated from the total number of hours the chef has worked from clock in time to clock out time.
3) Each Ingredients in the pantry has a unique ingredient id, name, date which comprises of the date of manufacture and expiry date, and quantity of those ingredients in stock.
4) Each customer who comes to the restaurants has a unique customer id, name, phone number, and address.
5) Each order taken from the customer gets a unique order id to identify the orders made by customers and quantity of that particular meal.
6) Each meal has a unique meal id, name and price.
7) Every waiter working in the restaurants has a unique waiter id, name, phone number, address and salary information.
8) Each payment for the particular order has a payment id, price, mode of transaction, and date.
9) Each manager must instruct at least 1 chef in the restaurant and each chef must have at least 1 manager to instruct them.
10) Each chef requires at least 1 ingredient to make a dish and each ingredient must be used by at 1 or more chefs.
11) Each chef must have at least 1 order to prepare a dish and each order must be handled by only 1 chef.
12) Each customer must place at least 1 order and each order can have only 1 customer ordering it.
13) Each order must have at least 1 meal listed from menu and one of the meal must be present in at least one order.
14) Each order must have only 1 payment and each payment has only 1 order.
15) Each customer makes a payment for the order that has been placed and they have to make at least 1 payment. Each payment must have at least 1 order to process the payment.
16) Each customer is attended by only 1 waiter and each waiter can attend 1 customer or no customers at all.

Step 2: Enhanced Entity Relationship Diagram (EERD)



The above picture shows the ER-diagram for the Restaurant Management System Database project.

About each entities:

Manager:
Managers in every restaurants are required to provide personal information about them in-order for the restaurant to contact them if needed, so why I have taken Manager Name, Address, and Phone number as part of the personal information part. Manager Id is usually given to differentiate between multiple managers handling a restaurant and uniquely identify them.

Chef:
Every chef has been given a unique Chef Id to distinguish them and also to keep a record of unique personal information about each chef. Apart from Chef Id, I have considered name, address, and phone as part of collecting information about them just in case if one of the chefs do not show up. Clock in and Clock-out time are monitored for each chef to keep track of how many hours they are working and also to ensure whether if the chefs are working for allotted time.

Ingredients:
Every ingredient gets a unique ingredient id to identify them and track information about them when required for the kitchen. Each ingredient has a name, manufacturing date and expiry date to keep track of which ingredients are fresh, ingredients that need to be trashed, and also the ingredients that are soon to expire to use them soon. Quantity keeps track how many ingredients are currently there in the restaurant and it will help the managers and chefs to keep track of ingredients that may run out soon.

Customers:
Basic information is taken about the customers who come to the restaurant like their name, address, phone number to keep a record of the customers just in case when the restaurant employees need to pull some information about them like missing orders, or customers who have frequently visited the restaurant to give them some discount occasionally to make them keep coming to the restaurant.

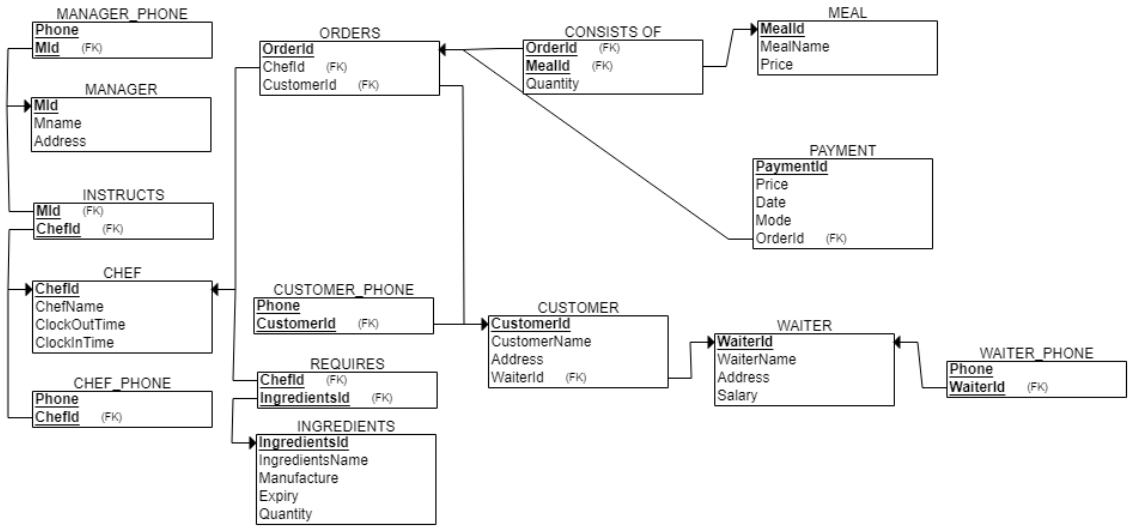
Orders:
Order keeps track of all the orders that has been placed in the restaurant which has a unique order Id so that if there are any missing orders, or if customer wish to opt for a delivery mode, it'll be easier if a record of orders is maintained. Additionally, the employees can also keep track of number of orders that have been placed in the restaurant to get an idea of either how much people like the food in the restaurant, or if there is a dip in sales then they can use order details to find the food items that are not preferred by customers.

Meal:
Meal entity is the menu for the restaurant where in each meal has a unique meal id, name and also the price for each of these meals. This entity can be used by employees in a restaurant to get an overall idea of all the menu options that the restaurant is providing. Any order made in the restaurant will carry a meal Id so that the chefs can understand the dish that they have to make for the customers.

Payment:
Every order that has been made in the restaurant will have a payment and the payment is uniquely recognized by a payment id. Additionally, information like mode of payment, date and price are also considered as each payment will contain a price and this will be especially very useful to find how many transactions have been made in the restaurant. This can give a good picture of sales in the restaurant depending on the amount that has been collected in the particular day.

Waiter:
Each waiter gets a waiter Id to uniquely identify them and information like name, address, salary and phone number of the waiters are taken. This is to ensure that at any point if a waiter does not show up for work, having their personal information will be useful to contact them. This information is also useful in cases when the managers want to know which waiters attended the customers if in case a complaint about the waiter has been made.

Step 3: Relational Schema



Step 4: Normalization

Most of the tables are in first normal form as there are multi-valued attributes like phone number in tables manager, customer, chef and waiter where separate records for each of them having multi phone numbers are created. Other than phone numbers, there are no other attributes that can possibly be multi-valued.

I have not used Denormalization for this project because there is a likely chance of ending up with Data Anomalies since I have incorporated phone number as a multi-valued attribute in tables like customer, manager, chef, and waiters, if any change was made to one of the attributes then other attributes that have been duplicated for different phone numbers must also be updated. There is a chance that for the same person, we may have records of different information so why I decided to not Denormalize my data to avoid anomalies.

Another reason for not Denormalizing the data is because the data I currently hold for the restaurant may become inconsistent due to frequent updates and there is a chance that my data may not consistent for a customer, or if an order information is updated frequently then getting information about orders and about its details may not be possible as there could be a chance I may retrieve a data that has either not been updated, or may have been updated only in place leaving out other entries that need to be updated as well.

Step 5: Create and Populate DB

I have used the data that I created on my own to create and insert into the tables. I have picked close to 5-15 queries for each of the tables created.

There are in total 15 tables that were created for this project and each table contains at least one candidate key. Below are 4 sample tables taken out of the 15 to explain how the attributes were created and the datatype of these attributes.

For the chef table, the following datatype was considered:
ChefId - Integer
ChefName -Varchar(with limit 255)
ClockInTime -TIME(hh:mm:ss)
ClockOutTime -TIME(hh:mm:ss)

ChefID was considered as a primary key

SQL Statement:

```
CREATE TABLE Chef(
   ChefId int NOT NULL,
   ChefName varchar(255) NOT NULL,
   ClockInTime TIME NOT NULL,
   ClockOutTime TIME NOT NULL,
   PRIMARY KEY (ChefId)
);
```

For manager, the following datatype was used:
MId -Integer
MName -Varchar
Address -Varchar

Mid was taken as primary key

SQL Statement:

```
CREATE TABLE Manager(
   MId int NOT NULL,
   MName varchar(255) NOT NULL,
   Address varchar(255),
   PRIMARY KEY (MId)
);
```

For meal table, the datatypes are:
MealId -Integer
MealName -Varchar
Price -Integer

MealId was taken as primary key

SQL Statement:

```
CREATE TABLE Meal(
   MealId int NOT NULL,
   MealName varchar(255) NOT NULL,
   Price int NOT NULL,
   PRIMARY KEY (MealId)
);
```

For customer, following was considered:

CustomerId -Integer
CustomerName -Varchar
Address -Varchar

CustomerID was taken as primary key

SQL Statement:

```
CREATE TABLE Customer(
   CustomerId int NOT NULL,
   CustomerName varchar(255) NOT NULL,
   Address varchar(255),
   PRIMARY KEY (CustomerId)
);
```

Step 6: SQL statements:

Query 1: number of days from expiration

SELECT i.IngredientsId,i.IngredientsName,i.Quantity,DATEDIFF(i.ExpiryDate, i.ManufactureDate) AS "Number of days between manufacture and expiry" FROM Ingredients as i;

+ Options

	IngredientsId	IngredientsName	Quantity	Number of days between manufacture and expiry
Edit Copy Delete	234	Hummus	5	130
Edit Copy Delete	235	Marinara Sauce	3	365
Edit Copy Delete	236	Cheddar Cheese	6	118
Edit Copy Delete	237	Mozzarella Cheese	3	82
Edit Copy Delete	238	Canola oil	7	389
Edit Copy Delete	239	Bread	15	9
Edit Copy Delete	240	BlackBean	50	184

Query 2: derived attribute (find hours between clock in and clock out time)

select cf.ChefId,cf.ChefName,TIMEDIFF(cf.ClockOutTime, cf.ClockInTime) AS "Number of Hours worked" FROM Chef as cf;

+ Options

	ChefId	ChefName	Number of Hours worked
Edit Copy Delete	30456	Roxy	08:59:42
Edit Copy Delete	30457	Kaylee	10:00:45
Edit Copy Delete	30458	Bill	09:29:59
Edit Copy Delete	30459	Mark	08:29:30
Edit Copy Delete	30460	Gordon	07:01:00
Edit Copy Delete	30461	Karla	09:58:05

Query 3: get meal name and quantity for the specific order

SELECT ct.OrderId, m.MealName, ct.Quantity FROM ConsistsOf as ct INNER JOIN Meal as m ON ct.MealId=m.MealId ORDER BY ct.OrderId;

+ Options

OrderId ▲ 1	MealName	Quantity
90	Black Bean Sandwich	1
90	Grilled Cheese Sandwich	1
91	Grilled Cheese Sandwich	1
92	Black Bean Panini	1
93	Cheese Panini	2
94	Cheesy Croutons	1
95	Grilled Cheese Sandwich	1
96	Black Bean Panini	3
97	Black Bean Sandwich	1
98	Cheese Panini	2
99	Black Bean Sandwich	1
99	Cheesy Croutons	1
100	Black Bean Panini	1

Query 4: remaining ingredients

SELECT r.IngredientsId,i.IngredientsName,i.Quantity-COUNT(r.IngredientsId) as "remaining" FROM Requires as r INNER JOIN Ingredients as i ON r.IngredientsId=i.IngredientsId GROUP BY r.IngredientsId;

+ Options

IngredientsId	IngredientsName	remaining
234	Hummus	1
235	Marinara Sauce	1
236	Cheddar Cheese	4
237	Mozzarella Cheese	1
238	Canola oil	5
239	Bread	7
240	BlackBean	46

Query 5: find the customers who frequently come to restaurant.

SELECT o.CustomerId,cm.CustomerName,COUNT(o.CustomerId) as "Frequent Customers" FROM Orders as o INNER JOIN Customer as cm ON o.CustomerId=cm.CustomerId GROUP BY o.CustomerId;

CustomerId	CustomerName	Frequent Customers
9123410	Chandler	1
9123411	Pheobe	1
9123412	Gunther	1
9123456	Joey	2
9123457	Rachel	2
9123458	Monica	2
9123459	Ross	2

Query 6: Most liked meal among customers

SELECT cs.MealId,m.MealName FROM ConsistsOf as cs INNER JOIN Meal as m ON cs.MealId=m.MealId GROUP BY cs.MealId HAVING SUM(cs.Quantity)=(SELECT MAX(total) FROM (SELECT SUM(quantity) as total FROM ConsistsOf GROUP BY MealId) as a);

MealId	MealName
3	Black Bean Panini

Query 7: chef names starting with 'k'

SELECT * FROM Chef as cf WHERE cf.ChefName LIKE 'k%';

	ChefId	ChefName	ClockInTime	ClockOutTime
Edit Copy Delete	30457	Kaylee	09:30:00	19:30:45
Edit Copy Delete	30461	Karla	09:02:45	19:00:50

Query 8: How many customers are the waiters handling:

SELECT cm.WaiterId,w.WaiterName,COUNT(cm.CustomerId) as "number of customers handled" FROM Customer as cm INNER JOIN Waiter as w ON cm.WaiterId=w.WaiterId GROUP BY cm.WaiterId;

WaiterId	WaiterName	number of customers handled
20	Alex	1
21	Tam	3
22	Rachel	1
23	London	2

Query 9: customers who made more than 1 order:

SELECT o.CustomerId, cm.CustomerName,cm.Address, COUNT(o.OrderId) FROM Orders as o JOIN Customer as cm ON o.CustomerId=cm.CustomerId GROUP BY o.CustomerId HAVING COUNT(o.OrderId)>1;

+ Options

CustomerId	CustomerName	Address	COUNT(o.OrderId)
9123456	Joey	123, Walnut street, Bloomington, IN 47404	2
9123457	Rachel	1320 E 10th St, Bloomington, IN 47405	2
9123458	Monica	900 E.Seventh Street,Bloomington, IN 47405	2
9123459	Ross	919 E 10th St, Bloomington, IN 47408	2

Query 10: customers who paid more than average of the total collected amount:

SELECT * FROM Payment as p WHERE p.Price>(SELECT AVG(p.Price) FROM Payment as p);

	PaymentId	Price	PaymentDate	PaymentMode	OrderId
Edit Copy Delete	10150	26	2022-05-02	card	90
Edit Copy Delete	10153	23	2022-05-02	card	93
Edit Copy Delete	10156	45	2022-05-02	cash	96
Edit Copy Delete	10158	23	2022-05-03	cash	98
Edit Copy Delete	10159	22	2022-05-03	cash	99

Most of the queries mentioned above are almost closely related but there are few queries that play a major role in boosting business for restaurants like:

1. Query 5 is used to find the customers who come frequently was provided. This query in turn will help restaurant employees to know the customers who visit their restaurant frequently and can offer discounts for their orders in order to retain such frequent customers.
2. Query 6 is another important query where we find the food item that is popular in the restaurant. If managers and chefs get an idea of the food item that is selling a lot then they stock them up in advance in order to avoid running out of supplies for best selling orders. Customers will also be happy if they know that their favourite food item can be ordered without any additional delay due to stock issues.
3. Query 4 can be used to find how much ingredients are remaining after chefs have taken some from the pantry. Keeping track of this will help managers to know before hand on which ingredients are about to get over and can stock them up in advance. This will also reduce the delay in giving out orders.
4. Query 9 can be used to find the customer who have made more than 1 order as this will help the restaurant owners understand whether if customers like having dishes in the restaurant or not. Frequent orders can be interpreted that customers like the dishes and using Query 5 as well we can make sure that customers are satisfied by providing some coupons or discount that they can use to make more additional orders. This will in turn also help in improving overall ratings of the restaurant when customers frequently come to the restaurant.

This project has been created for direct interactions with end-users via an application for employees of the restaurant who want to manage orders, keep track of stock availability, and transactions that happen from the restaurant.

Application:

Application will be a website where all the details will be available to be monitored by the Managers, Chefs, and Waiters to keep track of orders, stock availability and transactions.

Users:
The users are the employees of the restaurant.

Connection to database:
Front end will be a website and backend will be a PHP script that will update or add new records into the tables in the database.