Project Design Report

Pick Ur Plate

Team Venkata

Satyanarayana Vinay Achanta[A02395874]

Hari Chandana Kotnani[A02396013]

Kotte Venkata Shiva Sai Dheeraj[A02394047]

Table of Contents

Introduction

Pick Ur Plate is a Database developed to provide the facilities for ordering food online that can be delivered to their place from nearby restaurants and bakeries. Pick Ur Plate was inspired by the thought of providing a complete food ordering and delivery solution from the best neighborhood restaurants and bakeries to the urban foodie. we will organize the data of our own exclusive fleet of delivery personnel to pick up orders from restaurants and deliver them to customers. Having our own fleet gives us the flexibility to offer customers a no-minimum order policy on any restaurant for all partner restaurants that we work with.

Generally, we go to a restaurant and order food by choosing items that are listed on the menu card. The main motivation of Pick Ur Plate is to make life easy by ordering food online and to get delivered to our place through our tablets or mobile phones. This document has some specific replacements such as a paper-based menu card being replaced with an electronic format. This also helps by providing a home delivery system, which reduces the effort and time (wastage) of going to restaurants. The ordered items are given to the chefs in the electronic format only, which is confirmed for sure by the customer. Later, some provisions are also provided for the cancellation of the items.

The important aspects of the database are:

- The first aspect deals with menu management (i.e., controlling the list of items that can be ordered by the customer).
- The second aspect deals with order retrieval, this part is taken care of by the restaurant to keep track of all the incoming orders and to update the status of the orders already been processed**.**

Requirements Analysis

Data Requirements

- Customers can have customer_id which uniquely identifies the customer, customer name, phone number, email id, and address.
- Customers get customer support if they face any issue which contains ticket_id, customer_id, order_id, Ticket_status and the issue.
- Customers provide ratings for the orders which contain order_id, customer_id, rating_value, Feedback.
- Customers place an order which contains order_date, order_id, number of items and total price.
- The order has delivery details. delivery details consist of delivery_address, order_id and employee_id. Each employee has a unique empoyee_id, emp_phn_num, and employee_name.
- The payment made for the order contains payment_id, card_details, and mode of payment.
- Sometimes customers may have coupons they need to enter the unique coupon code, coupon_value, and description.
- Order contains food details that consist of food_id, food_name, food_type, food_price and quantity.s
- Orders of each restaurant are managed by the restaurant admin. Admin needs username and password.
- The restaurant is uniquely identified by restaurant id, restaurant name, restaurant address, phone_number and Restaurant_email.
- Menu in the Restaurants has item_id, item_name, item_description, category, item_type and item_price.
- Each restaurant has chef who is uniquely identified by chef_id, chef_name and phone_number.

Functional Requirements

- Functional Requirement 1: Get all the customer support tickets that are active.
- Functional Requirement 2: Details about the number of customers who paid with different payment modes.
- Functional Requirement 3: Get the information about the customer who applied coupons while payment.
- Functional Requirement 4: Get the maximum rating of all the restaurants.
- Functional Requirement 5: At least one item needs to be selected to proceed with the payment.
- Functional Requirement 6: Get the count of issues got from each restaurant.
- Functional Requirement 7: Get the customer details who ordered food from the location example: get the orders placed from the "university pines".
- Functional Requirement 8: Get the order details that were ordered in particular year.
- Functional Requirement 9: Get the chef details who worked for each restaurant.
- Functional Requirement 10: Get the details of the admin who manages each restaurant.

Constraints:

- Rating must be in the range of 1 to 5 which implies bad to great!
- The phone number field should only allow 10 digits.
- The card number should allow only 16 digits.

Trigger:

- Trigger is implemented to get the current days date in orders table automatically.

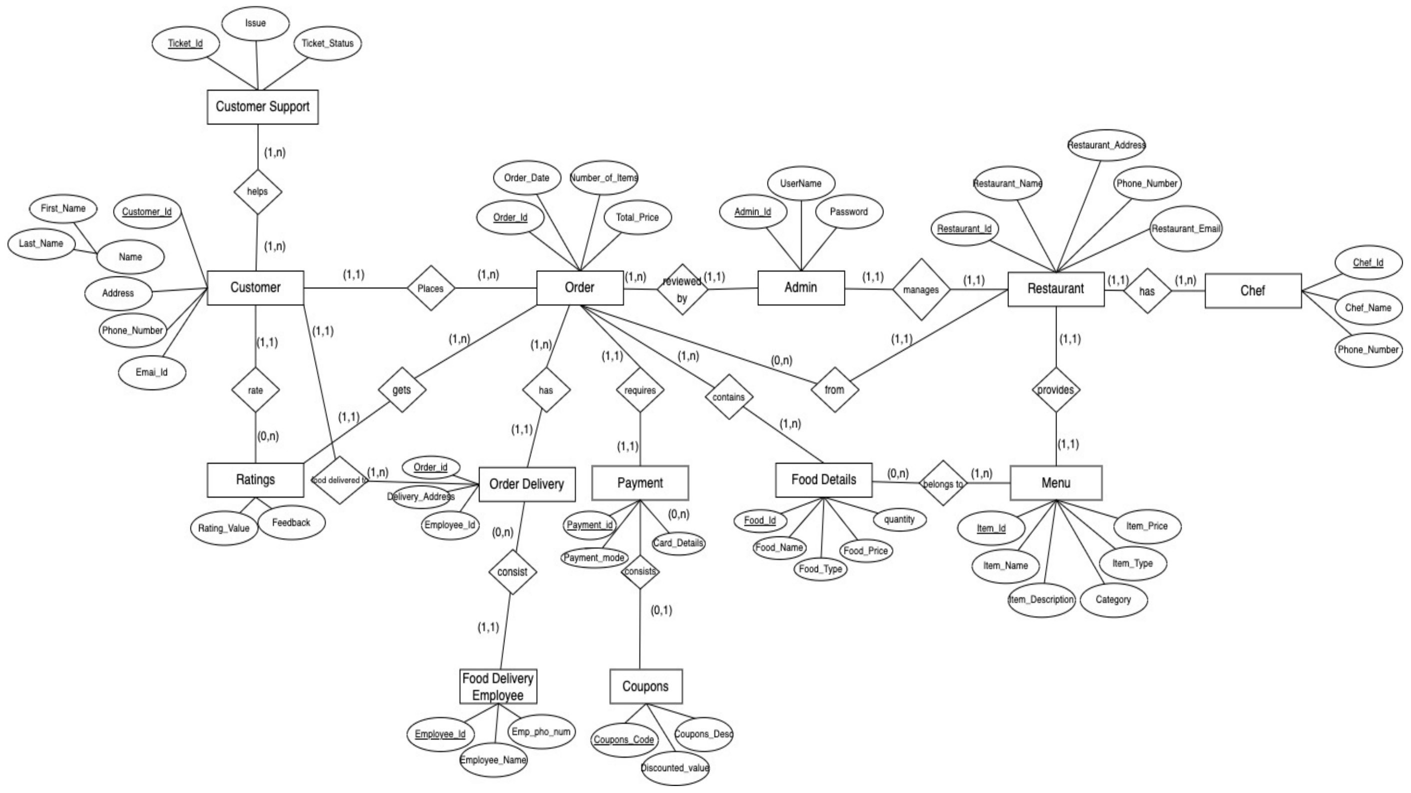
```
DELIMITER $$
CREATE TRIGGER AddDateTrigger
BEFORE INSERT ON Orders
FOR EACH ROW
BEGIN
SET NEW.OrderDate=NOW();
END$$
DELIMITER $$
```

View:

- Customers view on restaurants & their name.

```
CREATE OR REPLACE VIEW customer_restaurant AS
SELECT R.restaurant_name, M.item_name, M.item_price, M.item_description, M.item_type,
M.category FROM restaurant R, menu M where R.restaurant_id = M.restaurant_id;
```

ER Model:

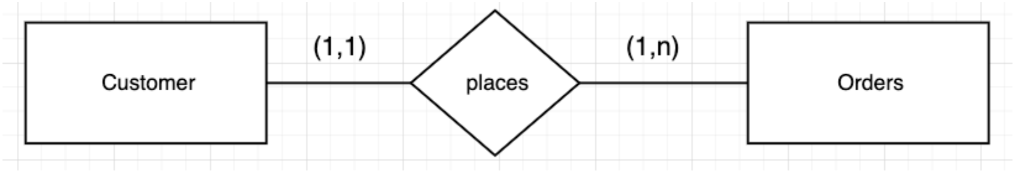

Ticket_Id
Issue
Ticket_Status
Customer Support
(1,n)
helps
(1,n)
First_Name
Customer_Id
Last_Name
Name
Address
Phone_Number
Emai_Id
Customer
(1,1)
Places
(1,n)
Order_Date
Number_of_Items
Order_Id
Total_Price
Order
(1,n)
reviewed by
(1,1)
UserName
Admin_Id
Password
Admin
(1,1)
manages
(1,1)
Restaurant_Address
Restaurant_Name
Phone_Number
Restaurant_Id
Restaurant_Email
Restaurant
(1,1)
has
(1,n)
Chef
Chef_Id
Chef_Name
Phone_Number
(1,1)
rate
(0,n)
(1,1)
(1,n)
gets
(1,1)
(1,n)
has
(1,1)
(1,1)
requires
(1,1)
(1,n)
contains
(1,n)
(0,n)
from
(1,1)
(1,1)
provides
(1,1)
Ratings
Rating_Value
Feedback
food delivered to
(1,n)
Order_id
Delivery_Address
Employee_Id
Order Delivery
(0,n)
consist
(1,1)
Food Delivery Employee
Employee_Id
Emp_pho_num
Employee_Name
Payment
Payment_id
Payment_mode
Card_Details
(0,n)
consists
(0,1)
Coupons
Coupons_Code
Coupons_Desc
Discounted_value
Food Details
Food_Id
Food_Name
Food_Type
Food_Price
quantity
(0,n)
belongs to
(1,n)
Menu
Item_Id
Item_Name
Item_Description
Category
Item_Type
Item_Price

Relational Model:

Relationships

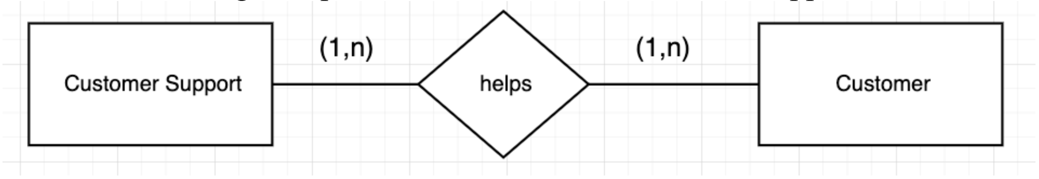
Relationship 1: Places

Cardinalities: Customer can place minimum of one order and maximum of n orders and one order should have minimum of one customer maximum of one customer.



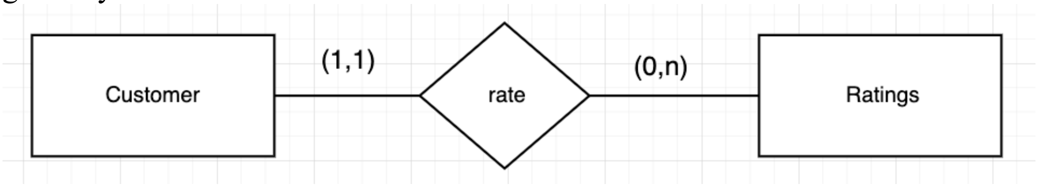
Relationship 2: Helps

Cardinalities: Customer Support can help minimum of one customer and maximum of n customers and one customer can get help from minimum of one customer support maximum of n customer support.



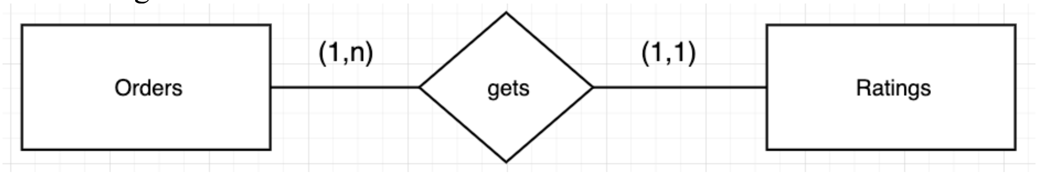
Relationship 3: Rate

Cardinalities: Customer can provide minimum zero rating and maximum of n ratings and one rating is given by have minimum of one customer maximum of one customer.



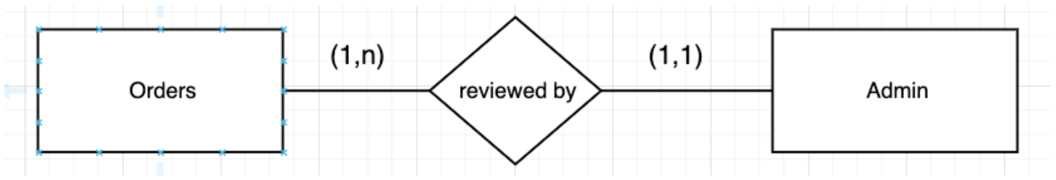
Relationship 4: Gets

Cardinalities: One Order can get minimum of one rating and maximum of one rating and one rating should be given to minimum of one order and maximum of n order.



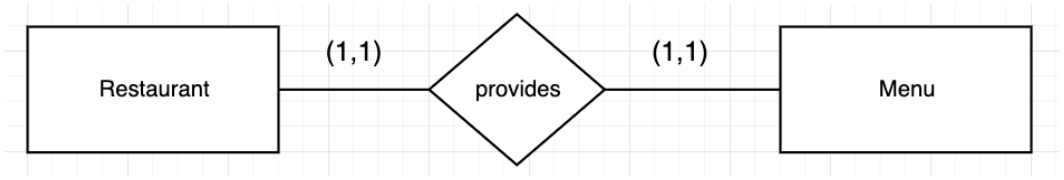
Relationship 5: Reviewed By

Cardinalities: Order can be reviewed by minimum of one admin and maximum of one admin and one admin reviews minimum of one order and maximum of n orders.



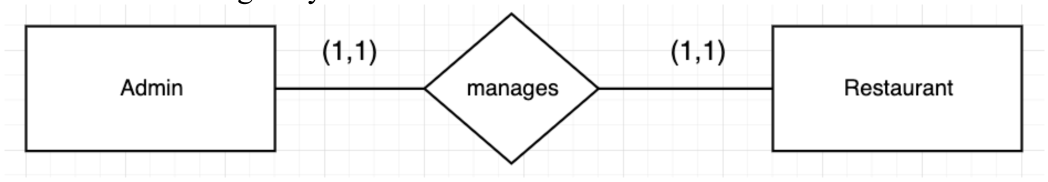
Relationship 6: Provides

Cardinalities: Restaurant can have minimum of one menu and maximum of one menu and one menu is contained in minimum of one restaurant maximum of one restaurant.



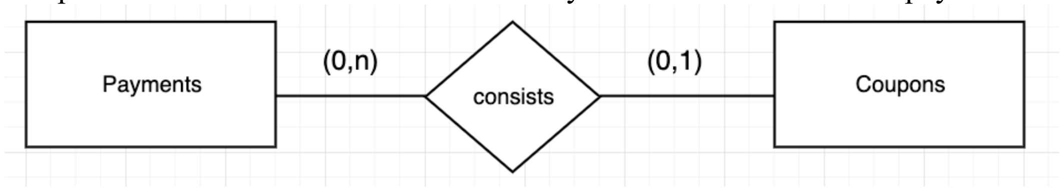
Relationship 7: Manages

Cardinalities: Admin can manage minimum of one restaurant and maximum of one restaurant and one restaurant is managed by minimum of one admin maximum of one admin.



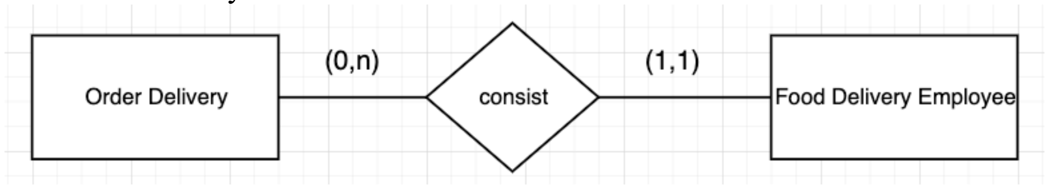
Relationship 8: Consists

Cardinalities: Payments consists of minimum of zero coupons and maximum of one coupon and one Coupon can be used in minimum of zero Payment and maximum of n payments.



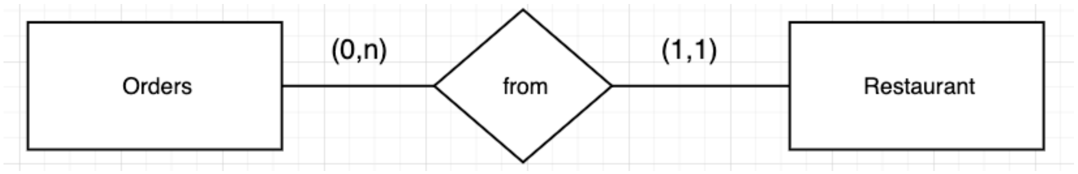
Relationship 9: Consist

Cardinalities: One Order Delivery has minimum of one delivery employees and maximum of one delivery employee and one delivery employee can have minimum of zero order delivery and maximum of n order delivery.



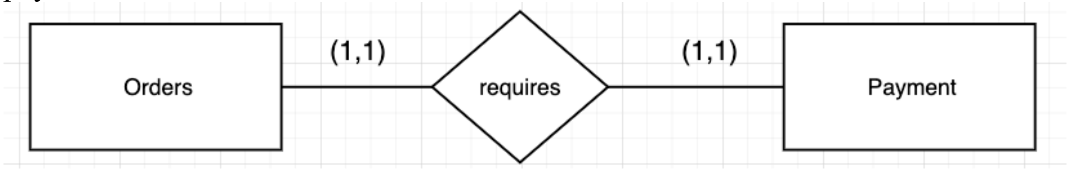
Relationship 10: From

Cardinalities: One order will be from minimum of one restaurant, maximum of one restaurant and restaurant will send minimum of zero order and maximum of n orders



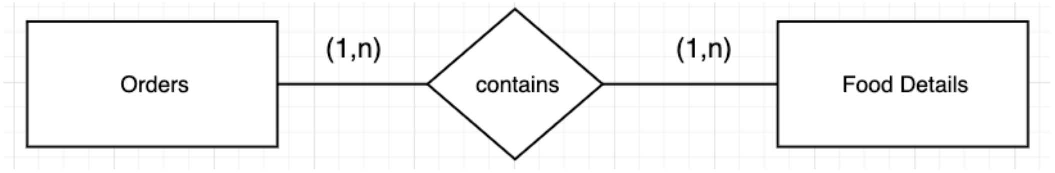
Relationship 11: Requires

Cardinalities: One order requires minimum of one payment and maximum of one payment and one payment should be done for minimum of one order maximum of one order.



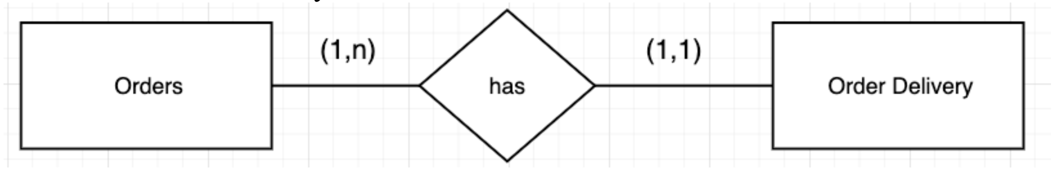
Relationship 12: Contains

Cardinalities: One order contains minimum of one food item details and maximum of n food item details and one food item details can be in minimum of one order maximum of n order.



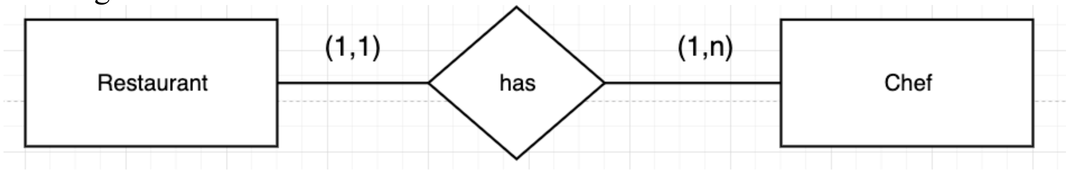
Relationship 13: Has

Cardinalities: One order has minimum of 1 order delivery details and maximum of 1 order delivery details and order delivery can be done for minimum of one order or maximum of n orders.



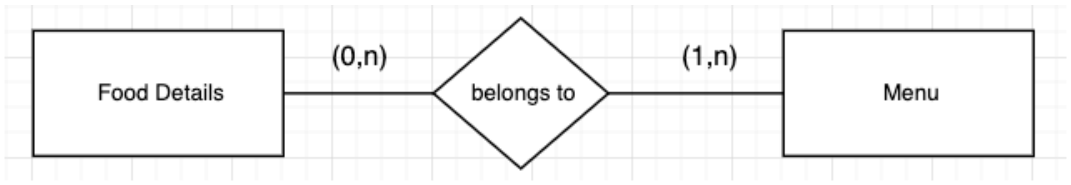
Relationship 14: Has

Cardinalities: One restaurant has minimum of one chef and maximum of n chef's and one chef will be working in minimum of one restaurant and maximum of one restaurant.



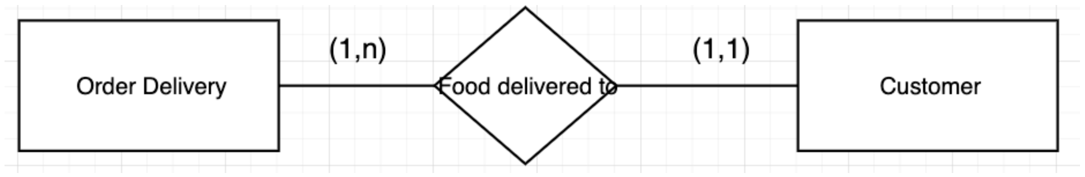
Relationship 15: Belongs to

Cardinalities: One food details can be present in minimum of one menu and maximum of n menus and menu will have minimum of zero Food details and maximum of n food details.



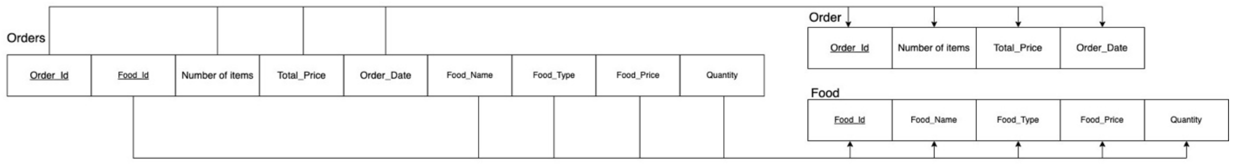
Relationship 16: Food Delivered to

Cardinalities: Customer receives order from minimum of 1 delivery employee and maximum of 1 delivery employee and one Delivery employee should deliver order to minimum of 1 customer maximum of n customer.



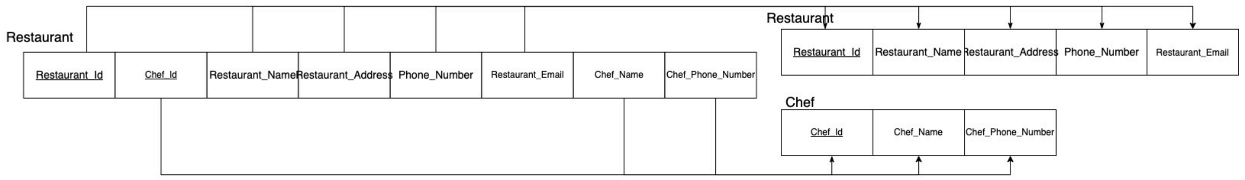
Changes to the relational model to get 3NF:

1. Order Table Normalized:



As you can see the functional dependency from the above diagram violates normalization forms. As we can see that the key is defined by a combination of OrderId and Fid and the respective functional dependency is shown in the image now we see that it violates the 1st NF as well as the 2nd NF(obviously as for 2NF we require 1NF) we see that a customer can order multiple food items and if we keep that with the order entity it will violate the rule for 1st NF and if somebody tells that he wants only one food item to make it to 1st NF(which is wrong) but we will face another problem, which is violating the rule for 2nd NF i.e. partial key dependency and therefore the solution to this creates two separate entity with a relationship between both order and Food.

2. Restaurant Table Normalized:



Each restaurant can have multiple chefs. if one Restaurant has several chefs, all other attributes in the ticket entity are repeated. So, a new table "**Chefs**" has been created with ChefID and Chef Name attributes. And building a relationship between these two entities.

All the other entities follow 3NF (Normal Form) there is no redundant data in each table and every other attribute is dependent on the primary key.

3NF Normalized Relations:

Customer Support

- TicketID (PK),
- Issue,
- Ticket_Status,
- Issue and Ticket_Status are fully dependent on PassengerID

Customer

- Customer_ID (PK)
- First_Name,
- Last_Name,
- Address,
- Phone number,
- Email_id,
- All customer details will be depended by customer_id.

Ratings

- Customer_ID, Order_ID(PK),
- Rating_value,
- Feedback,
- Attributes are completely dependent on these primary keys.

Order

- Order_ID (PK),
- Order date,
- Total price,

- Number of items,
- Admin_ID,
- As each Order_ID is uniquely maintained in our database, all the attributes of the order entity are dependent on the orderID.

Admin:

- Admin_ID (PK),
- Username,
- Password,
- Attributes are uniquely identified for each admin.

Restaurant:

- Restaurant_ID (PK),
- Restaurant_name,
- Restaurant_Address,
- Phone_number,
- Restaurant_Email
- All attributes are dependent on primary key Restaurant_id.

Chef:

- Chef_ID (PK),
- Chef_Name,
- Phone_Number,
- All attributes are dependent on chef_ID.

Menu:

- Item_ID (PK),
- Item_name,
- Item_description,
- Item_type,
- category,
- Item_price,
- All attributes are fully dependent of primary key -Item_Id.

Food details:

- Food_ID (PK),
- FoodName,
- FoodPrice,
- FoodPrice,
- FoodType,
- Quantity,
- All attributes are fully dependent of primary key -Food_Id.

Order Delivery Details:

- Order_ID (PK),
- Delivery address,
- Attributes can be uniquely identified by the ‘Order_id

Food Delivery Employee:

- Emp_ID (PK),
- Employee's name,
- phone number,
- Attributes can be uniquely identified by the ‘Emp_id

Payment:

- Payment_ID (PK),
- Payment mode,
- Card details,
- Attributes are dependent on primary key.

Coupon:

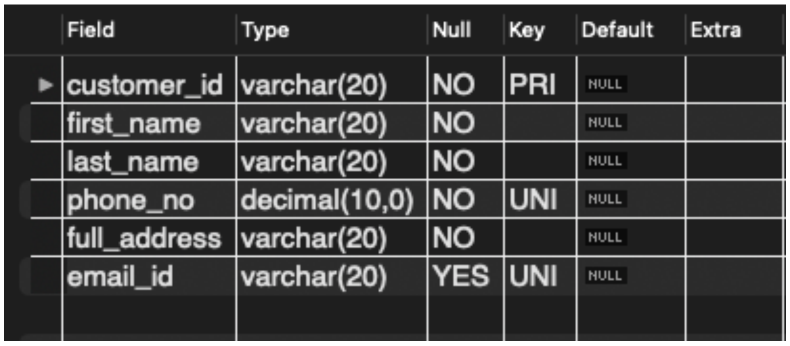
- Coupon_code (PK),
- Coupon_value,
- Coupon_Desc can be obtained from coupon.

Data Dictionary:

Entities and Attributes Description:

Entity 1: Customer

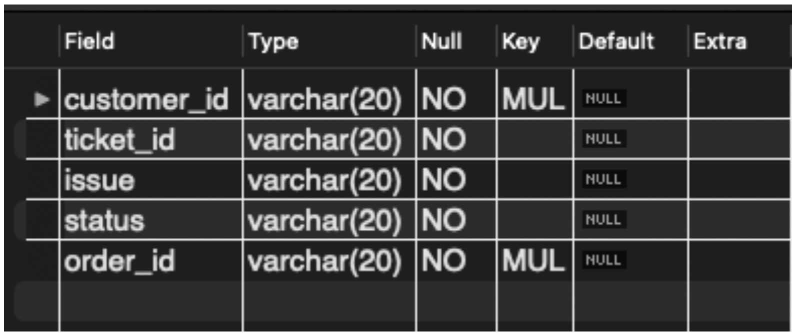
- Customer_Id: Primary key which helps to identify a customer uniquely.
- First_Name: First Name of the Customer.
- Last_Name: Last Name of the Customer.
- Address: Address of the Customer.
- Phone_Number: Phone Number of the Customer.
- Email_Id: Email Id of the Customer.



Field	Type	Null	Key	Default	Extra
customer_id	varchar(20)	NO	PRI	NULL	
first_name	varchar(20)	NO		NULL	
last_name	varchar(20)	NO		NULL	
phone_no	decimal(10,0)	NO	UNI	NULL	
full_address	varchar(20)	NO		NULL	
email_id	varchar(20)	YES	UNI	NULL	

Entity 2: Customer Support

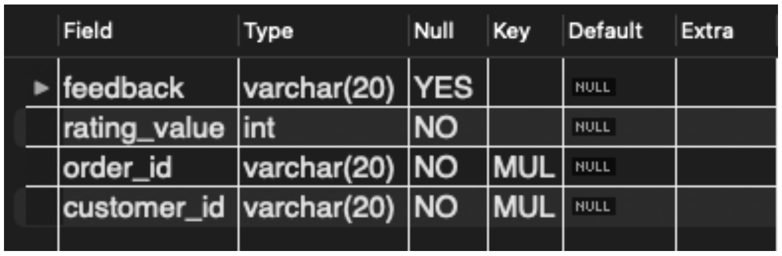
- Ticket_Id: Primary key which helps to identify a ticket raised from customer uniquely.
- Issue: Description of Issue faced by the Customer.
- Ticket_Status: Ticket Status whether the ticket is in progress or resolved or not yet taken up by any customer support.



Field	Type	Null	Key	Default	Extra
customer_id	varchar(20)	NO	MUL	NULL	
ticket_id	varchar(20)	NO		NULL	
issue	varchar(20)	NO		NULL	
status	varchar(20)	NO		NULL	
order_id	varchar(20)	NO	MUL	NULL	

Entity 3: Ratings

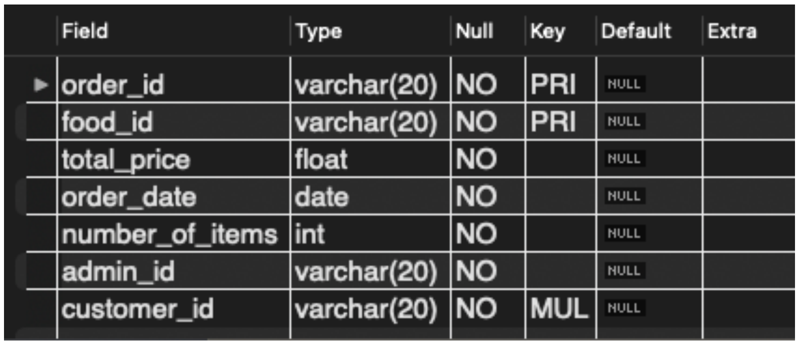
- Rating_Value: Rating Value provided by customer from 1 to 5.
- Feedback: Feedback given by customer.



Field	Type	Null	Key	Default	Extra
feedback	varchar(20)	YES		NULL	
rating_value	int	NO		NULL	
order_id	varchar(20)	NO	MUL	NULL	
customer_id	varchar(20)	NO	MUL	NULL	

Entity 4: Order

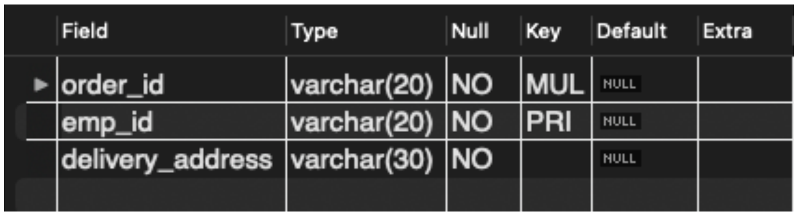
- Order_Id: Primary key which helps to identify an order placed by customer uniquely.
- Order_Date: Date when the order was placed.
- Number_Of_Items: Number of items ordered.
- Total_Price: Total Price of the order placed including all taxes.



Field	Type	Null	Key	Default	Extra
order_id	varchar(20)	NO	PRI	NULL	
food_id	varchar(20)	NO	PRI	NULL	
total_price	float	NO		NULL	
order_date	date	NO		NULL	
number_of_items	int	NO		NULL	
admin_id	varchar(20)	NO		NULL	
customer_id	varchar(20)	NO	MUL	NULL	

Entity 5: Order Delivery

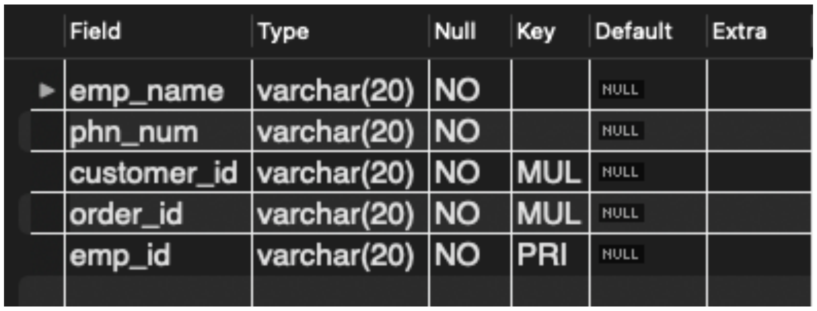
- Order_Id: Primary key which helps to identify an order delivered by customer uniquely.
- Delivery_Address: Delivery Address provided by the customer at the time of order placed.
- Employee_Id: Id of the Employee who delivered that order to the customer.



Field	Type	Null	Key	Default	Extra
order_id	varchar(20)	NO	MUL	NULL	
emp_id	varchar(20)	NO	PRI	NULL	
delivery_address	varchar(30)	NO		NULL	

Entity 6: Food Delivery Employee

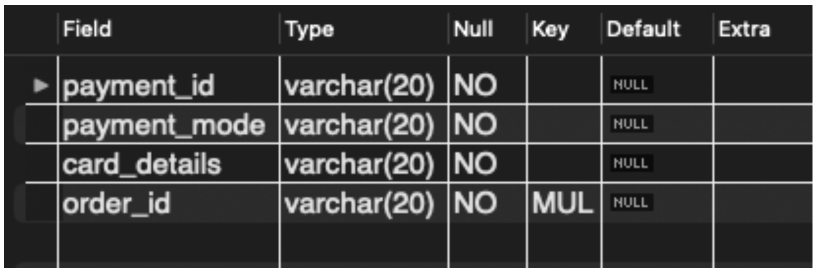
- Employee_Id: Primary key which helps to identify an food delivery employee uniquely.
- Employee_Name: Name of the Food Delivery Employee.
- Emp_pho_num: Food Delivery Employee Phone Number



Field	Type	Null	Key	Default	Extra
emp_name	varchar(20)	NO		NULL	
phn_num	varchar(20)	NO		NULL	
customer_id	varchar(20)	NO	MUL	NULL	
order_id	varchar(20)	NO	MUL	NULL	
emp_id	varchar(20)	NO	PRI	NULL	

Entity 7: Payment

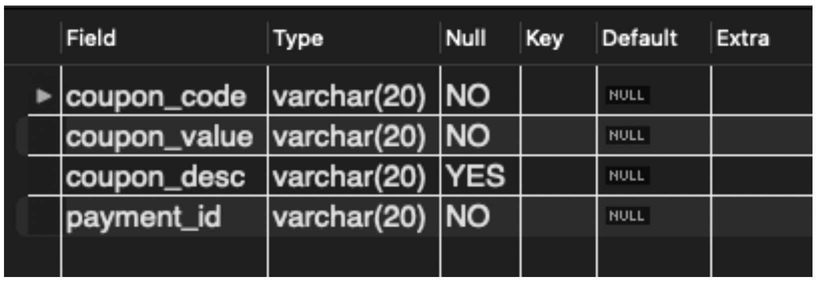
- Payment_Id: Primary key which helps to identify a payment done by customer uniquely.
- Payment_Mode: Mode of payment done by the customer like Credit Card, Debit card.
- Card_Details: Card number provided by the customer.



Field	Type	Null	Key	Default	Extra
payment_id	varchar(20)	NO		NULL	
payment_mode	varchar(20)	NO		NULL	
card_details	varchar(20)	NO		NULL	
order_id	varchar(20)	NO	MUL	NULL	

Entity 8: Coupons

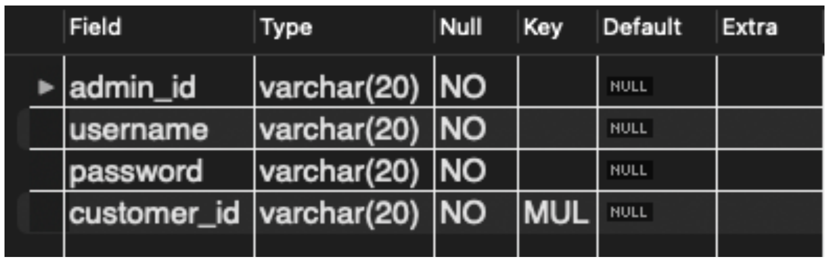
- Coupon_Code: Coupon Code which helps to provide discount for customers.
- Discounted_Value: Percentage of discounted amount for the coupon.
- Coupon_Desc: Description of the coupon code like Thanksgiving, New Year and so on.



Field	Type	Null	Key	Default	Extra
coupon_code	varchar(20)	NO		NULL	
coupon_value	varchar(20)	NO		NULL	
coupon_desc	varchar(20)	YES		NULL	
payment_id	varchar(20)	NO		NULL	

Entity 9: Admin

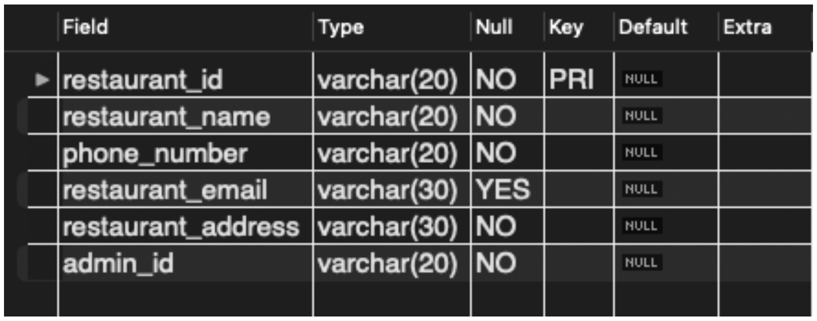
- Admin_Id: Primary key which helps to identify an Admin of Restaurant uniquely.
- UserName: Username of the Admin.
- Password: Password of Admin.



Field	Type	Null	Key	Default	Extra
admin_id	varchar(20)	NO		NULL	
username	varchar(20)	NO		NULL	
password	varchar(20)	NO		NULL	
customer_id	varchar(20)	NO	MUL	NULL	

Entity 10: Restaurant

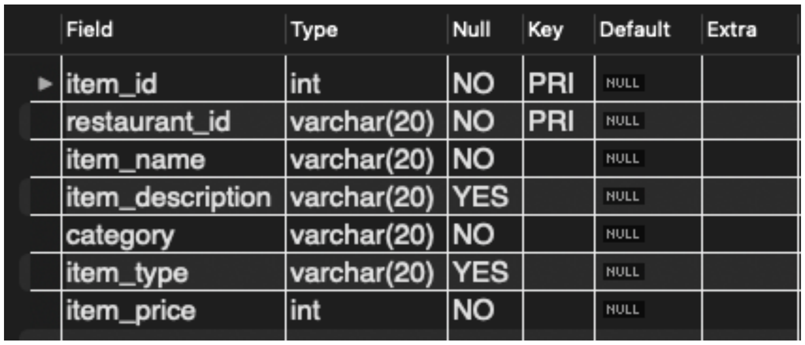
- Restaurant_Id: Primary key which helps to identify a Restaurant uniquely.
- Restaurant_Name: Name of the Restaurant.
- Restaurant_Address: Address of the Restaurant.
- Restaurant_Email: Email of the Restaurant.
- Phone_Number: Phone Number of the Restaurant.



Field	Type	Null	Key	Default	Extra
restaurant_id	varchar(20)	NO	PRI	NULL	
restaurant_name	varchar(20)	NO		NULL	
phone_number	varchar(20)	NO		NULL	
restaurant_email	varchar(30)	YES		NULL	
restaurant_address	varchar(30)	NO		NULL	
admin_id	varchar(20)	NO		NULL	

Entity 11: Menu

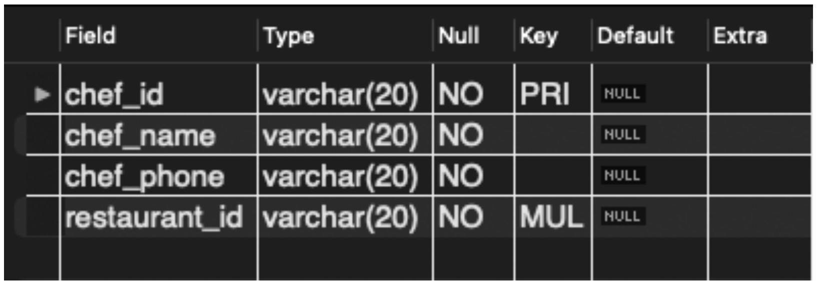
- Item_Id: Primary key which helps to identify food items present in menu in the Restaurant uniquely.
- Item_Name: Name of the Food Item.
- Item_Description: Description of the ingredients present in the food item.
- Category: This attribute show the cuisine of the dish like Indian Food, Korean Food and so on.
- Item_Type: Under which type the food item come under like Dessert, Starter, Main Course and so on.
- Item_Price: Price of the item.



Field	Type	Null	Key	Default	Extra
item_id	int	NO	PRI	NULL	
restaurant_id	varchar(20)	NO	PRI	NULL	
item_name	varchar(20)	NO		NULL	
item_description	varchar(20)	YES		NULL	
category	varchar(20)	NO		NULL	
item_type	varchar(20)	YES		NULL	
item_price	int	NO		NULL	

Entity 12: Chef

- Chef_Id: Primary key which helps to identify a Chef in the Restaurant uniquely.
- Chef_Name: Name of the Chef.
- Phone_Number: Phone Number of the Chef.

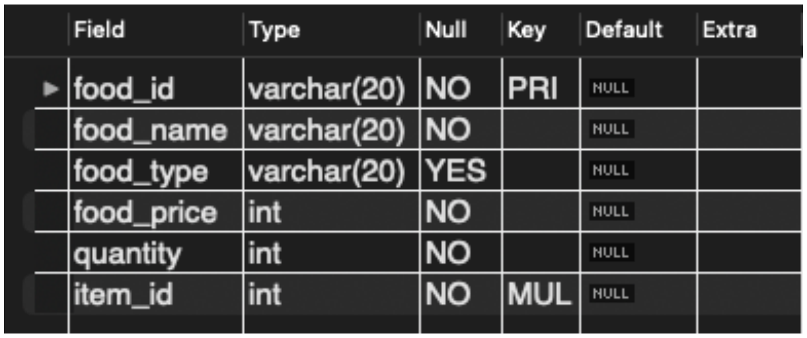


Field	Type	Null	Key	Default	Extra
chef_id	varchar(20)	NO	PRI	NULL	
chef_name	varchar(20)	NO		NULL	
chef_phone	varchar(20)	NO		NULL	
restaurant_id	varchar(20)	NO	MUL	NULL	

Entity 13: Food Details

- Food_Id: Primary key which helps to identify a food ordered by the customer from the menu items uniquely.
- Food_Name: Name of the Food item customer wants to buy.

- Food_Type: Under which type the food item come under like Dessert, Starter, Main Course and so on.
- Food_Price: The Price of the food item.
- Quantity: Quantity of one food item customer wants to order.



Field	Type	Null	Key	Default	Extra
food_id	varchar(20)	NO	PRI	NULL	
food_name	varchar(20)	NO		NULL	
food_type	varchar(20)	YES		NULL	
food_price	int	NO		NULL	
quantity	int	NO		NULL	
item_id	int	NO	MUL	NULL	

Teamwork

Every team member contributed equally from the start choosing entities and attribute, modelling..etc., Specifically:

Satyanarayana Vinay Achanta:

- Creating Relational model.
- Creating ER diagram.
- Converting into Normal forms.

Hari Chandana Kotnani:

- Functional and data requirements.
- Creating tables and inserting values into it.
- Entities and Attributes.

Kotte Venkata Shiva Sai Dheeraj:

- Creating ER Diagram.
- Creation of sql files.
- Implementing and executing queries.

Summary

Online Food Ordering System uses MYSQL Server so there is not any chance of data loss or data security. A customer can choose to have the food delivered or for pick-up. The process consists of a customer choosing the restaurant of their choice, scanning the menu items, choosing an item, and finally choosing for pick-up or delivery. Payment is then administered by paying with a credit card or debit card or in cash at the restaurant when going to pick up. The database informs the customer of the food quality, duration of food preparation, and when the food is ready for pick-up or the amount of time it will take for delivery.

Key Findings

- Clear client requirements will make the modeling and implementation easy.
- Normalization will reduce data redundancy and simplify the query process.

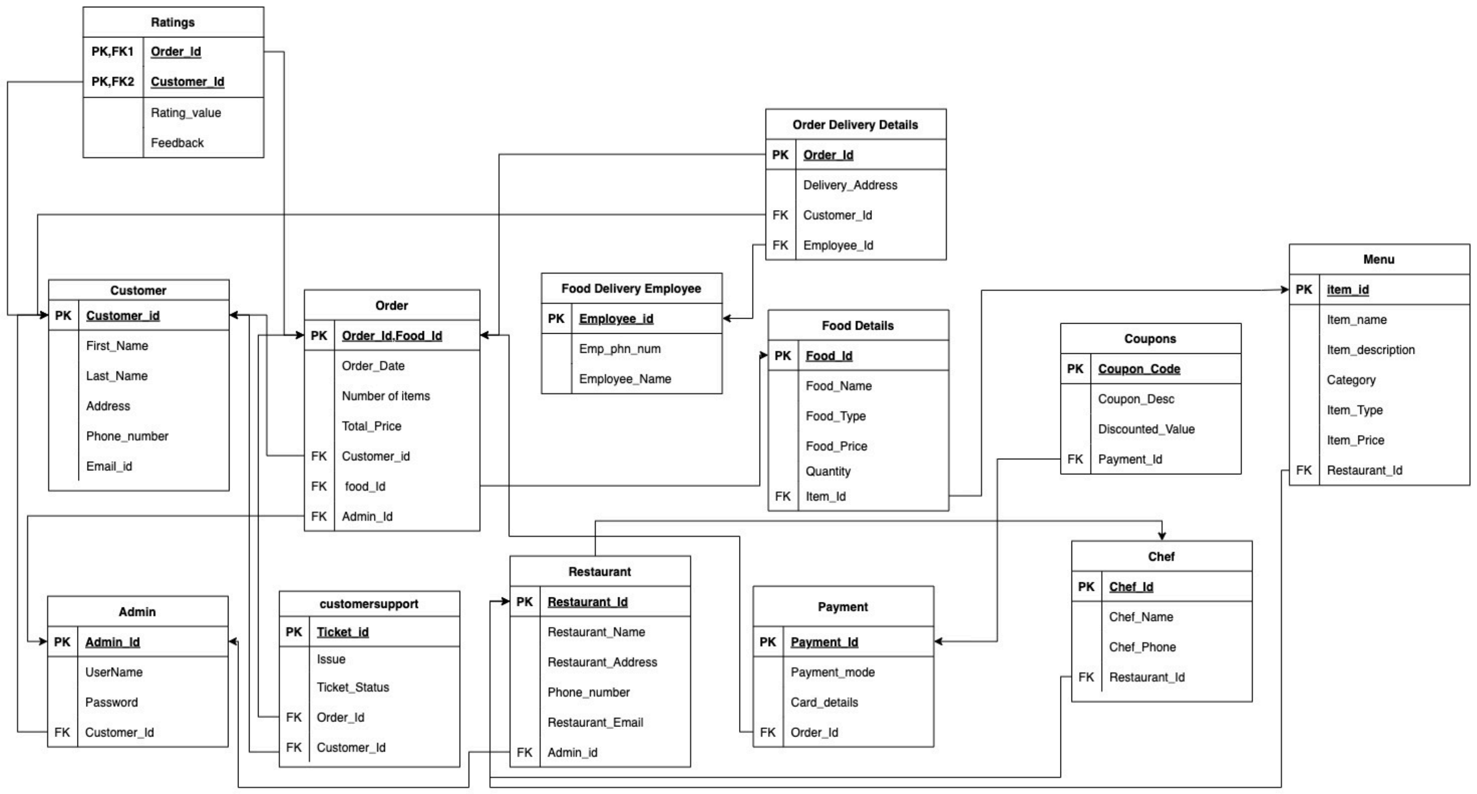
Challenges

- 3NF Normalization.
- There are many considerations while choosing tables and relations between them.

Significance

- This system will help to generate more profits and organize the restaurant better.
- It also allows restaurant owners to save on labor costs and restaurant space needed to serve such customers.

Appendix A


Ratings
PK,FK1 Order_Id
PK,FK2 Customer_Id
Rating_value
Feedback
Order Delivery Details
PK Order_Id
Delivery_Address
FK Customer_Id
FK Employee_Id
Menu
PK item_id
Item_name
Item_description
Category
Item_Type
Item_Price
FK Restaurant_Id
Customer
PK Customer_id
First_Name
Last_Name
Address
Phone_number
Email_id
Order
PK Order_Id,Food_Id
Order_Date
Number of items
Total_Price
FK Customer_id
FK food_Id
FK Admin_Id
Food Delivery Employee
PK Employee_id
Emp_phn_num
Employee_Name
Food Details
PK Food_Id
Food_Name
Food_Type
Food_Price
Quantity
FK Item_Id
Coupons
PK Coupon_Code
Coupon_Desc
Discounted_Value
FK Payment_Id
Chef
PK Chef_Id
Chef_Name
Chef_Phone
FK Restaurant_Id
Admin
PK Admin_Id
UserName
Password
FK Customer_Id
customersupport
PK Ticket_id
Issue
Ticket_Status
FK Order_Id
FK Customer_Id
Restaurant
PK Restaurant_Id
Restaurant_Name
Restaurant_Address
Phone_number
Restaurant_Email
FK Admin_id
Payment
PK Payment_Id
Payment_mode
Card_details
FK Order_Id

Appendix B

Table creation

```
create database food_ordering_database;
use food_ordering_database;


-- Create customer table
CREATE TABLE customer
(
   customer_id varchar(20) NOT NULL,
   first_name varchar(20) NOT NULL,
   last_name varchar(20) NOT NULL,
   phone_no numeric NOT NULL,
   full_address varchar(20) NOT NULL,
   email_id varchar(20),
   CONSTRAINT customer_pkey PRIMARY KEY (customer_id),
   CONSTRAINT customer_customer_id_key UNIQUE (customer_id),
   CONSTRAINT customer_email_id_key UNIQUE (email_id),
   CONSTRAINT customer_phone_no_key UNIQUE (phone_no),
   CONSTRAINT customer_phone_no_check CHECK (phone_no < '10000000000' AND phone_no >
'1000000000')
);



-- Create Restaurant Table
CREATE TABLE restaurant
(
        restaurant_id varchar(20) NOT NULL,
        restaurant_name varchar(20) NOT NULL,
        phone_number varchar(20) NOT NULL,
        restaurant_email varchar(30),
        restaurant_address varchar(30) NOT NULL,
        admin_id varchar(20) NOT NULL,
    CONSTRAINT res_pkey PRIMARY KEY (restaurant_id)
);

-- Create menu table
CREATE TABLE menu
(
  item_id Integer NOT NULL,
  restaurant_id varchar(20) NOT NULL,
  item_name varchar(20) NOT NULL,
  item_description varchar(20),
  category varchar(20) NOT NULL,
  item_type varchar(20),
  item_price integer NOT NULL,
  CONSTRAINT menu_pkey PRIMARY KEY (item_id, restaurant_id),
  CONSTRAINT menu_rid_fkey FOREIGN KEY (restaurant_id)
  REFERENCES restaurant(restaurant_id) MATCH SIMPLE
  ON UPDATE NO ACTION ON DELETE NO ACTION
);

-- Create fooddetails table
CREATE TABLE fooddetails
(
  food_id varchar(20) NOT NULL,
  food_name varchar(20) NOT NULL,
  food_type varchar(20),
  food_price Integer NOT NULL,
  quantity Integer NOT NULL,
  item_id Integer NOT NULL,
  CONSTRAINT order_pkey PRIMARY KEY (food_id),
  FOREIGN KEY (item_id)
  REFERENCES menu(item_id) MATCH SIMPLE
  ON UPDATE NO ACTION ON DELETE NO ACTION
);

-- Create Admin table
CREATE TABLE Admin
(
  admin_id varchar(20) NOT NULL,
  username varchar(20) NOT NULL,
  password varchar(20) NOT NULL,
  customer_id varchar(20) NOT NULL,
  FOREIGN KEY (customer_id)
  REFERENCES customer (customer_id) MATCH SIMPLE
  ON UPDATE NO ACTION ON DELETE NO ACTION
);

-- Create Orders Table
CREATE TABLE orders
(
  order_id varchar(20) NOT NULL,
  food_id varchar(20) NOT NULL,
  total_price float NOT NULL,
  order_date date NOT NULL,
  number_of_items integer NOT NULL,
  admin_id varchar(20) NOT NULL,
  customer_id varchar(20) NOT NULL,
  CONSTRAINT order_pkey PRIMARY KEY (order_id, food_id),
  FOREIGN KEY (customer_id)
  REFERENCES customer (customer_id) MATCH SIMPLE
  ON UPDATE NO ACTION ON DELETE NO ACTION,
  FOREIGN KEY (food_id)
  REFERENCES fooddetails(food_id) MATCH SIMPLE
  ON UPDATE NO ACTION ON DELETE NO ACTION
);



-- Create rating Table
CREATE TABLE rating
(
  feedback varchar(20),
  rating_value integer NOT NULL,
  order_id varchar(20) NOT NULL,
  customer_id varchar(20) NOT NULL,
  FOREIGN KEY (order_id) REFERENCES orders (order_id) MATCH SIMPLE
  ON UPDATE NO ACTION ON DELETE NO ACTION,
  FOREIGN KEY (customer_id) REFERENCES customer(customer_id) MATCH SIMPLE
  ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT rating_value CHECK(rating_value > 0 AND rating_value <= 5)
);

-- Create coupons Table
CREATE TABLE coupons
```

```
(
   coupon_code varchar(20) NOT NULL,
   coupon_value varchar(20) NOT NULL,
   coupon_desc varchar(20),
   payment_id varchar(20) NOT NULL
);

-- Create Chef Table
CREATE TABLE chef
(
           chef_id varchar(20) NOT NULL,
           chef_name varchar(20) NOT NULL,
           chef_phone varchar(20) NOT NULL,
           restaurant_id varchar(20) NOT NULL,
      PRIMARY KEY (chef_id),
           FOREIGN KEY (restaurant_id) REFERENCES restaurant(restaurant_id) MATCH SIMPLE
           ON UPDATE NO ACTION ON DELETE NO ACTION
);

-- Create Customersupport Table
CREATE TABLE customersupport
(
   customer_id varchar(20) NOT NULL,
   ticket_id varchar(20) NOT NULL,
   issue varchar(20) NOT NULL,
   status varchar(20) NOT NULL,
   order_id varchar(20) NOT NULL,
   FOREIGN KEY (order_id) REFERENCES orders (order_id) MATCH SIMPLE
   ON UPDATE NO ACTION ON DELETE NO ACTION,
   FOREIGN KEY (customer_id) REFERENCES customer(customer_id) MATCH SIMPLE
   ON UPDATE NO ACTION ON DELETE NO ACTION
);

-- Create food delivery person Table
CREATE TABLE food_delivery_employee
(
   emp_name varchar(20) NOT NULL,
   phn_num varchar(20) NOT NULL,
   customer_id varchar(20) NOT NULL,
   order_id varchar(20) NOT NULL,
   emp_id varchar(20) NOT NULL,
   PRIMARY KEY (emp_id),
   FOREIGN KEY (order_id) REFERENCES orders (order_id) MATCH SIMPLE
   ON UPDATE NO ACTION ON DELETE NO ACTION,
   FOREIGN KEY (customer_id) REFERENCES customer(customer_id) MATCH SIMPLE
   ON UPDATE NO ACTION ON DELETE NO ACTION
);

-- Create Order Delivery Details Table
CREATE TABLE order_delivery_details
(
   order_id varchar(20) NOT NULL,
   emp_id varchar(20) NOT NULL,
   delivery_address varchar(30) NOT NULL,
   PRIMARY KEY (emp_id),
   FOREIGN KEY (order_id) REFERENCES orders (order_id) MATCH SIMPLE
   ON UPDATE NO ACTION ON DELETE NO ACTION
);
```

```
-- Create Payment Table
CREATE TABLE payment
(
	payment_id varchar(20) NOT NULL,
	payment_mode varchar(20) NOT NULL,
	card_details varchar(20) NOT NULL,
	order_id varchar(20) NOT NULL,
	FOREIGN KEY (order_id) REFERENCES orders (order_id) MATCH SIMPLE
	ON UPDATE NO ACTION ON DELETE NO ACTION,
	CONSTRAINT customer_card_details_check CHECK (card_details > 1000000000000000
AND card_details < 10000000000000000)
);
```

INSERT VALUES INTO TABLES

```
use food_ordering_database;

-- insert value in customer table
insert into customer values("c1", "vinay", "A", "2234567890", "597   forest gate",
"vinay@gmail.com");
insert into customer values("c2", "chandana", "K", "2234567899", "697 university pines",
"chandana@gmail.com");
insert into customer values("c3", "ash", "Al", "3234567889", "789 university pines",
"ash@gmail.com");
insert into customer values("c4", "nit", "N", "3234567787", "778 kampus corner", "nit@gmail.com");
insert into customer values("c5", "dheeraj", "D", "9423456666", "899 university pines",
"dheeraj@gmail.com");
insert into customer values("c6", "sai", "S", "3255567890", "122 mountainview", "sai@gmail.com");
insert into customer values("c7", "vamsi", "V", "4235557890", "344 mountainview",
"vamsi@gmail.com");


-- insert value in restaurant table
insert into restaurant values("1", "annapurna mess","1234567890", "annapurnamess@gmail.com", "123
East 321 north apt-2", "a1");
insert into restaurant values("2", "flechazo", "2234567190", "flechazo@gmail.com","898 East 888
north apt-3", "a2");
insert into restaurant values("3", "barbecue", "3234567890", "barbecue@gmail.com","999 East 777
north apt-4", "a3");
insert into restaurant values("4", "marine", "5234566890", "marine@gmail.com", "101 East 391 north
apt-2","a4");
insert into restaurant values("5", "warensberg", "2234667890", "warensberg@gmail.com", "111 East
421 north apt-2", "a5");
insert into restaurant values("6", "manaruchulu", "7234557890", "manaruchulu@gmail.com", "113
East 391 north apt-9", "a6");
insert into restaurant values("7", "pizzahut", "9234567890", "pizzahut@gmail.com", "192 East 921
north apt-9", "a7");


-- insert value in menu table
insert into menu values("11", "1", "ChickenTikkaMasala", "Chicken with masala", "Indian", "curry",
"12");
insert into menu values("22", "2", "Cavatappi", "pasta", "Italian", "maincusine", "16");
insert into menu values("33", "3", "Eggrice", "Fried rice", "Chinese", "lunch", "9");
insert into menu values("44", "4", "Oniondosa", "Dosa", "South Indian", "breakfast", "11");
insert into menu values("55", "5", "Kimchistew", "noodles with soup", "Korean", "snacks", "20");
insert into menu values("66", "6", "Pancakes", "Made with flour", "American", "breakfast", "23");
insert into menu values("77", "7", "chickentikka", "roasted chicken", "Indian", "starter", "19");


-- insert value in fooddetails table
insert into fooddetails values("f1", "SAMOSA", "snacks", 19, 10, 11);
insert into fooddetails values("f2", "MIXED PLATTER", "snacks", 34, 7, 22);

insert into fooddetails values("f3", "TIKKA MASALA", "curry", 18, 4, 33);

insert into fooddetails values("f4", "SAAG", "curry", 13, 7, 11);

insert into fooddetails values("f5", "CHANAMASALA", "curry", 21, 8, 55);

insert into fooddetails values("f7", "SAMOSA", "snacks", 33, 4, 77);


-- insert value in Admin table

insert into Admin values("a1", "kate", "Kate123", 'c1');

insert into Admin values("a2", "tom", "tom456", 'c2');

insert into Admin values("a3", "jerry", "jerry676", 'c3');

insert into Admin values("a4", "sam", "sam555", 'c4');

insert into Admin values("a5", "rakul", "rakul999", 'c5');

insert into Admin values("a6", "sophie", "sophie989", 'c6');

insert into Admin values("a7", "katty", "katty656", 'c7');



-- insert value in orders table

insert into orders(order_id, food_id, total_price, number_of_items, admin_id, customer_id)
values("O1", "f1", 10.123, 3, "a1", "c1");

insert into orders(order_id, food_id, total_price, number_of_items, admin_id, customer_id)
values("O2", "f2", "15", "4", "a2", "c2");

insert into orders(order_id, food_id, total_price, number_of_items, admin_id, customer_id)
values("O3", "f3", "18", "5", "a3", "c3");

insert into orders(order_id, food_id, total_price, number_of_items, admin_id, customer_id)
values("O4", "f4", "11", "1", "a4", "c4");

insert into orders(order_id, food_id, total_price, number_of_items, admin_id, customer_id)
values("O5", "f5", "13", "1", "a5", "c5");

insert into orders(order_id, food_id, total_price, number_of_items, admin_id, customer_id)
values("O7", "f7", "32", "3", "a7", "c7");

insert into orders(order_id, food_id, total_price, number_of_items, admin_id, customer_id)
values("O8", "f3", "32", "3", "a7", "c7");
```

```
insert into orders(order_id, food_id, total_price, number_of_items, admin_id, customer_id)
values("O9", "f4", "32", "3", "a7", "c7");


-- insert value in rating table
insert into rating values("delicious", "4", "O1", "c1");
insert into rating values("average", "3", "O2","c2");
insert into rating values("awesome", "5", "O4", "c4");
insert into rating values("not bad", "2", "O5", "c5");


-- insert value in coupons table
insert into coupons values("c1", "TAKE20","NEW LOGIN" , "p1");
insert into coupons values("c2", "LOVE5", "VALENTINES" , "p2");
insert into coupons values("c3", "SWIGGY", "SECOND ORDER" , "p3");
insert into coupons values("c4", "THANKSGIVING", "THANKSGIVING" , "p4");
insert into coupons values("c5", "CHRISTMAXEVE", "CHRISTMAS" , "p5");
insert into coupons values("c6", "HOLIDAYOFF", "HOLIDAY OFFER" , "p6");
insert into coupons values("c7", "WEEKENDBLUES", "SATURDAY OFFER" , "p7");


-- insert value in chef table
insert into chef values("ch1", "singh", '1238567890',"1");
insert into chef values("ch2", "jamil" , '6239687990', "2");
insert into chef values("ch3", "dillion" , '9234567890', "3");
insert into chef values("ch4", "raasa", '7234567890', "4");
insert into chef values("ch5", "adriyana" , '2234567890', "5");
insert into chef values("ch6", "venky" , '5234767890', "6");
insert into chef values("ch7", "jessica", '1234767890', "7");


-- insert value in customersupport table
insert into customersupport values("c1", "t1", "food not delivered", "in progress" , "O1");
insert into customersupport values("c2", "t2", "wrong food delivered", "in progress" , "O2");
insert into customersupport values("c3", "t3", "food tastes bad", "in progress" , "O3");
insert into customersupport values("c4", "t4", "app isn't responding", "in progress" , "O4");
insert into customersupport values("c5", "t5", "payment declined", "in progress" , "O5");
insert into customersupport values("c6", "t6", "refund not received", "in progress" , "O7");

-- insert value in food delivery employee table
insert into food_delivery_employee values("ram", "1234567890", "C1", "O1" , "r1");
insert into food_delivery_employee values("sam", "2123456789", "C2", "O2" , "r2");
insert into food_delivery_employee values("vam", "3214567898", "C3", "O3" , "r3");
insert into food_delivery_employee values("pam", "5214567898", "C4", "O4" , "r4");
insert into food_delivery_employee values("aam", "6214567898", "C5", "O5" , "r5");
insert into food_delivery_employee values("cam", "8214567598", "C7", "O7" , "r7");

-- insert value in order delivery details table
insert into order_delivery_details values("O1", "vinay1", "597   forest gate");
insert into order_delivery_details values("O2", "chandana2", "697 university pines");
insert into order_delivery_details values("O3", "ashwiniakka3", "789 university pines");
insert into order_delivery_details values("O4", "nithya4", "778 kampus corner");
insert into order_delivery_details values("O5", "dheeraj5", "899 university pines");
insert into order_delivery_details values("O7", "vamsi7", "344 mountainview");



-- insert value in payment table
insert into payment values("p1", "cash", 1000000000000001, "o1");
insert into payment values("p2", "card", 4400112344111111, "o2");
insert into payment values("p3", "card", 7700112343111111, "o3");
insert into payment values("p4", "card", 9900332343111111, "o4");
insert into payment values("p5", "cash", 1100112343111111, "o5");
```

insert into payment values("p7", "coupon", 3003132343111111, "o7");

Functional requirement queries

use food_ordering_database;

-- Get all the customer support tickets that are active.

SELECT * FROM customersupport WHERE status = "in progress";

-- Details about the number of customers who paid with different payment modes.

SELECT payment_mode, COUNT(payment_mode) FROM Customer C, orders O, Payment P WHERE C.customer_id = O.customer_id and O.order_id = P.order_id GROUP BY payment_mode;

-- Get the information about the customer who applied coupons while payment.

SELECT C.first_name, O.order_id, C1.coupon_code FROM Customer C, orders O, Payment P, Coupons C1 WHERE C.customer_id = O.customer_id and O.order_id = P.order_id and P.payment_id = C1.payment_id;

-- Get the maximum rating of all the restaurants.

SELECT Res.restaurant_name, R.rating_value FROM rating R, orders O, restaurant Res WHERE R.order_id = O.order_id and O.admin_id = Res.admin_id and R.rating_value >= all(SELECT R1.rating_value FROM rating R1);

-- CREATE OR REPLACE VIEW customer_restaurant AS

SELECT R.restaurant_name, M.item_name, M.item_price, M.item_description, M.item_type, M.category FROM restaurant R, menu M where R.restaurant_id = M.restaurant_id;

SELECT * FROM customer_restaurant;

SELECT payment_mode, COUNT(payment_mode) FROM Customer C, orders O, Payment P WHERE C.customer_id = O.customer_id and O.order_id = P.order_id GROUP BY payment_mode;

-- At least one item needs to be selected to proceed with the payment.

```
SELECT C.first_name, C.last_name, O.order_id FROM Customer C, orders O, Payment P WHERE
C.customer_id = O.customer_id and O.order_id = P.order_id and O.total_price > 0;
```

-- Get the customer details who ordered food from the location

```
SELECT * FROM customer c, order_delivery_details O , orders o1 WHERE O.order_id = o1.order_id
and c.customer_id = o1.customer_id and o.delivery_address like "%university pines%";
```

-- Get the order details that were ordered between selected dates.

```
SELECT * FROM orders WHERE order_date BETWEEN "2018-06-17" and "2020-06-25";
```

-- Get the count of issues got from each restaurant.

```
SELECT   C.customer_id, O.order_id FROM customersupport C, orders O, restaurant R, customer C1
WHERE C.order_id = O.order_id and O.admin_id = R.admin_id and C.customer_id = C1.customer_id;
```

-- Get the chef details who worked for each restaurant.

```
SELECT R.restaurant_name, C.chef_name FROM restaurant R, chef C WHERE R.restaurant_id =
C.restaurant_id;
```

-- Get the details of the admin who manages each restaurant.

```
SELECT A.username, r.restaurant_name, r.restaurant_address, r.phone_number FROM admin A,
restaurant r WHERE A.admin_id = r.admin_id;
```