Pick Ur Plate

TEAM PANDA

SATYANARAYANA VINAY ACHANTA(A02395874)

VENKATA SHIVA SAI DHEERAJ KOTTE(A02394047)

HARI CHANDANA KOTNANI(A02395874)

Introduction

- Pick Ur Plate database was developed to provide the facilities for ordering food online that can be delivered to their place from nearby restaurants.
- We will organize the data of our own exclusive fleet of delivery personnel to pick up orders from restaurants and deliver them to customers.
- The main objective of this project is to store the details of the customers, orders, chefs, restaurants, admin, menu, food details, food delivery employees, and payment.

Requirements Analysis

Data Requirements:

- Below is some of the important information which our online food ordering system needs to store
  - **Customer**(identified by the unique Customer Id, First Name, Last Name, Phone number, Email, and Address).
  - The customer can provide **reviews** to the restaurant and can contact **customer support** if any issue with the order occurs.
  - **Restaurant** (identified by a unique Restaurant Id, Name, Email, Phone, and Address) has **Chefs** (identified by a unique Chef Id).
  - **Order**(identified by the unique Order Id, Order Date, and the total price of the order customer bought from the restaurant)

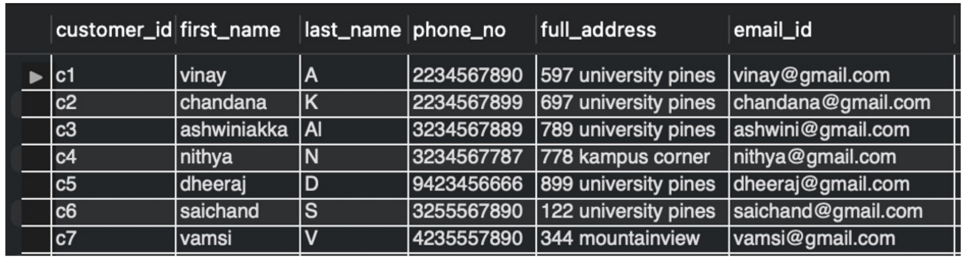
Requirements Analysis

Functional Requirements:

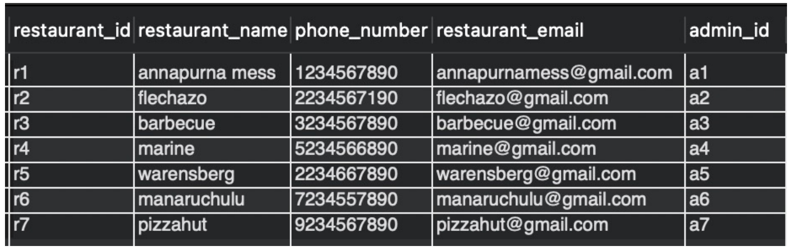
- Customer support should only have access to the details of the order placed by the customer.
- The chef can only have the details of the order not any other information about the customer.
- A customer can place one order from one restaurant at a time.
- Delivery employees should have details of the customer’s name, phone, location, and order id only.
- Rating must be in the range of 1 to 5 which implies bad to great!
- The username and password entered by the admin should match the data stored in the database.
- Ratings should be reviewed by the admin to consider the changes required to satisfy the customer.
- All food types should be present in the food details table.

Data

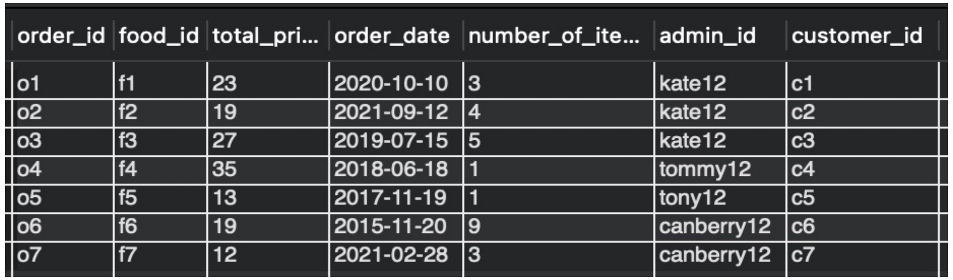
- We did not use any dataset to fill in tables. Instead, we filled out tables manually to test the functionalities based on common operations.



	customer_id	first_name	last_name	phone_no	full_address	email_id
▶	c1	vinay	A	2234567890	597 university pines	vinay@gmail.com
	c2	chandana	K	2234567899	697 university pines	chandana@gmail.com
	c3	ashwiniakka	Al	3234567889	789 university pines	ashwini@gmail.com
	c4	nithya	N	3234567787	778 kampus corner	nithya@gmail.com
	c5	dheeraj	D	9423456666	899 university pines	dheeraj@gmail.com
	c6	saichand	S	3255567890	122 university pines	saichand@gmail.com
	c7	vamsi	V	4235557890	344 mountainview	vamsi@gmail.com

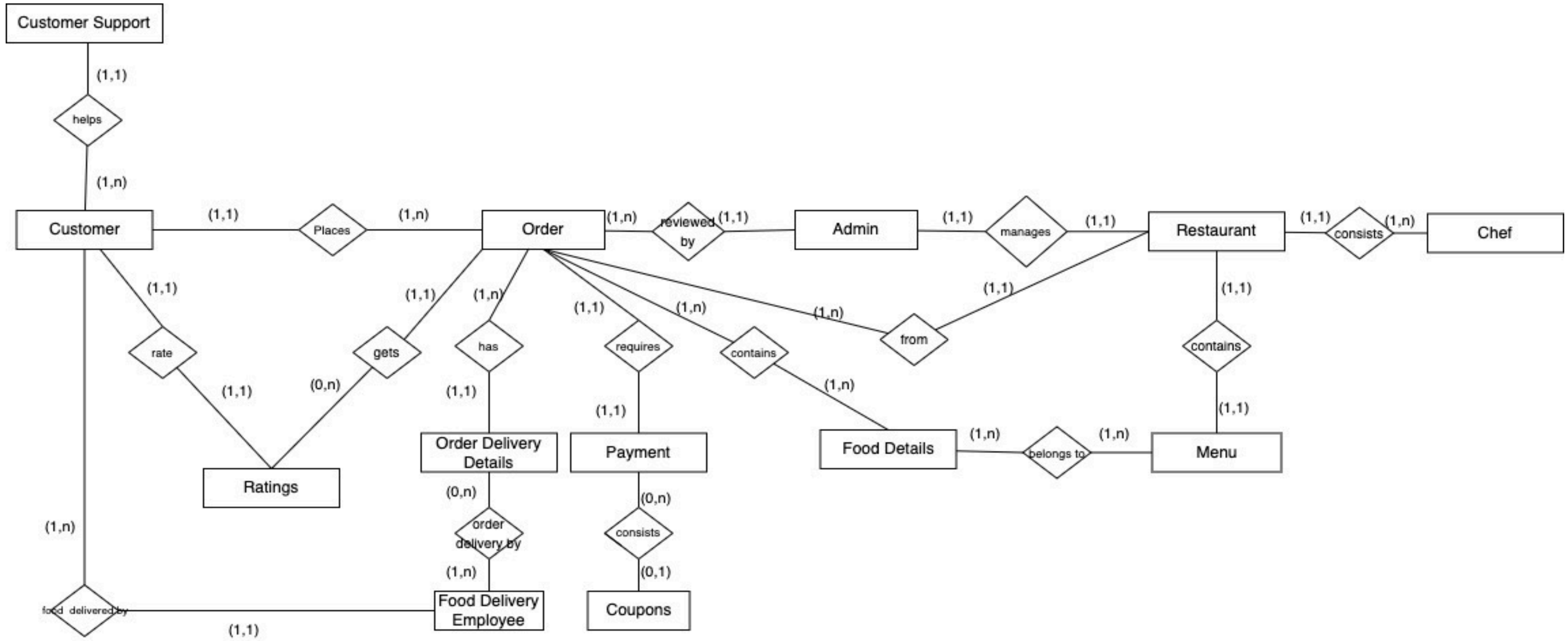


restaurant_id	restaurant_name	phone_number	restaurant_email	admin_id
r1	annapurna mess	1234567890	annapurnamess@gmail.com	a1
r2	flechazo	2234567190	flechazo@gmail.com	a2
r3	barbecue	3234567890	barbecue@gmail.com	a3
r4	marine	5234566890	marine@gmail.com	a4
r5	warensberg	2234667890	warensberg@gmail.com	a5
r6	manaruchulu	7234557890	manaruchulu@gmail.com	a6
r7	pizzahut	9234567890	pizzahut@gmail.com	a7

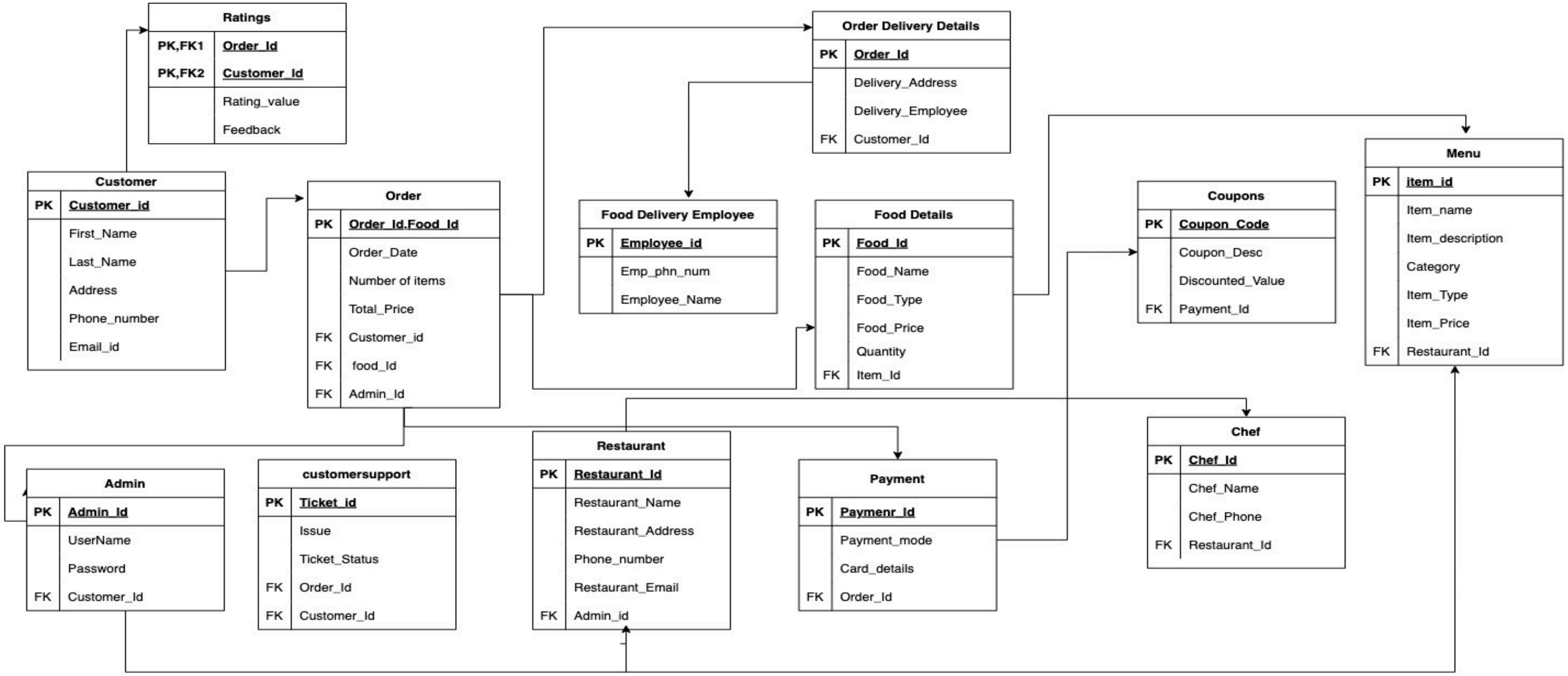


order_id	food_id	total_pri...	order_date	number_of_ite...	admin_id	customer_id
o1	f1	23	2020-10-10	3	kate12	c1
o2	f2	19	2021-09-12	4	kate12	c2
o3	f3	27	2019-07-15	5	kate12	c3
o4	f4	35	2018-06-18	1	tommy12	c4
o5	f5	13	2017-11-19	1	tony12	c5
o6	f6	19	2015-11-20	9	canberry12	c6
o7	f7	12	2021-02-28	3	canberry12	c7

ER Diagram


Customer Support
(1,1)
helps
(1,n)
Customer
(1,1)
Places
(1,n)
Order
(1,n)
reviewed by
(1,1)
Admin
(1,1)
manages
(1,1)
Restaurant
(1,1)
consists
(1,n)
Chef
(1,1)
rate
(1,1)
(1,1)
gets
(0,n)
(1,n)
has
(1,1)
(1,1)
requires
(1,1)
(1,n)
contains
(1,n)
(1,n)
from
(1,1)
(1,1)
contains
(1,1)
Ratings
Order Delivery Details
(0,n)
order delivery by
(1,n)
Food Delivery Employee
Payment
(0,n)
consists
(0,1)
Coupons
Food Details
(1,n)
belongs to
(1,n)
Menu
(1,n)
food delivered by
(1,1)

Relational Model



	Ratings
PK,FK1	Order_Id
PK,FK2	Customer_Id
	Rating_value
	Feedback

	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

	Order Delivery Details
PK	Order_Id
	Delivery_Address
	Delivery_Employee
FK	Customer_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

	Menu
PK	item_id
	Item_name
	Item_description
	Category
	Item_Type
	Item_Price
FK	Restaurant_Id

	Chef
PK	Chef_Id
	Chef_Name
	Chef_Phone
FK	Restaurant_Id

	Admin
PK	Admin_Id
	UserName
	Password
FK	Customer_Id

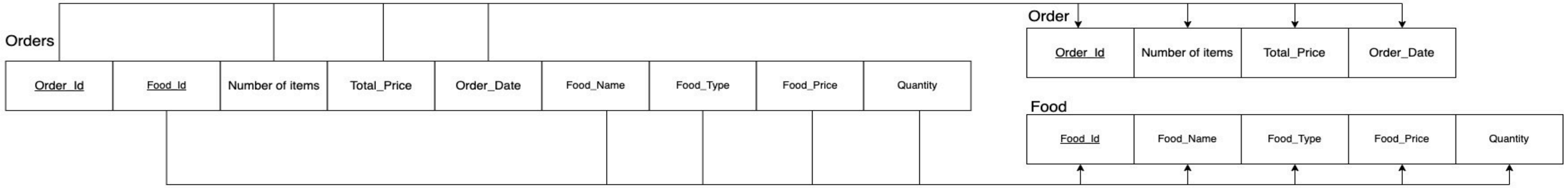
	customersupport
PK	Ticket_id
	Issue
	Ticket_Status
FK	Order_Id
FK	Customer_Id

	Restaurant
PK	Restaurant_Id
	Restaurant_Name
	Restaurant_Address
	Phone_number
	Restaurant_Email
FK	Admin_id

	Payment
PK	Paymenr_Id
	Payment_mode
	Card_details
FK	Order_Id

Relational Model

- Most of the tables are already in Normal Form.
- **Challenged by Normal forms:**



Orders

<u>Order_Id</u>	<u>Food_Id</u>	Number of items	Total_Price	Order_Date	Food_Name	Food_Type	Food_Price	Quantity

Order

<u>Order_Id</u>	Number of items	Total_Price	Order_Date

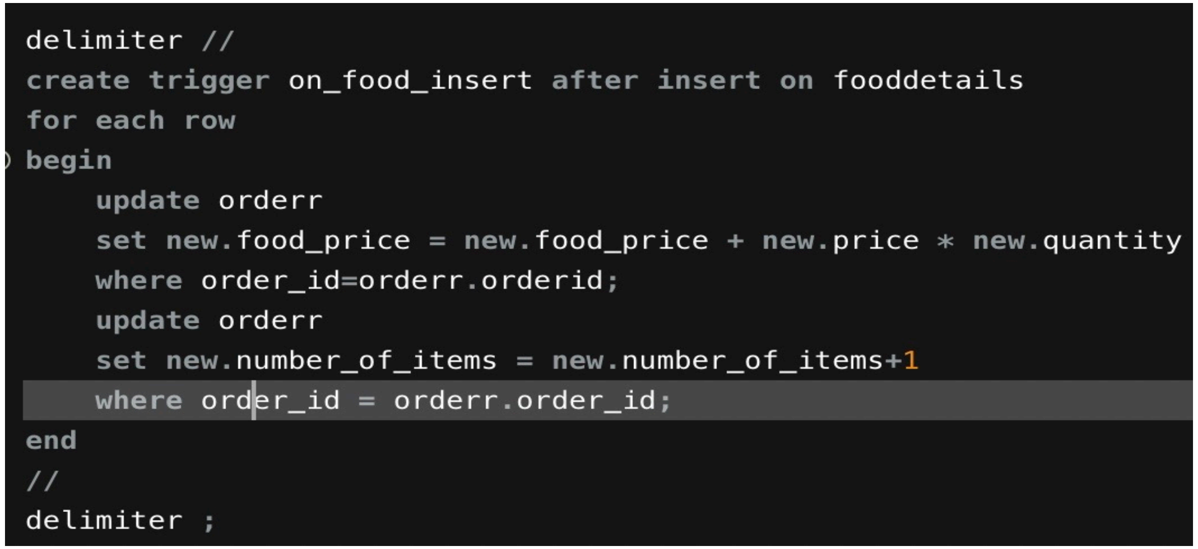
Food

<u>Food_Id</u>	Food_Name	Food_Type	Food_Price	Quantity

- The primary key is defined by the combination of OrderId and Food_Id. 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 he will face another problem, which is violating the rule for 2nd NF.

Functionalities

- Get the highest-rated restaurant out of all.
- Get the most sold food item in each of the restaurants.
- A trigger is used for automatic updates in my project basically when a user enters the food item(in the food table) he wants the total price will be automatically updated in the order table.



```
delimiter //
create trigger on_food_insert after insert on fooddetails
for each row
begin
    update orderr
    set new.food_price = new.food_price + new.price * new.quantity
    where order_id=orderr.orderid;
    update orderr
    set new.number_of_items = new.number_of_items+1
    where order_id = orderr.order_id;
end
//
delimiter ;
```

Summary and Conclusion

- **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.

Questions?