Food Delivery Intermediaries

Group 1: Haokai Gou, Harsh Wardhan, Jeet Patel, Kirsten Hugh, Sarah Alikhan, Scarlet Wessin, Snehal Jitendra Deshmukh

Table of Contents

Background Information

Background Information

Food Delivery: Business Application

Food delivery has become an increasingly important aspect of modern-day living. According to a recent survey by Mintel, approximately one-third of the American population orders food online at least twice per week, highlighting the popularity of food delivery services (Hendelmann, 2023). In addition, the COVID-19 pandemic has further accelerated the growth of the food delivery industry, as more people opt for the convenience and safety of home delivery.

The global food delivery market is expected to continue its rapid expansion in the coming years, with a projected value of $182.3 billion by 2024 (Spdload, 2023). This growth is driven by the increasing availability of food delivery services across a wide range of businesses, including restaurants, grocery stores, meal kit services, catering companies, and food trucks. With the rise of mobile apps and online ordering platforms, food delivery has become more accessible and convenient than ever before, making it an important part of the overall food industry landscape.

Food Delivery: Business Application

Food delivery has become an essential aspect of the food industry, with businesses around the world recognizing its potential and adopting it as a valuable service. It is a type of courier service that involves transporting meals or food items from stores, restaurants, or third-party applications to customers who have placed orders and require delivery on an immediate or scheduled basis (Hendelmann, 2023).

The growth of the food delivery industry can be attributed to several factors, including technological advancements that have made it easier and more efficient to order and deliver food. Mobile apps and online platforms have transformed the way people order food, enabling them to browse menus, customize orders, and track deliveries in real-time.

In addition, changing consumer habits have played a significant role in the rise of food delivery. Consumers are increasingly seeking convenience and accessibility, and food delivery services offer a hassle-free way to enjoy their favorite meals without leaving their homes or offices. This has been especially important during the COVID-19 pandemic, which has led to an increase in demand for contactless delivery options.

Overall, food delivery has become an important business application for a range of food industry players, from small local restaurants to large multinational chains. By embracing this service, businesses can increase their reach and tap into new customer segments, while providing a valuable service that meets the evolving needs of today's consumers.

> *"What food item do you think was delivered in the earliest documented case of food delivery?*
>
> ***Yes, it was pizza."***

Food Delivery: Model

In a food delivery system, there are typically three parties involved.

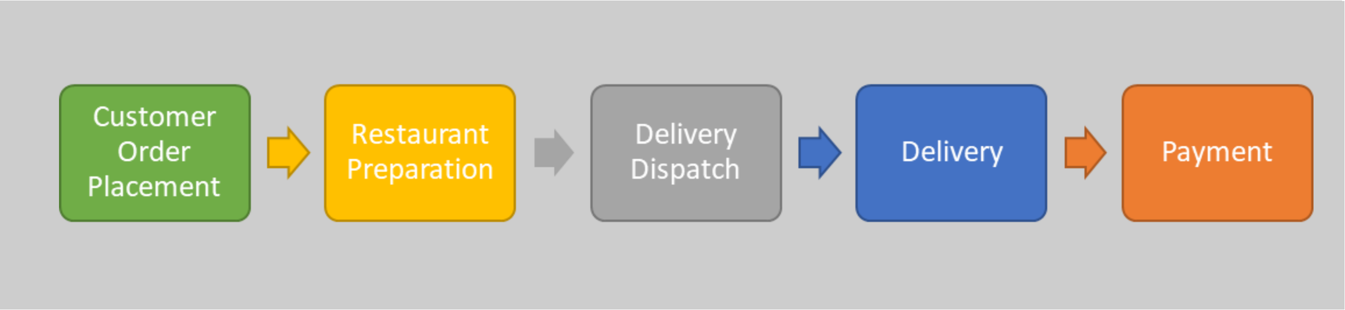
1. **The customer/ person ordering**: Customers can place their orders online via a third-party app or a restaurant's website, or they can place the order over the phone. The order usually includes details such as the type of food, the quantity, the delivery address, and the preferred delivery time.

2. **The delivery agent:** who is responsible for transporting the food from the restaurant to the customer's location. The delivery agent can be an employee of the restaurant or a third-party delivery service provider. In recent years, many third-party delivery service providers have emerged, such as Uber Eats, DoorDash, and Grubhub, which act as intermediaries between the customer and the restaurant.

3. **The restaurant where the order is placed**: The restaurant prepares the food, packages it, and hands it over to the delivery agent for transportation. The restaurant is responsible for ensuring that the food is of high quality and meets the customer's expectations. Additionally, the restaurant needs to coordinate with the delivery agent to ensure that the food is delivered on time and in good condition. In some cases, restaurants may also have their own delivery system, which means they handle both the preparation and delivery of the food.

MODEL OPERATIONS:

In broader terms, below is the overview of how the model works or its operations:



Customer Order Placement → Restaurant Preparation → Delivery Dispatch → Delivery → Payment

Customer Order Placement:

Customers place their food orders through a mobile app or website, they browse through the available restaurant options selecting their preferred restaurant, menu items, and delivery address. Finally, they review the details of their selection on the checkout page. On the checkout page, they will then set the tip amount, payment method, delivery address, order delivery time, and if there is a promotion they wish to apply to their order.

Restaurant Preparation:

Upon confirmation of payment, the restaurant receives the order and prepares the food, packaging it for delivery.

Delivery Dispatch:

The delivery service matches available drivers in the area to the order, dispatches a delivery driver to pick up the food from the restaurant and deliver it to the customer.

Delivery:

The delivery driver transports the food to the customer, following the optimized route provided to him/her by the delivery service's dispatch system.

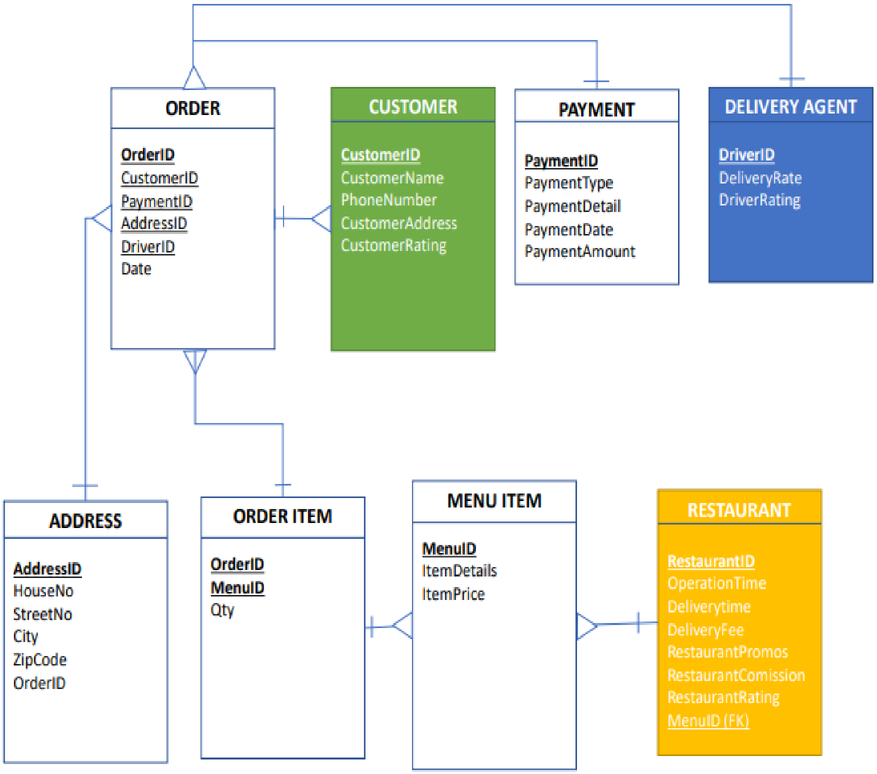
Payment:

The customer pays for the food, delivery fee, obtains a discount (if available) and any other applicable charges through the delivery service's mobile app or website. The customer payment details are verified prior to finalizing the order. The delivery service also pays the restaurant for the cost of the food and takes a commission for facilitating the transaction. The food delivery intermediary or service also pays the driver for the delivery of food.

Entity Relationship Diagram

Online Food Ordering System ER Diagram

The ER Diagram for Online Food Ordering System illustrates the entity relationships between each entity.



Entities in the Model are Customer, Restaurant, Menu Item, Order, Delivery Agent, Payment and Address

Entities And Their Relationships

Entities in the Model:

The Food Delivery System's Model shall consist of the following entities.

1. **Customer:**

   This entity would represent the customers who use the online food delivery service to order food from restaurants. Primary key for this entity will be CustomerID.

2. **Restaurant:**

   This entity would represent the restaurants that provide food to customers through the online food delivery service. Primary key for this entity will be RestaurantID.

3. **Menu Item:**

   This entity would represent the different items available on the menus of restaurants. Primary key for this entity will be MenuID.

4. **Order:**

   This entity would represent the orders placed by customers for food items from restaurants. Primary key for this entity will be OrderID.

5. **Delivery Agent:**

   This entity would represent the agents responsible for delivering food items to customers. Primary key for this entity will be DriverID.

6. **Payment:**

   This entity would represent the payments made by customers for their food orders. Primary key for this entity will be PaymentID.

7. **Address:**

   This entity would represent the delivery addresses of customers. Primary key for this entity will be AddressID.

Relationship and Cardinality Between the Entities

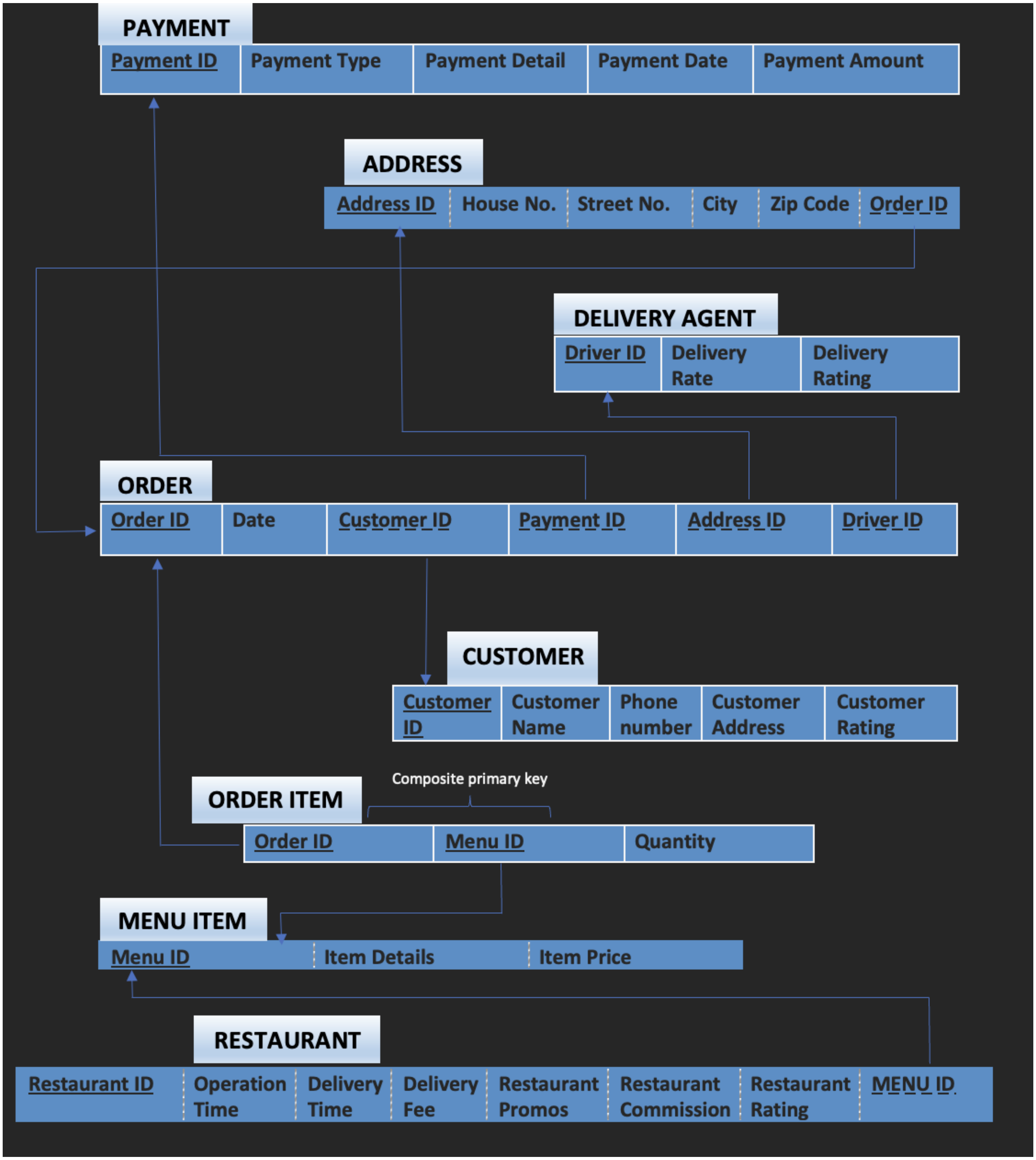
These entities can be related to each other in the following ways:

- A customer can place multiple orders and an order can be placed by a single customer. This is a one-to-many relationship.
- A restaurant can have multiple menu items and a menu item can belong to only one restaurant. This is a one-to-many relationship.
- An order can consist of multiple menu items and a menu item can be a part of multiple orders. This is a many-to-many relationship and can be resolved by creating a new entity, Order Item, to represent the relationship between Orders and Menu Items with a composite Primary Key of OrderID and MenuID.

- An order can be placed for a single delivery address and a delivery address can receive multiple orders. This is a one-to-many relationship.
- An order can be paid for using a single payment and a payment can be used to pay for multiple orders. This is a one-to-many relationship.
- An order can be delivered by a single delivery agent and a delivery agent can deliver multiple orders. This is a one-to-many relationship.

Logical Schema 3rd Normal Form

Food Delivery Logical Schema



PAYMENT

Payment ID	Payment Type	Payment Detail	Payment Date	Payment Amount

ADDRESS

Address ID	House No.	Street No.	City	Zip Code	Order ID

DELIVERY AGENT

Driver ID	Delivery Rate	Delivery Rating

ORDER

Order ID	Date	Customer ID	Payment ID	Address ID	Driver ID

CUSTOMER

Customer ID	Customer Name	Phone number	Customer Address	Customer Rating

ORDER ITEM

Composite primary key

Order ID	Menu ID	Quantity

MENU ITEM

Menu ID	Item Details	Item Price

RESTAURANT

Restaurant ID	Operation Time	Delivery Time	Delivery Fee	Restaurant Promos	Restaurant Commission	Restaurant Rating	MENU ID

RELATIONS and ATTRIBUTES in the schema

The logical schema defines the structure of the data itself and the relationships between the various attributes, tables, and entries.

The above Logical Schema conforms to the ***3rd Normal Form***.

- All the database tables have atomic attributes. There are no columns in any of the tables with multiple values. Hence, the condition for the 1st Normal Form is satisfied.

- All the database tables do not have partial dependency. All columns have full functional dependencies. Hence, the condition for the 2nd Normal Form is satisfied.

- All the database tables do not have transitive dependency. Hence, all the conditions for the 3rd Normal Form are satisfied.

- Our database system consists of 8 relations PAYMENT, ADDRESS, DELIVERY AGENT, CUSTOMER, ORDER, ORDER ITEM, MENU AND RESTAURANT.

- PAYMENT has PaymentID, DELIVERY AGENT has DriverID and CUSTOMER has CustomerID as primary keys respectively.

- ADDRESS has Address ID as primary key and Order ID as foreign key.

- ORDER has OrderID as primary key and primary keys of PAYMENT, ADDRESS, DRIVER, CUSTOMER as foreign keys.

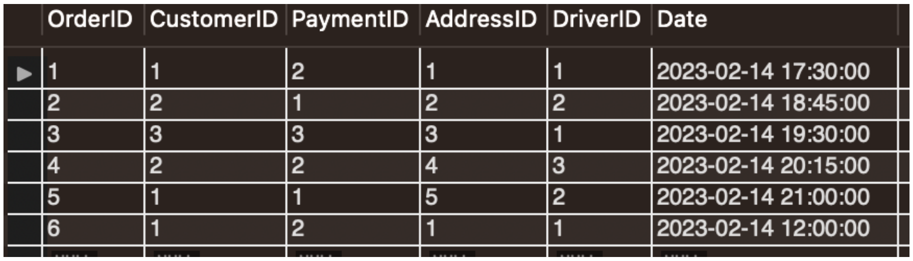
- RESTAURANT has RestaurantID as primary key and MenuID as foreign key, which is also the primary key for MENU.

- ORDER ITEM has a composite primary key made of two attributes OrderID and MenuID which are also foreign keys from the relation ORDER and MENU.

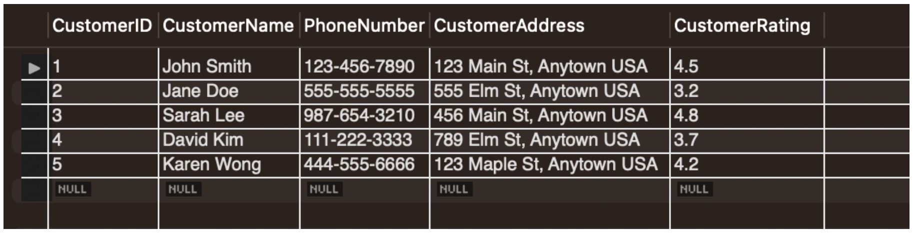
MySQL Workbench

Food Delivery: Creating the Database

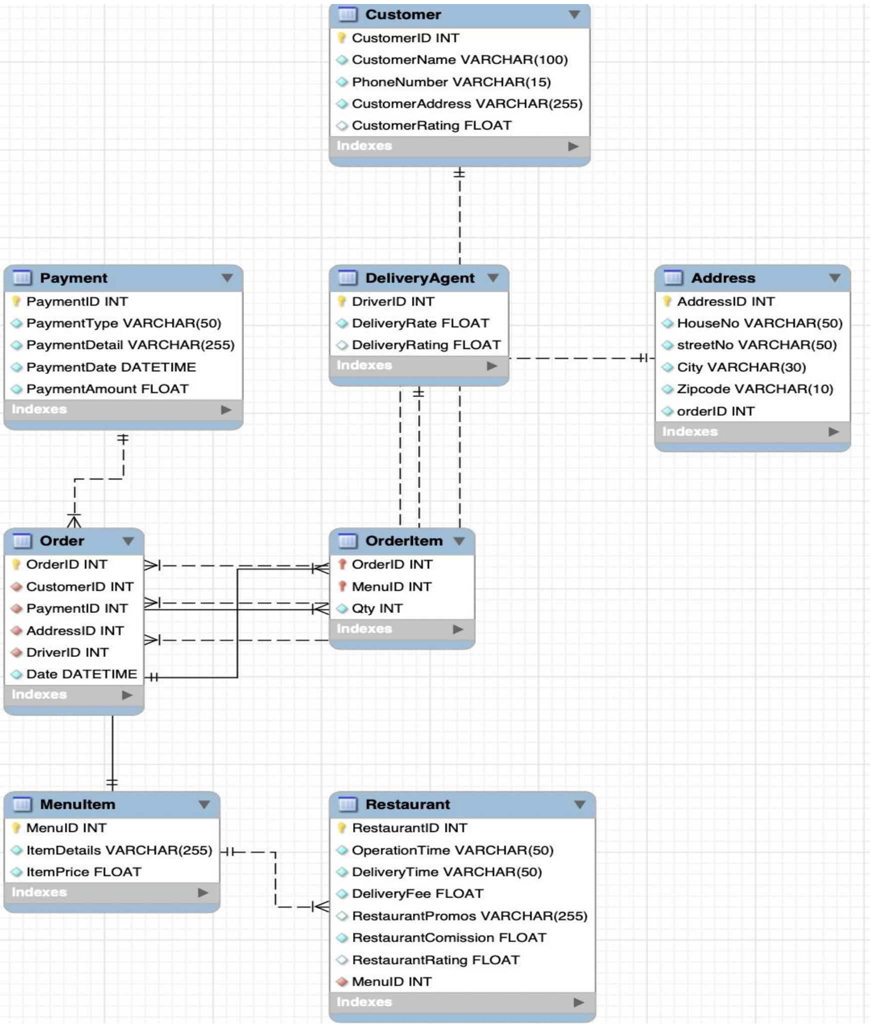
MYSQL WORKBENCH CODES



	OrderID	CustomerID	PaymentID	AddressID	DriverID	Date
▶	1	1	2	1	1	2023-02-14 17:30:00
	2	2	1	2	2	2023-02-14 18:45:00
	3	3	3	3	1	2023-02-14 19:30:00
	4	2	2	4	3	2023-02-14 20:15:00
	5	1	1	5	2	2023-02-14 21:00:00
	6	1	2	1	1	2023-02-14 12:00:00



	CustomerID	CustomerName	PhoneNumber	CustomerAddress	CustomerRating
▶	1	John Smith	123-456-7890	123 Main St, Anytown USA	4.5
	2	Jane Doe	555-555-5555	555 Elm St, Anytown USA	3.2
	3	Sarah Lee	987-654-3210	456 Main St, Anytown USA	4.8
	4	David Kim	111-222-3333	789 Elm St, Anytown USA	3.7
	5	Karen Wong	444-555-6666	123 Maple St, Anytown USA	4.2
	NULL	NULL	NULL	NULL	NULL



Customer
CustomerID INT
CustomerName VARCHAR(100)
PhoneNumber VARCHAR(15)
CustomerAddress VARCHAR(255)
CustomerRating FLOAT
Indexes

Payment
PaymentID INT
PaymentType VARCHAR(50)
PaymentDetail VARCHAR(255)
PaymentDate DATETIME
PaymentAmount FLOAT
Indexes

DeliveryAgent
DriverID INT
DeliveryRate FLOAT
DeliveryRating FLOAT
Indexes

Address
AddressID INT
HouseNo VARCHAR(50)
streetNo VARCHAR(50)
City VARCHAR(30)
Zipcode VARCHAR(10)
orderID INT
Indexes

Order
OrderID INT
CustomerID INT
PaymentID INT
AddressID INT
DriverID INT
Date DATETIME
Indexes

OrderItem
OrderID INT
MenuID INT
Qty INT
Indexes

MenuItem
MenuID INT
ItemDetails VARCHAR(255)
ItemPrice FLOAT
Indexes

Restaurant
RestaurantID INT
OperationTime VARCHAR(50)
DeliveryTime VARCHAR(50)
DeliveryFee FLOAT
RestaurantPromos VARCHAR(255)
RestaurantComission FLOAT
RestaurantRating FLOAT
MenuID INT
Indexes

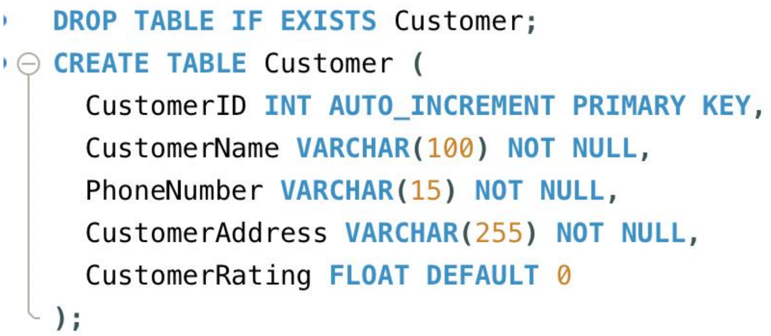
How to Create the Database

This SQL code creates a database schema for a food ordering system.

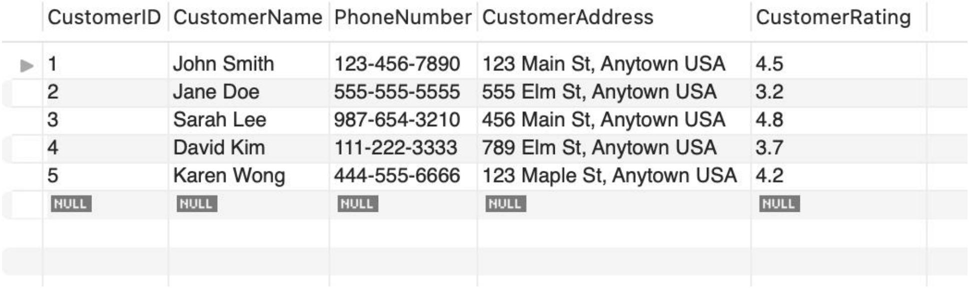
The schema includes tables for storing customer information, menu items and their prices, payment details, delivery agents, restaurant information, addresses, orders, and order items.

Customer table

The Customer table stores information about customers, including their ID, name, phone number, address, and rating and customer id is the primary key.



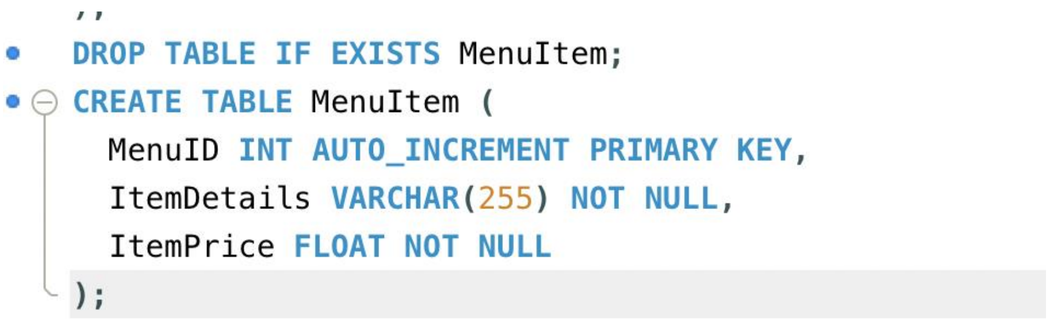
```
DROP TABLE IF EXISTS Customer;
CREATE TABLE Customer (
  CustomerID INT AUTO_INCREMENT PRIMARY KEY,
  CustomerName VARCHAR(100) NOT NULL,
  PhoneNumber VARCHAR(15) NOT NULL,
  CustomerAddress VARCHAR(255) NOT NULL,
  CustomerRating FLOAT DEFAULT 0
);
```



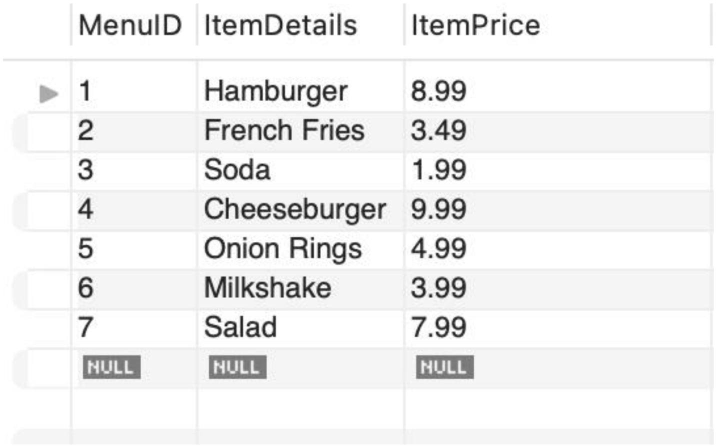
CustomerID	CustomerName	PhoneNumber	CustomerAddress	CustomerRating
1	John Smith	123-456-7890	123 Main St, Anytown USA	4.5
2	Jane Doe	555-555-5555	555 Elm St, Anytown USA	3.2
3	Sarah Lee	987-654-3210	456 Main St, Anytown USA	4.8
4	David Kim	111-222-3333	789 Elm St, Anytown USA	3.7
5	Karen Wong	444-555-6666	123 Maple St, Anytown USA	4.2
NULL	NULL	NULL	NULL	NULL

MenuItem table

The MenuItem table stores details about the available menu items, including the ID, name, and price and menu id is the primary key.



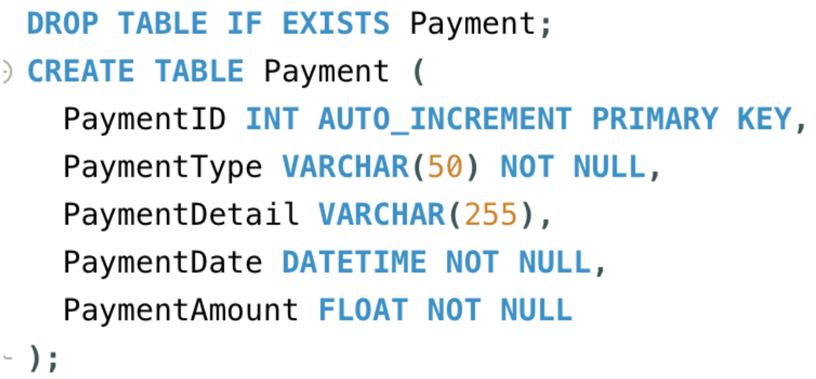
```
,,
DROP TABLE IF EXISTS MenuItem;
CREATE TABLE MenuItem (
  MenuID INT AUTO_INCREMENT PRIMARY KEY,
  ItemDetails VARCHAR(255) NOT NULL,
  ItemPrice FLOAT NOT NULL
);
```



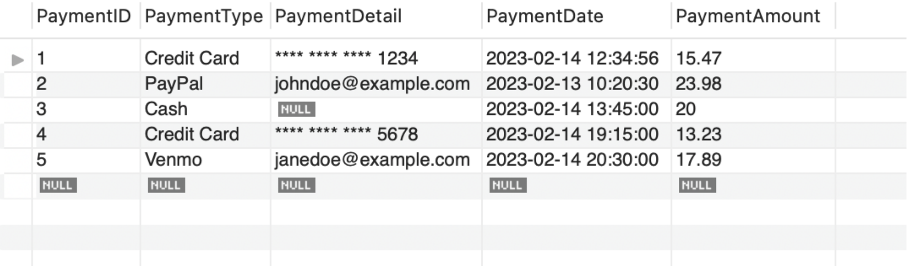
MenuID	ItemDetails	ItemPrice
1	Hamburger	8.99
2	French Fries	3.49
3	Soda	1.99
4	Cheeseburger	9.99
5	Onion Rings	4.99
6	Milkshake	3.99
7	Salad	7.99
NULL	NULL	NULL

Payment table

The Payment table stores details about the payments made by customers, including the ID, payment type, payment details, payment date, and amount and payment id is the primary key.



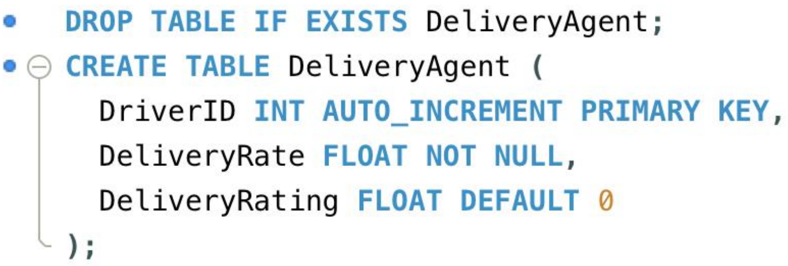
```
DROP TABLE IF EXISTS Payment;
CREATE TABLE Payment (
  PaymentID INT AUTO_INCREMENT PRIMARY KEY,
  PaymentType VARCHAR(50) NOT NULL,
  PaymentDetail VARCHAR(255),
  PaymentDate DATETIME NOT NULL,
  PaymentAmount FLOAT NOT NULL
);
```



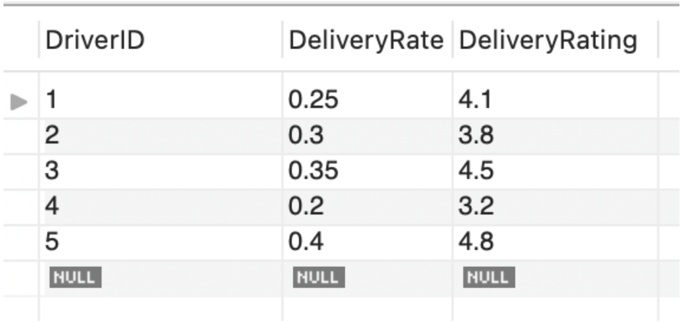
PaymentID	PaymentType	PaymentDetail	PaymentDate	PaymentAmount
1	Credit Card	**** **** **** 1234	2023-02-14 12:34:56	15.47
2	PayPal	johndoe@example.com	2023-02-13 10:20:30	23.98
3	Cash	NULL	2023-02-14 13:45:00	20
4	Credit Card	**** **** **** 5678	2023-02-14 19:15:00	13.23
5	Venmo	janedoe@example.com	2023-02-14 20:30:00	17.89
NULL	NULL	NULL	NULL	NULL

DeliveryAgent table

The DeliveryAgent table stores information about the delivery agents, including their ID, delivery rate, and rating and driver id is the primary key.



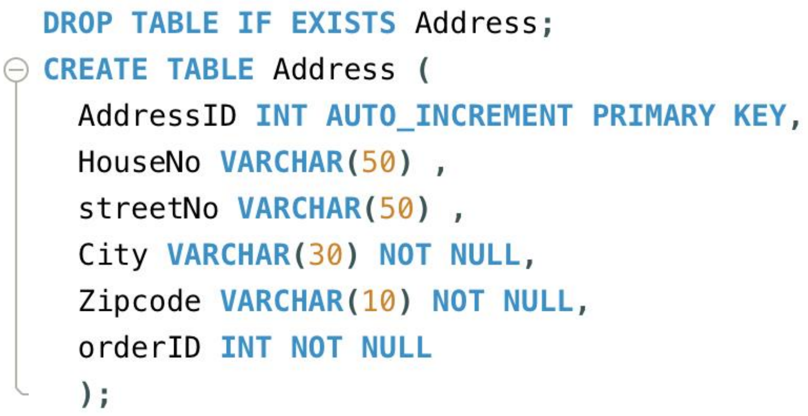
```
DROP TABLE IF EXISTS DeliveryAgent;
CREATE TABLE DeliveryAgent (
    DriverID INT AUTO_INCREMENT PRIMARY KEY,
    DeliveryRate FLOAT NOT NULL,
    DeliveryRating FLOAT DEFAULT 0
);
```



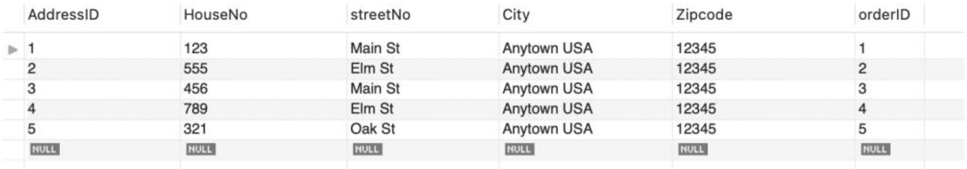
DriverID	DeliveryRate	DeliveryRating
1	0.25	4.1
2	0.3	3.8
3	0.35	4.5
4	0.2	3.2
5	0.4	4.8
NULL	NULL	NULL

Address table

The Address table stores information about the delivery addresses, including the ID, house number, street number, city, zip code, and associated order ID and address id is the primary key.



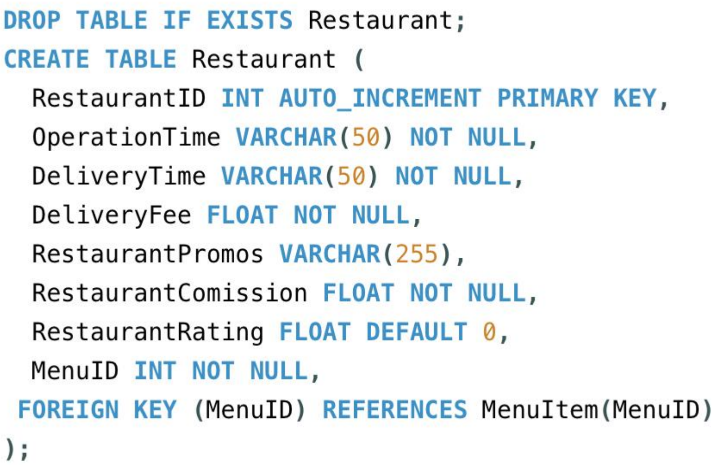
```
DROP TABLE IF EXISTS Address;
CREATE TABLE Address (
  AddressID INT AUTO_INCREMENT PRIMARY KEY,
  HouseNo VARCHAR(50) ,
  streetNo VARCHAR(50) ,
  City VARCHAR(30) NOT NULL,
  Zipcode VARCHAR(10) NOT NULL,
  orderID INT NOT NULL
  );
```



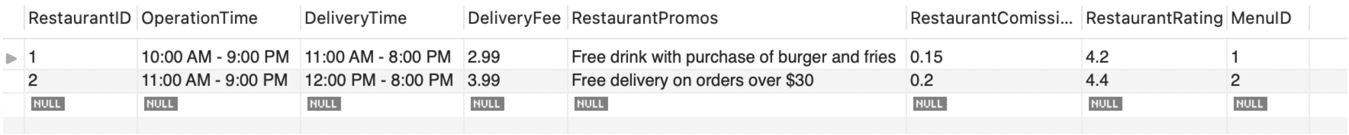
AddressID	HouseNo	streetNo	City	Zipcode	orderID
1	123	Main St	Anytown USA	12345	1
2	555	Elm St	Anytown USA	12345	2
3	456	Main St	Anytown USA	12345	3
4	789	Elm St	Anytown USA	12345	4
5	321	Oak St	Anytown USA	12345	5
NULL	NULL	NULL	NULL	NULL	NULL

Restaurant table

The Restaurant table stores information about the restaurants, including the ID, operation time, delivery time, delivery fee, restaurant promotions, commission rate, rating, and associated menu ID and restaurant id is the primary key.



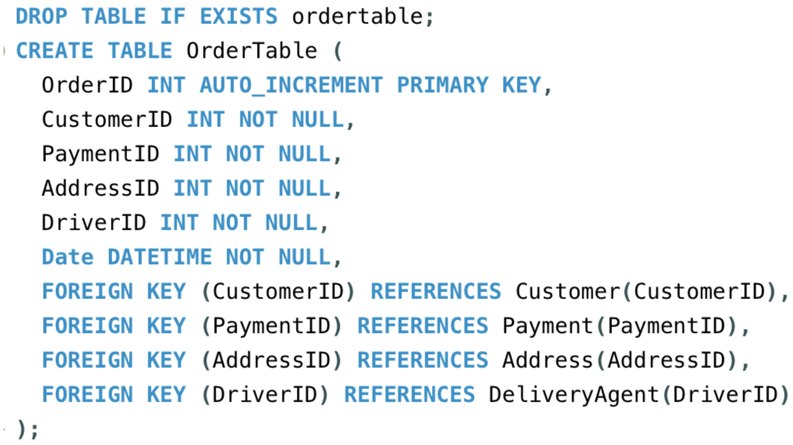
```
DROP TABLE IF EXISTS Restaurant;
CREATE TABLE Restaurant (
  RestaurantID INT AUTO_INCREMENT PRIMARY KEY,
  OperationTime VARCHAR(50) NOT NULL,
  DeliveryTime VARCHAR(50) NOT NULL,
  DeliveryFee FLOAT NOT NULL,
  RestaurantPromos VARCHAR(255),
  RestaurantComission FLOAT NOT NULL,
  RestaurantRating FLOAT DEFAULT 0,
  MenuID INT NOT NULL,
 FOREIGN KEY (MenuID) REFERENCES MenuItem(MenuID)
);
```



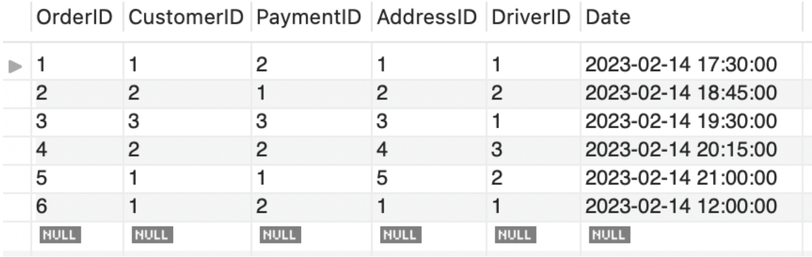
RestaurantID	OperationTime	DeliveryTime	DeliveryFee	RestaurantPromos	RestaurantComissi...	RestaurantRating	MenuID
1	10:00 AM - 9:00 PM	11:00 AM - 8:00 PM	2.99	Free drink with purchase of burger and fries	0.15	4.2	1
2	11:00 AM - 9:00 PM	12:00 PM - 8:00 PM	3.99	Free delivery on orders over $30	0.2	4.4	2
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

OrderTable table

The OrderTable table stores information about the orders made by customers, including the ID, customer ID, payment ID, address ID, delivery agent ID, and date and order id is the primary key.



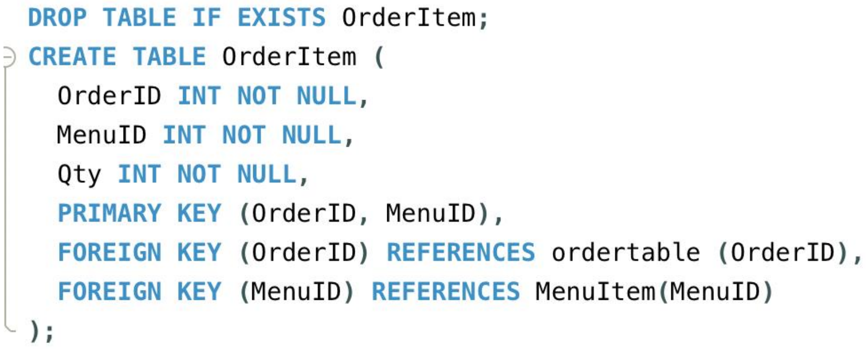
```
DROP TABLE IF EXISTS ordertable;
CREATE TABLE OrderTable (
  OrderID INT AUTO_INCREMENT PRIMARY KEY,
  CustomerID INT NOT NULL,
  PaymentID INT NOT NULL,
  AddressID INT NOT NULL,
  DriverID INT NOT NULL,
  Date DATETIME NOT NULL,
  FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID),
  FOREIGN KEY (PaymentID) REFERENCES Payment(PaymentID),
  FOREIGN KEY (AddressID) REFERENCES Address(AddressID),
  FOREIGN KEY (DriverID) REFERENCES DeliveryAgent(DriverID)
);
```



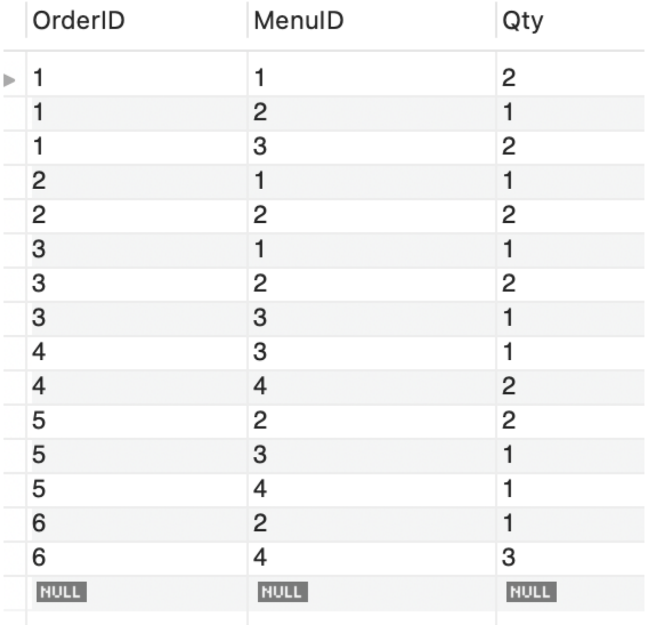
OrderID	CustomerID	PaymentID	AddressID	DriverID	Date
1	1	2	1	1	2023-02-14 17:30:00
2	2	1	2	2	2023-02-14 18:45:00
3	3	3	3	1	2023-02-14 19:30:00
4	2	2	4	3	2023-02-14 20:15:00
5	1	1	5	2	2023-02-14 21:00:00
6	1	2	1	1	2023-02-14 12:00:00
NULL	NULL	NULL	NULL	NULL	NULL

OrderItem table

The OrderItem table stores information about the items in each order, including the order ID, menu ID, and quantity. Order id and menu id are primary keys.



```
DROP TABLE IF EXISTS OrderItem;
CREATE TABLE OrderItem (
  OrderID INT NOT NULL,
  MenuID INT NOT NULL,
  Qty INT NOT NULL,
  PRIMARY KEY (OrderID, MenuID),
  FOREIGN KEY (OrderID) REFERENCES ordertable (OrderID),
  FOREIGN KEY (MenuID) REFERENCES MenuItem(MenuID)
);
```



OrderID	MenuID	Qty
1	1	2
1	2	1
1	3	2
2	1	1
2	2	2
3	1	1
3	2	2
3	3	1
4	3	1
4	4	2
5	2	2
5	3	1
5	4	1
6	2	1
6	4	3
NULL	NULL	NULL

MYSQL MODIFICATION QUERIES

How to Modify the Database

Following are the modification queries for MYSQL.

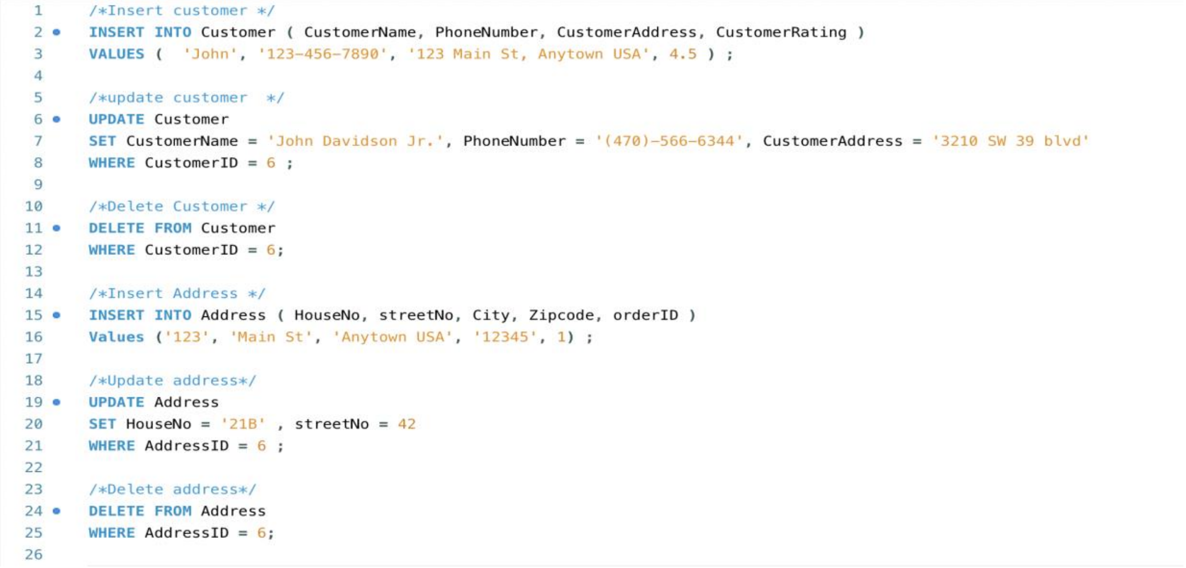
1) Insert
2) Update
3) Delete

In total there are 8 entities in the project and for the same tables have been created assigning the values to them.

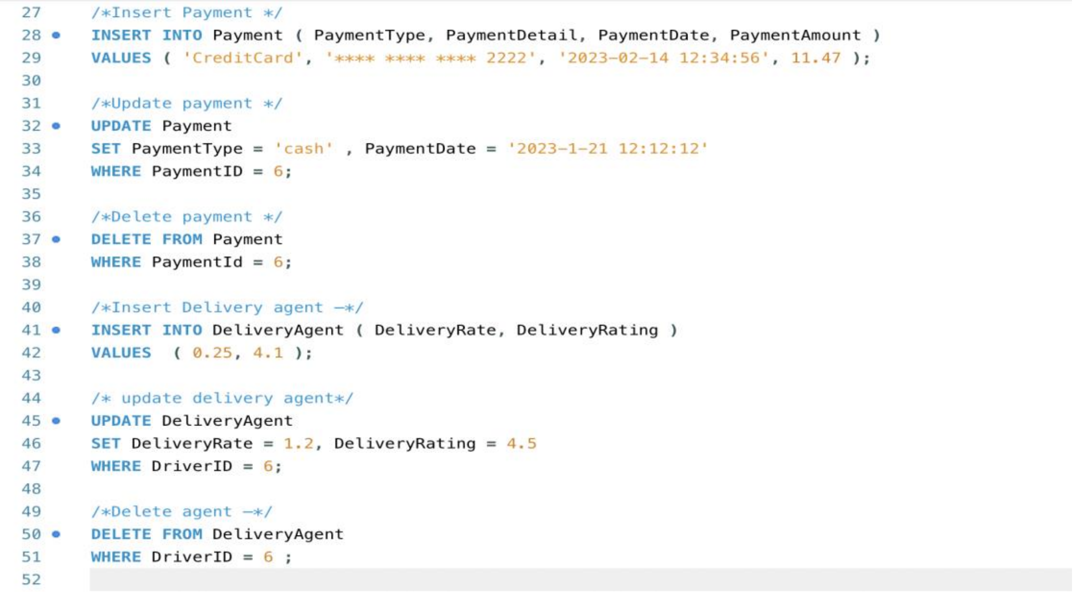
The modification codes allow to make changes in the table and that could be done either by:

1. Inserting some data in the table through the insert query
2. Secondly, any data that needs to be updated within the table can be updated through the update query.
3. Thirdly the delete query helps in removing all the data that is not required.

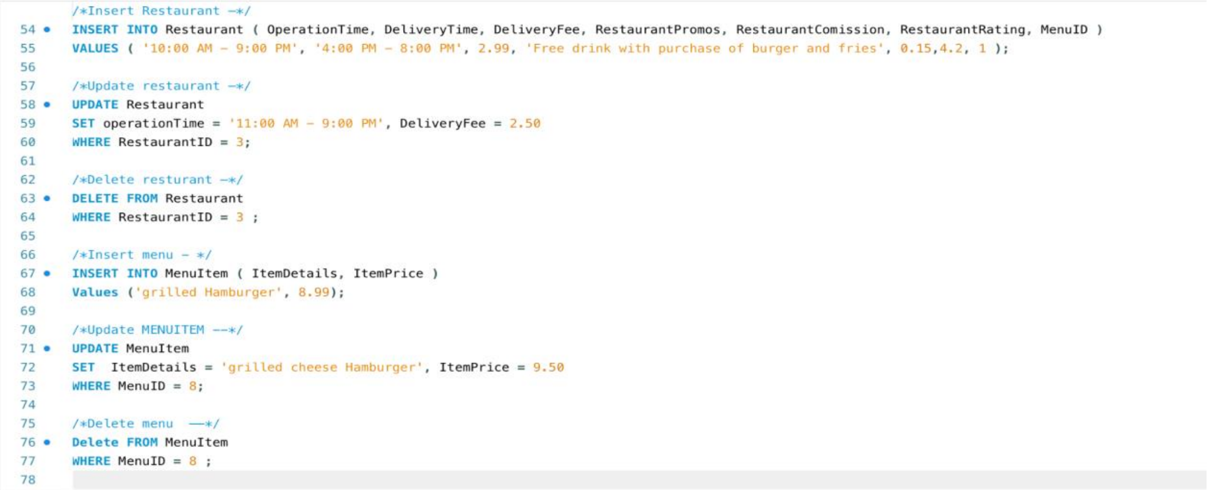
In the code below we have selected a particular entity starting from customer to orderitem and made all three modifications to them (i.e. - insert, update, and delete).



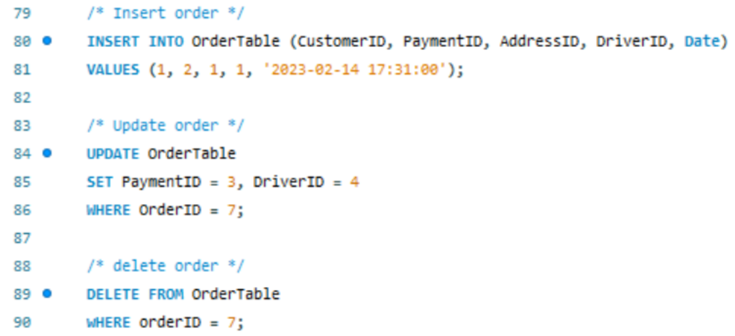
```sql
/*Insert customer */
INSERT INTO Customer ( CustomerName, PhoneNumber, CustomerAddress, CustomerRating )
VALUES (  'John', '123-456-7890', '123 Main St, Anytown USA', 4.5 ) ;

/*update customer  */
UPDATE Customer
SET CustomerName = 'John Davidson Jr.', PhoneNumber = '(470)-566-6344', CustomerAddress = '3210 SW 39 blvd'
WHERE CustomerID = 6 ;

/*Delete Customer */
DELETE FROM Customer
WHERE CustomerID = 6;

/*Insert Address */
INSERT INTO Address ( HouseNo, streetNo, City, Zipcode, orderID )
Values ('123', 'Main St', 'Anytown USA', '12345', 1) ;

/*Update address*/
UPDATE Address
SET HouseNo = '21B' , streetNo = 42
WHERE AddressID = 6 ;

/*Delete address*/
DELETE FROM Address
WHERE AddressID = 6;
```



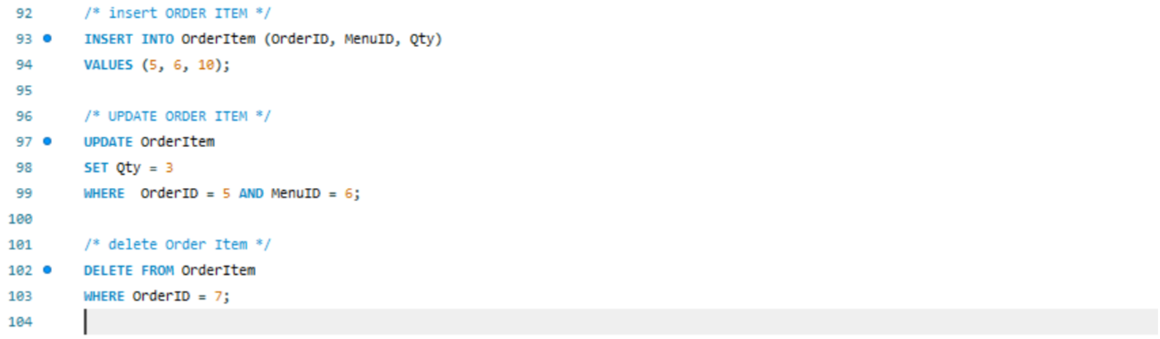
```sql
/*Insert Payment */
INSERT INTO Payment ( PaymentType, PaymentDetail, PaymentDate, PaymentAmount )
VALUES ( 'CreditCard', '**** **** **** 2222', '2023-02-14 12:34:56', 11.47 );

/*Update payment */
UPDATE Payment
SET PaymentType = 'cash' , PaymentDate = '2023-1-21 12:12:12'
WHERE PaymentID = 6;

/*Delete payment */
DELETE FROM Payment
WHERE PaymentId = 6;

/*Insert Delivery agent -*/
INSERT INTO DeliveryAgent ( DeliveryRate, DeliveryRating )
VALUES  ( 0.25, 4.1 );

/* update delivery agent*/
UPDATE DeliveryAgent
SET DeliveryRate = 1.2, DeliveryRating = 4.5
WHERE DriverID = 6;

/*Delete agent -*/
DELETE FROM DeliveryAgent
WHERE DriverID = 6 ;

```



```sql
/*Insert Restaurant -*/
INSERT INTO Restaurant ( OperationTime, DeliveryTime, DeliveryFee, RestaurantPromos, RestaurantComission, RestaurantRating, MenuID )
VALUES ( '10:00 AM - 9:00 PM', '4:00 PM - 8:00 PM', 2.99, 'Free drink with purchase of burger and fries', 0.15,4.2, 1 );

/*Update restaurant -*/
UPDATE Restaurant
SET operationTime = '11:00 AM - 9:00 PM', DeliveryFee = 2.50
WHERE RestaurantID = 3;

/*Delete resturant -*/
DELETE FROM Restaurant
WHERE RestaurantID = 3 ;

/*Insert menu - */
INSERT INTO MenuItem ( ItemDetails, ItemPrice )
Values ('grilled Hamburger', 8.99);

/*Update MENUITEM --*/
UPDATE MenuItem
SET  ItemDetails = 'grilled cheese Hamburger', ItemPrice = 9.50
WHERE MenuID = 8;

/*Delete menu  --*/
Delete FROM MenuItem
WHERE MenuID = 8 ;

```



```sql
/* Insert order */
INSERT INTO OrderTable (CustomerID, PaymentID, AddressID, DriverID, Date)
VALUES (1, 2, 1, 1, '2023-02-14 17:31:00');

/* Update order */
UPDATE OrderTable
SET PaymentID = 3, DriverID = 4
WHERE OrderID = 7;

/* delete order */
DELETE FROM OrderTable
WHERE orderID = 7;
```



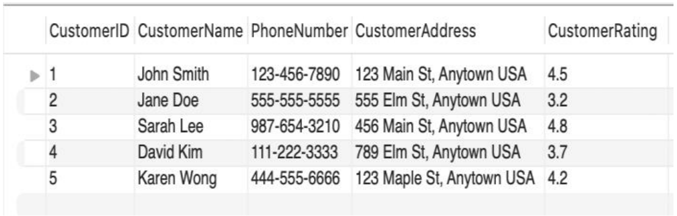
```
/* insert ORDER ITEM */
INSERT INTO OrderItem (OrderID, MenuID, Qty)
VALUES (5, 6, 10);

/* UPDATE ORDER ITEM */
UPDATE OrderItem
SET Qty = 3
WHERE  OrderID = 5 AND MenuID = 6;

/* delete Order Item */
DELETE FROM OrderItem
WHERE OrderID = 7;
```

Insert Query

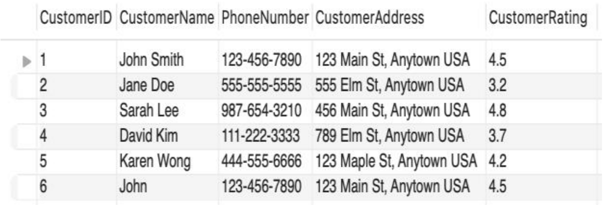
Following are the examples of changes that were been made to the tables after the insert query was used in the MYSQL

1) For the first table i.e. customer in total there were 5 values and after the insert query, a 6th value was added(John).
2) Secondly, for the Menu table the 8th value has been included in the table (i.e. the grilled hamburger).

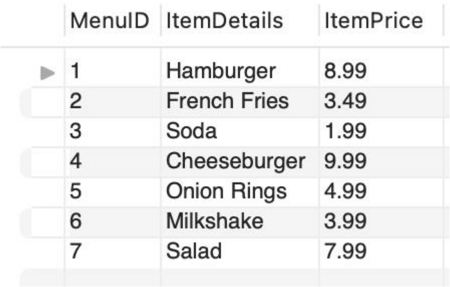
Before **After**



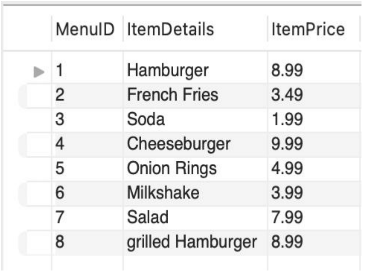
CustomerID	CustomerName	PhoneNumber	CustomerAddress	CustomerRating
1	John Smith	123-456-7890	123 Main St, Anytown USA	4.5
2	Jane Doe	555-555-5555	555 Elm St, Anytown USA	3.2
3	Sarah Lee	987-654-3210	456 Main St, Anytown USA	4.8
4	David Kim	111-222-3333	789 Elm St, Anytown USA	3.7
5	Karen Wong	444-555-6666	123 Maple St, Anytown USA	4.2



CustomerID	CustomerName	PhoneNumber	CustomerAddress	CustomerRating
1	John Smith	123-456-7890	123 Main St, Anytown USA	4.5
2	Jane Doe	555-555-5555	555 Elm St, Anytown USA	3.2
3	Sarah Lee	987-654-3210	456 Main St, Anytown USA	4.8
4	David Kim	111-222-3333	789 Elm St, Anytown USA	3.7
5	Karen Wong	444-555-6666	123 Maple St, Anytown USA	4.2
6	John	123-456-7890	123 Main St, Anytown USA	4.5



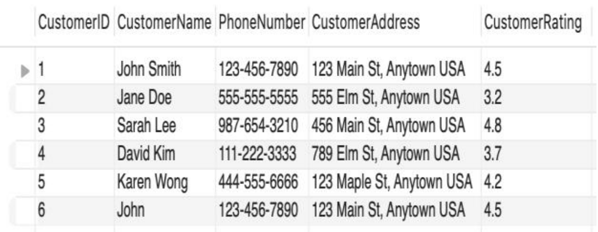
MenuID	ItemDetails	ItemPrice
1	Hamburger	8.99
2	French Fries	3.49
3	Soda	1.99
4	Cheeseburger	9.99
5	Onion Rings	4.99
6	Milkshake	3.99
7	Salad	7.99



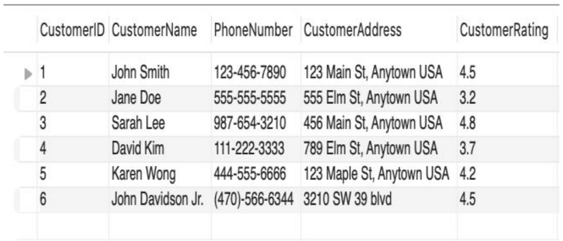
MenuID	ItemDetails	ItemPrice
1	Hamburger	8.99
2	French Fries	3.49
3	Soda	1.99
4	Cheeseburger	9.99
5	Onion Rings	4.99
6	Milkshake	3.99
7	Salad	7.99
8	grilled Hamburger	8.99

Update Query

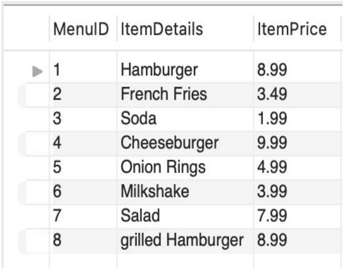
1) For the update query for the customer table the data for “john” changed with all other details.
2) Secondly, for the menu table the name gets updated for the 8th value after the update query is executed



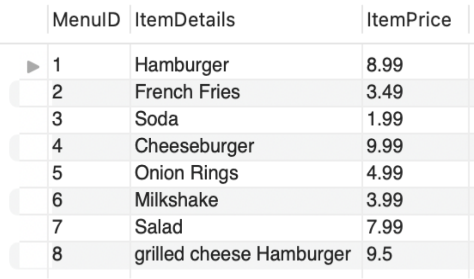
CustomerID	CustomerName	PhoneNumber	CustomerAddress	CustomerRating
1	John Smith	123-456-7890	123 Main St, Anytown USA	4.5
2	Jane Doe	555-555-5555	555 Elm St, Anytown USA	3.2
3	Sarah Lee	987-654-3210	456 Main St, Anytown USA	4.8
4	David Kim	111-222-3333	789 Elm St, Anytown USA	3.7
5	Karen Wong	444-555-6666	123 Maple St, Anytown USA	4.2
6	John	123-456-7890	123 Main St, Anytown USA	4.5



CustomerID	CustomerName	PhoneNumber	CustomerAddress	CustomerRating
1	John Smith	123-456-7890	123 Main St, Anytown USA	4.5
2	Jane Doe	555-555-5555	555 Elm St, Anytown USA	3.2
3	Sarah Lee	987-654-3210	456 Main St, Anytown USA	4.8
4	David Kim	111-222-3333	789 Elm St, Anytown USA	3.7
5	Karen Wong	444-555-6666	123 Maple St, Anytown USA	4.2
6	John Davidson Jr.	(470)-566-6344	3210 SW 39 blvd	4.5



MenuID	ItemDetails	ItemPrice
1	Hamburger	8.99
2	French Fries	3.49
3	Soda	1.99
4	Cheeseburger	9.99
5	Onion Rings	4.99
6	Milkshake	3.99
7	Salad	7.99
8	grilled Hamburger	8.99

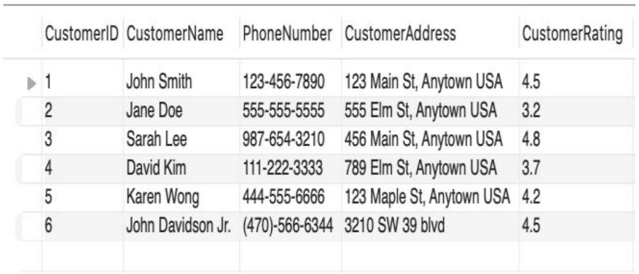


MenuID	ItemDetails	ItemPrice
1	Hamburger	8.99
2	French Fries	3.49
3	Soda	1.99
4	Cheeseburger	9.99
5	Onion Rings	4.99
6	Milkshake	3.99
7	Salad	7.99
8	grilled cheese Hamburger	9.5

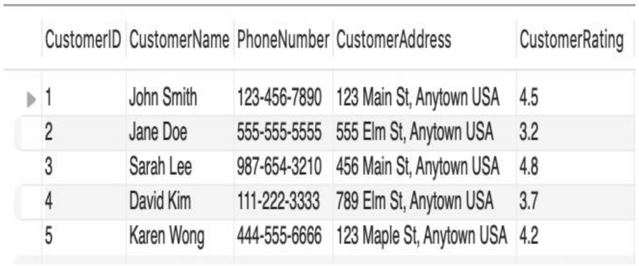
Delete Query

1) For the delete query the customer table the 6th value (i.e. - john davidson jr) gets deleted after we execute the delete query.
2) And for the Menu table it's the same thing where in the 8th value (i.e. - grilled cheese hamburger) it gets deleted after the query is executed.

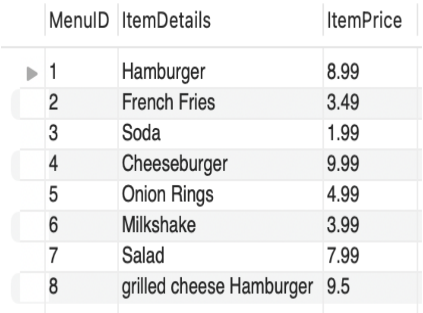
Before **After**



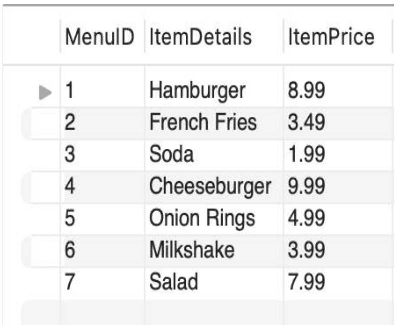
CustomerID	CustomerName	PhoneNumber	CustomerAddress	CustomerRating
1	John Smith	123-456-7890	123 Main St, Anytown USA	4.5
2	Jane Doe	555-555-5555	555 Elm St, Anytown USA	3.2
3	Sarah Lee	987-654-3210	456 Main St, Anytown USA	4.8
4	David Kim	111-222-3333	789 Elm St, Anytown USA	3.7
5	Karen Wong	444-555-6666	123 Maple St, Anytown USA	4.2
6	John Davidson Jr.	(470)-566-6344	3210 SW 39 blvd	4.5



CustomerID	CustomerName	PhoneNumber	CustomerAddress	CustomerRating
1	John Smith	123-456-7890	123 Main St, Anytown USA	4.5
2	Jane Doe	555-555-5555	555 Elm St, Anytown USA	3.2
3	Sarah Lee	987-654-3210	456 Main St, Anytown USA	4.8
4	David Kim	111-222-3333	789 Elm St, Anytown USA	3.7
5	Karen Wong	444-555-6666	123 Maple St, Anytown USA	4.2



MenuID	ItemDetails	ItemPrice
1	Hamburger	8.99
2	French Fries	3.49
3	Soda	1.99
4	Cheeseburger	9.99
5	Onion Rings	4.99
6	Milkshake	3.99
7	Salad	7.99
8	grilled cheese Hamburger	9.5

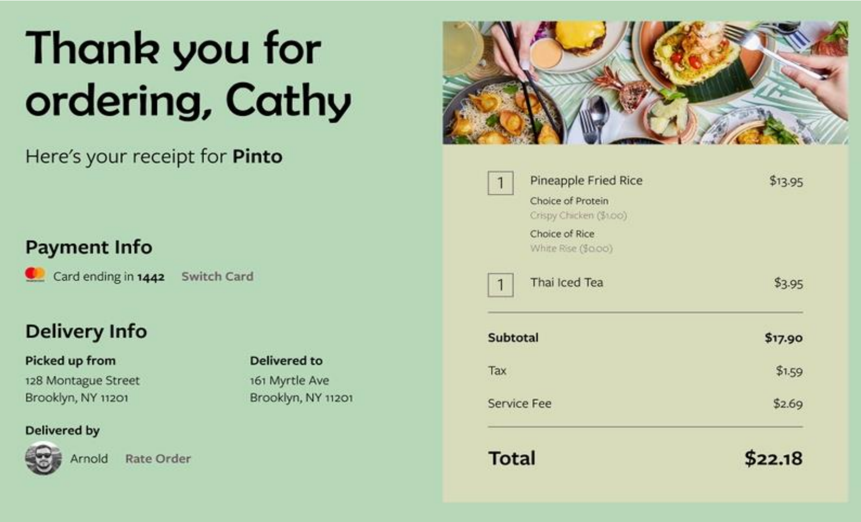


MenuID	ItemDetails	ItemPrice
1	Hamburger	8.99
2	French Fries	3.49
3	Soda	1.99
4	Cheeseburger	9.99
5	Onion Rings	4.99
6	Milkshake	3.99
7	Salad	7.99

Example SQL Queries

Example Query 1: Order Receipt

Querying a customer's account for an order receipt can be considered one of the most common queries in a food delivery service app. Many customers check their receipts after they have placed their order to double-check if all of their information and charges have gone through correctly. A customer's order receipt can include their name, the restaurant's name, order date and time, name and quantity of food items, payment information and total, delivery address, and delivery driver. Though, more or less of these items can be included. A mockup example of a food delivery service receipt is shown below:



Thank you for ordering, Cathy

Here's your receipt for **Pinto**

Payment Info

Card ending in **1442** Switch Card

Delivery Info

Picked up from
128 Montague Street
Brooklyn, NY 11201

Delivered to
161 Myrtle Ave
Brooklyn, NY 11201

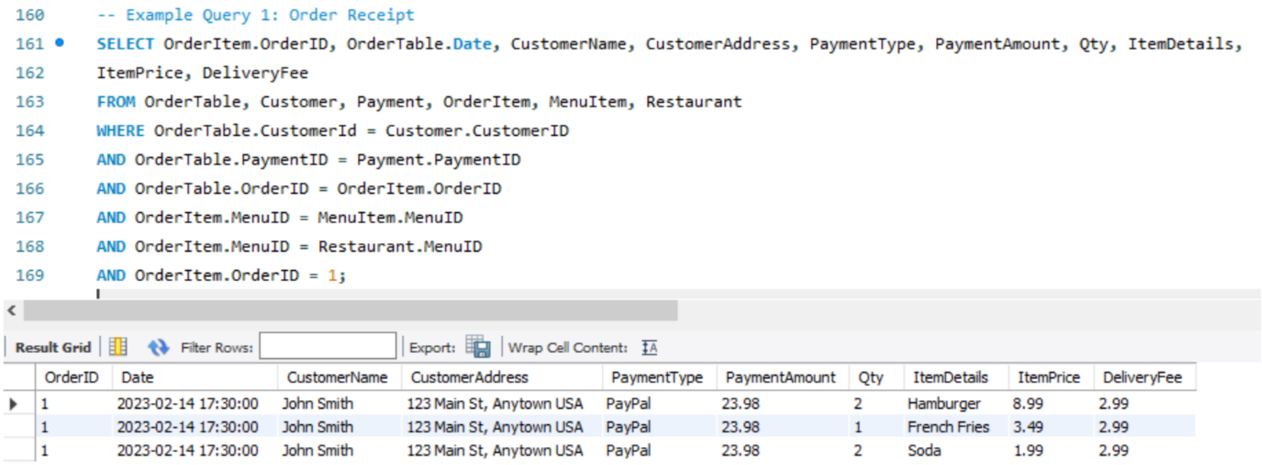
Delivered by

Arnold Rate Order

1	Pineapple Fried Rice Choice of Protein Crispy Chicken ($1.00) Choice of Rice White Rise ($0.00)	$13.95
1	Thai Iced Tea	$3.95
	Subtotal	**$17.90**
	Tax	$1.59
	Service Fee	$2.69
	Total	**$22.18**

.

SQL Code & Query Result:



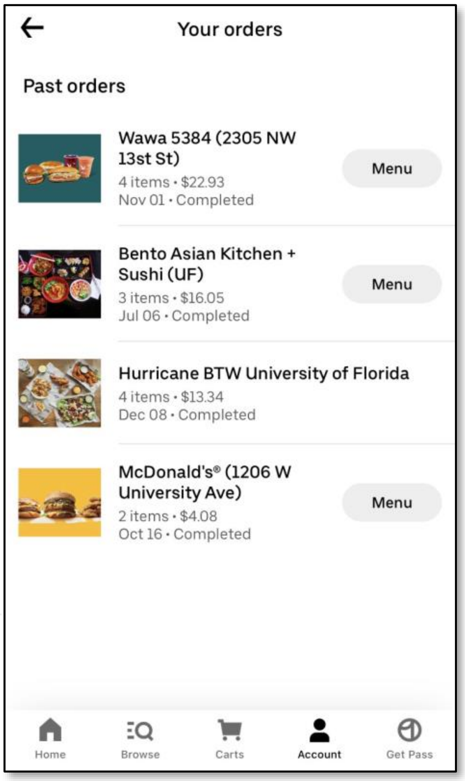
```
-- Example Query 1: Order Receipt
SELECT OrderItem.OrderID, OrderTable.Date, CustomerName, CustomerAddress, PaymentType, PaymentAmount, Qty, ItemDetails,
ItemPrice, DeliveryFee
FROM OrderTable, Customer, Payment, OrderItem, MenuItem, Restaurant
WHERE OrderTable.CustomerId = Customer.CustomerID
AND OrderTable.PaymentID = Payment.PaymentID
AND OrderTable.OrderID = OrderItem.OrderID
AND OrderItem.MenuID = MenuItem.MenuID
AND OrderItem.MenuID = Restaurant.MenuID
AND OrderItem.OrderID = 1;
```

OrderID	Date	CustomerName	CustomerAddress	PaymentType	PaymentAmount	Qty	ItemDetails	ItemPrice	DeliveryFee
1	2023-02-14 17:30:00	John Smith	123 Main St, Anytown USA	PayPal	23.98	2	Hamburger	8.99	2.99
1	2023-02-14 17:30:00	John Smith	123 Main St, Anytown USA	PayPal	23.98	1	French Fries	3.49	2.99
1	2023-02-14 17:30:00	John Smith	123 Main St, Anytown USA	PayPal	23.98	2	Soda	1.99	2.99

Our example of an order receipt contains the order ID, date/time stamp, the customer's name, the customer's address, payment type, payment amount, quantity and names of food items ordered, each food item's price, and the delivery fee. Generating an order receipt takes information from six different tables: OrderTable, Customer, Payment, OrderItem, MenuItem, and Restaurant. Therefore, to produce all of this information into one query, many table joins are used. Then to select a receipt from a specific order, the condition OrderItem.OrderID = 1 is used to specify the retrieval of information from only OrderID 1.

Example Query 2: Customer Order History

Similar to an order receipt, a customer may wish to view all of their previous orders made on their account. To do this, a query for customer order history is enabled within the customer's account on the app. This function is enabled so that a customer can view their history to recreate one of their past orders. In another case, they may want to check how many orders they have made within the past week, for example. In the example, order history includes all past orders made by one customer, the quantity of items ordered, restaurant names, and order dates. Though, order history can include more or less of these items. See the image to the right, demonstrating the information queried when a customer views past orders made under their account.



Your orders

Past orders

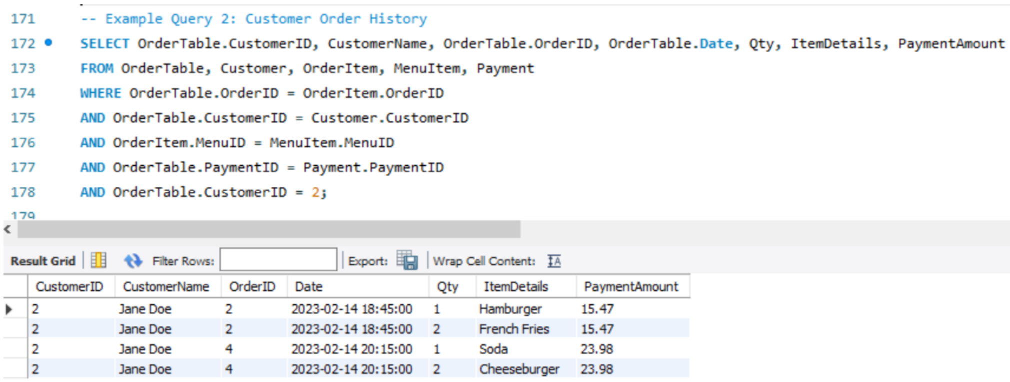
Wawa 5384 (2305 NW 13st St)
4 items • $22.93
Nov 01 • Completed

Bento Asian Kitchen + Sushi (UF)
3 items • $16.05
Jul 06 • Completed

Hurricane BTW University of Florida
4 items • $13.34
Dec 08 • Completed

McDonald's® (1206 W University Ave)
2 items • $4.08
Oct 16 • Completed

SQL Code & Query Result:



```
-- Example Query 2: Customer Order History
SELECT OrderTable.CustomerID, CustomerName, OrderTable.OrderID, OrderTable.Date, Qty, ItemDetails, PaymentAmount
FROM OrderTable, Customer, OrderItem, MenuItem, Payment
WHERE OrderTable.OrderID = OrderItem.OrderID
AND OrderTable.CustomerID = Customer.CustomerID
AND OrderItem.MenuID = MenuItem.MenuID
AND OrderTable.PaymentID = Payment.PaymentID
AND OrderTable.CustomerID = 2;
```

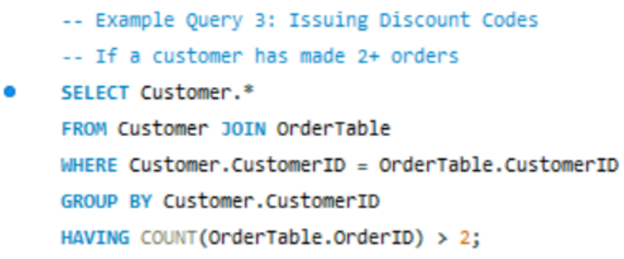
CustomerID	CustomerName	OrderID	Date	Qty	ItemDetails	PaymentAmount
2	Jane Doe	2	2023-02-14 18:45:00	1	Hamburger	15.47
2	Jane Doe	2	2023-02-14 18:45:00	2	French Fries	15.47
2	Jane Doe	4	2023-02-14 20:15:00	1	Soda	23.98
2	Jane Doe	4	2023-02-14 20:15:00	2	Cheeseburger	23.98

Our example of a query of a customer's order history includes the customer's id, customer's name, order id, date/time stamp, quantity and names of the food items ordered, and total payment amount. Producing a customer's order history includes information from five different tables: OrderTable, Customer, OrderItem, MenuItem, and Payment; thus many table joins were utilized. The condition OrderTable.CustomerID = 2 is also specified to retrieve only information from CustomerID 2. In the code above, you can see in Jane's order history she has made two orders, OrderID = 2 and 4, with the first order containing one hamburger and 2 French fries, and the second containing one soda and two cheeseburgers.

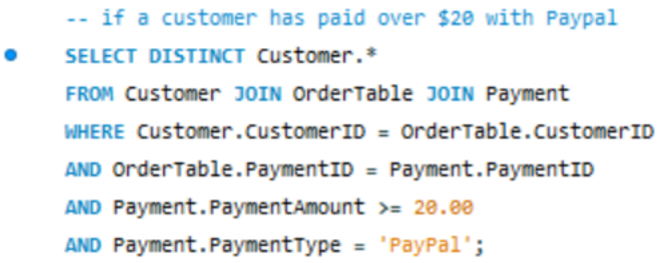
Example Query 3: Issuing Discount Codes

In the backend of the app, certain queries are designed so that once a customer's account reaches certain conditions, discount codes are issued. From a business perspective, marketing to customers so that they can earn certain discounts will incentivize them to use, for example, a different payment associated with a third party such as Paypal. This not only earns the third party a commission but also the food delivery service company by encouraging customers to spend more on their app in the future once they receive the discount code and place a new order with it.

SQL Code Examples:



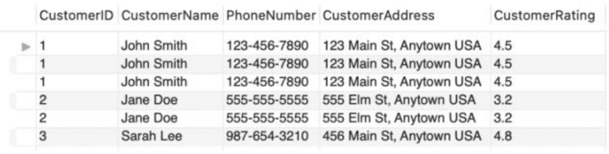
```
-- Example Query 3: Issuing Discount Codes
-- If a customer has made 2+ orders
SELECT Customer.*
FROM Customer JOIN OrderTable
WHERE Customer.CustomerID = OrderTable.CustomerID
GROUP BY Customer.CustomerID
HAVING COUNT(OrderTable.OrderID) > 2;
```



```
-- if a customer has paid over $20 with Paypal
SELECT DISTINCT Customer.*
FROM Customer JOIN OrderTable JOIN Payment
WHERE Customer.CustomerID = OrderTable.CustomerID
AND OrderTable.PaymentID = Payment.PaymentID
AND Payment.PaymentAmount >= 20.00
AND Payment.PaymentType = 'PayPal';
```

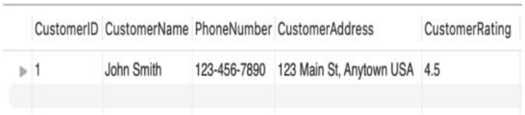
Query Results Before & After:

If a customer has made 2+ orders



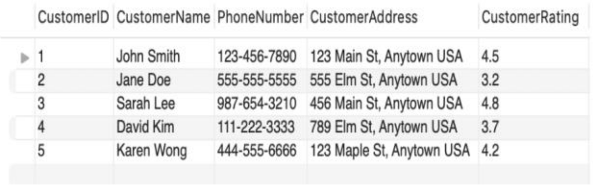
CustomerID	CustomerName	PhoneNumber	CustomerAddress	CustomerRating
1	John Smith	123-456-7890	123 Main St, Anytown USA	4.5
1	John Smith	123-456-7890	123 Main St, Anytown USA	4.5
1	John Smith	123-456-7890	123 Main St, Anytown USA	4.5
2	Jane Doe	555-555-5555	555 Elm St, Anytown USA	3.2
2	Jane Doe	555-555-5555	555 Elm St, Anytown USA	3.2
3	Sarah Lee	987-654-3210	456 Main St, Anytown USA	4.8

→



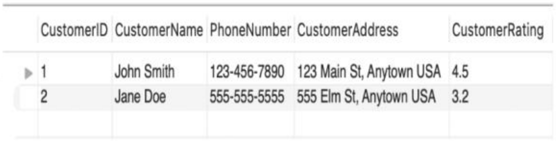
CustomerID	CustomerName	PhoneNumber	CustomerAddress	CustomerRating
1	John Smith	123-456-7890	123 Main St, Anytown USA	4.5

If a customer has paid over $20 with Paypal



CustomerID	CustomerName	PhoneNumber	CustomerAddress	CustomerRating
1	John Smith	123-456-7890	123 Main St, Anytown USA	4.5
2	Jane Doe	555-555-5555	555 Elm St, Anytown USA	3.2
3	Sarah Lee	987-654-3210	456 Main St, Anytown USA	4.8
4	David Kim	111-222-3333	789 Elm St, Anytown USA	3.7
5	Karen Wong	444-555-6666	123 Maple St, Anytown USA	4.2

→



CustomerID	CustomerName	PhoneNumber	CustomerAddress	CustomerRating
1	John Smith	123-456-7890	123 Main St, Anytown USA	4.5
2	Jane Doe	555-555-5555	555 Elm St, Anytown USA	3.2

.

Our two examples of what a query's conditions might look like when a customer gets a discount code issued to them are: when a customer makes two or more orders and when a customer pays over $20 with Paypal. The former of the two, joins the table OrderTable to Customer in order to return a list of customers that have made two or more orders with the condition COUNT(OrderTable.OrderID) > 2. The latter joins the tables Customer, OrderTable, and Payment with the conditions Payment.PaymentAmount >= 20.00 and Payment.PaymentType = 'Paypal', and produces a list of customer information where those conditions apply.

Bibliography:

[1] grubtech (Ed.). (2021, March 8). How did food delivery start? history of food delivery. How did food delivery start? History of food delivery. Retrieved February 13, 2023, from https://blog.grubtech.com/history-of-food-delivery

[2] Hendelmann, V. (2023, January 2). The Food Delivery Business Model – A Complete Guide. productmint. Retrieved February 13, 2023, from https://productmint.com/the-food-delivery-business-model-a-complete-guide/

[3] Spdload. (2023, February 9). Food Delivery Business Model 101 for founders. SpdLoad. Retrieved February 13, 2023, from https://spdload.com/blog/food-delivery-business-model/