mySQLFanclub

Manmitha Neelam
Evan Doran
Jimmy Wu
Jason Golubski

Table of Contents

Description of the Problem

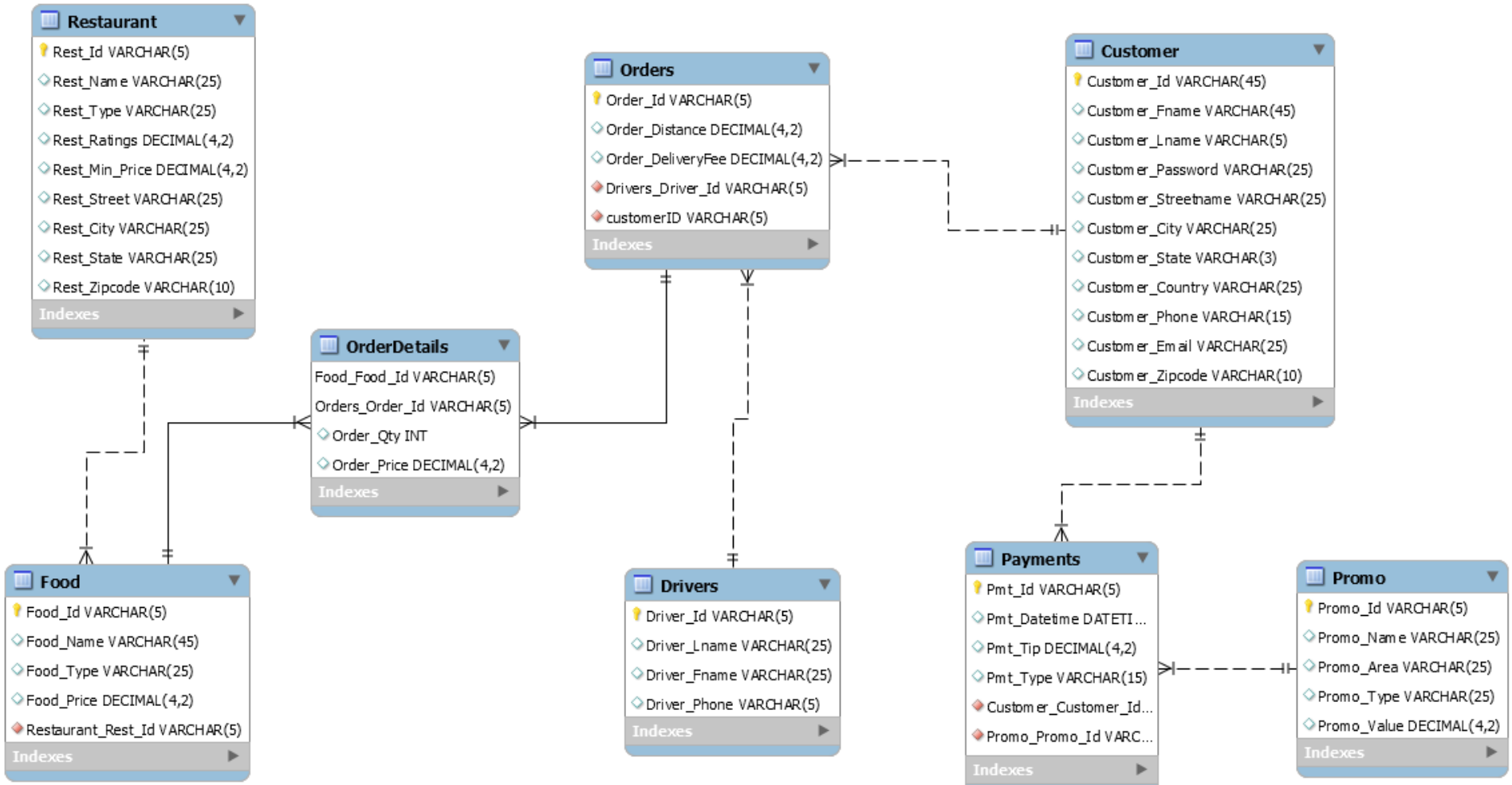
We are trying to model the process of ordering on UberEats from the customer's viewpoint using the mySQL database modeling format. The customer is given a unique id, and his/her first name, last name, password, location, and email are also stored in the database. A customer can have one or many orders. However, one particular order can only belong to one customer. A customer can also have many payments, but a particular payment can only belong to one customer.

A payment of the orders is classified by an id as the primary key and contains the date and time the payment was made, the tip amount of the payment, and how the payment was made. These payments are also associated with promo codes in which one promo code can have many payments, but only one promo code can be used on a payment. The promo code is identified by an id and has a name, an area classified as discounting from the subtotal or the delivery fee, type categorized as a percent or an amount, and a total value of the amount as a decimal after the promo is applied. The values can be used to discount from the total payment of the order.

An order contains an id, which is its primary key, the distance it originally from the customer, and the delivery fee. A particular order also has one driver. On the other hand, drivers can have many orders. Details about the driver stored in the database include the driver id as the primary key, the driver's last and first name, and the driver's phone number.

These orders also have many food items and one particular food item can be in many orders. This relationship is classified by order details that include the quantity of food items ordered and the overall price of the order. The food items are identified by food id, and the database contains information about the name of the food, the type of food, and the price of an individual unit of a food item. All these food items are linked to one particular restaurant. A certain restaurant on the app can have many food items, but a food item can only have one restaurant. The database identifies the restaurant by an id, also its primary key, and includes information about the restaurant name, type, ratings, minimum price of the food that can be delivered, and location.

Data Model


Restaurant
Rest_Id VARCHAR(5)
Rest_Name VARCHAR(25)
Rest_Type VARCHAR(25)
Rest_Ratings DECIMAL(4,2)
Rest_Min_Price DECIMAL(4,2)
Rest_Street VARCHAR(25)
Rest_City VARCHAR(25)
Rest_State VARCHAR(25)
Rest_Zipcode VARCHAR(10)
Indexes
Orders
Order_Id VARCHAR(5)
Order_Distance DECIMAL(4,2)
Order_DeliveryFee DECIMAL(4,2)
Drivers_Driver_Id VARCHAR(5)
customerID VARCHAR(5)
Indexes
Customer
Customer_Id VARCHAR(45)
Customer_Fname VARCHAR(45)
Customer_Lname VARCHAR(5)
Customer_Password VARCHAR(25)
Customer_Streetname VARCHAR(25)
Customer_City VARCHAR(25)
Customer_State VARCHAR(3)
Customer_Country VARCHAR(25)
Customer_Phone VARCHAR(15)
Customer_Email VARCHAR(25)
Customer_Zipcode VARCHAR(10)
Indexes
OrderDetails
Food_Food_Id VARCHAR(5)
Orders_Order_Id VARCHAR(5)
Order_Qty INT
Order_Price DECIMAL(4,2)
Indexes
Food
Food_Id VARCHAR(5)
Food_Name VARCHAR(45)
Food_Type VARCHAR(25)
Food_Price DECIMAL(4,2)
Restaurant_Rest_Id VARCHAR(5)
Indexes
Drivers
Driver_Id VARCHAR(5)
Driver_Lname VARCHAR(25)
Driver_Fname VARCHAR(25)
Driver_Phone VARCHAR(5)
Indexes
Payments
Pmt_Id VARCHAR(5)
Pmt_Datetime DATETI...
Pmt_Tip DECIMAL(4,2)
Pmt_Type VARCHAR(15)
Customer_Customer_Id...
Promo_Promo_Id VARC...
Indexes
Promo
Promo_Id VARCHAR(5)
Promo_Name VARCHAR(25)
Promo_Area VARCHAR(25)
Promo_Type VARCHAR(25)
Promo_Value DECIMAL(4,2)
Indexes

Word Dictionary

Table: **Customer**

Column Name	Description	Data Type	Size	Format	Key?
Customer_Id	Int digit to identify customer	Varchar	5	2	PK
Customer_Fname	Customer's first name	Varchar	25	Jerry	
Customer_Lname	Customer's last name	Varchar	25	Springer	
Customer_Password	Customer's password	Varchar	25	123456	
Customer_Streetname	Customer's street address	Varchar	25	12 Abcd St	
Customer_City	City that customer lives in	Varchar	25	Athens	
Customer_State	State that customer lives in	Varchar	3	GA	
Customer_Country	Country that customer lives in	Varchar	25	United States	
Customer_Phone	Customer's phone number	Varchar	15	1234567890	
Customer_Email	Customer's email address	Varchar	25	jerryspringer@gmail.com	
Customer_Zipcode	Customer's zip code for place of residence	Varchar	10	12345	

Table: **Drivers**

Column Name	Description	Data Type	Size	Format	Key?
Driver_Id	Integer digit assigned to identify driver	Varchar	5	2	PK
Driver_Lname	Driver's last name	Varchar	25	Lincoln	
Driver_Fname	Driver first name	Varchar	25	Abraham	
Driver_Phone	Driver's phone number	Varchar	15	1234567890	

Table: **Food**

Column Name	Description	Data Type	Size	Format	Key?
Food_Id	Unique integer id assigned to individual food items	Varchar	5	12	PK
Food_Name	Food name	Varchar	45	Frat Daddy	
Food_Type	Food type	Varchar	25	Calzone	
Food_Price	Food price of individual unit of that food	Decimal	4,2	1.00	
Restaurant_Rest_Id	Restaurant ID from where the food is from	Varchar	5	12	FK (ref: Restaurant)

Table: **OrderDetails**

Column Name	Description	Data Type	Size	Format	Key?
Food_Food_Id	Reference to unique integer id assigned to individual food items in Food table	Varchar	5	1	Part of PK
Orders_Order_Id	Reference to a unique integer value assigned to identify an individual order	Varchar	5	6	Part of PK
Order_Qty	Quantity of individual food items ordered	Int	5	4	
Order_Price	Price multiplied by quantity of a single product in that order	Decimal	4,2	43.00	

Table: **Orders**

Column Name	Description	Data Type	Size	Format	Key?
Order_Id	Unique integer id to identify individual orders	Varchar	5	12	PK
Order_Distance	Distance from Customer house from Restaurant Ordered from in Miles	Decimal	4,2	1.05	
Order_DeliveryFee	Fee added to Order based on Order Distance	Decimal	4,2	6.00	
Drivers_Driver_Id	ID of Driver assigned to Order	Varchar	5	12	Part of FK (ref: Drivers)
customerID	ID of Customer who ordered that Order	Varchar	5	12	Part of FK (ref: Customer)

Table: **Payments**

Column Name	Description	Data Type	Size	Format	Key?
Pmt_Id	Unique integer value assigned to individual payment	Varchar	5	12	PK
Pmt_Datetime	Date and time payment was made	Datetime	-	YYYY-MM-DD HH:MM:SS	
Pmt_Tip	Amount of tip added to payment	Decimal	4,2	3.21	
Pmt_Type	Form of Payment either card, venmo, or paypal	Varchar	15	Card	
Customer_Customer_Id	ID of Customer associated with this payment	Varchar	5	12	Part of FK (ref: Customer)

Promo_Promo_Id	ID of promo code associated with this payment	Varchar	5	12	Part of FK (ref: Promo)

Table: **Promo**

Column Name	Description	Data Type	Size	Format	Key?
Promo_Id	Unique ID Assigned to a Single Promotion	Varchar	5	1	PK
Promo_Name	Name of Promotion	Varchar	25	SAVE2	
Promo_Area	Whether the Promo affects a subtotal or delivery fee	Varchar	25	Subtotal	
Promo_Type	Whether the promotion is a percentage off or an amount off	Varchar	25	Amount	
Promo Value	The actual percentage or amount off	Decimal	4,2	2.00	

Table: **Restaurant**

Column Name	Description	Data Type	Size	Format	Key?
Rest_Id	Unique Int Assigned to Single Restaurants	Varchar	5	12	PK
Rest_Name	Name of Restaurant	Varchar	25	Taqueria Tsunami	
Rest_Type	Type of Cuisine Served at Restaurant	Varchar	25	Asian Fusion	
Rest_Ratings	Average Rating by Customers of this Restaurant from Yelp	Decimal	4,2	1.00	
Rest_Min_Price	Minimum Order Subtotal Required to Order from this Restaurant	Decimal	4,2	1.00	
Rest_Street	Restaurant Street Address	Varchar	25	1234 Bernie St	
Rest_City	City of Restaurant	Varchar	25	Athens	
Rest_State	State of Restaurant	Varchar	25	GA	
Rest_Zipcode	Zip-code of Restaurant	Varchar	10	12345	

Format, Queries, & Justifications

	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10
Multiple Table Join	X	X	X	X	X	X	X	X	X	X
Subquery					X		X			
Correlated Subquery			X							
GROUP BY	X			X		X	X	X		X
HAVING				X		X		X		
ORDER BY	X			X	X	X	X	X	X	
IN or NOT IN							X			
Built in function or a calculated field	X	X	X	X		X	X	X	X	X
REGEXP		X								
EXISTS or NOT EXISTS					X					

1.
This query calculates the total average prices of food in an ascending order from each restaurant. This is important because it helps us the cheapest average prices of food from restaurants which would be useful to customers.

```
SELECT Rest_Name AS 'Restaurant Name', AVG(Food_Price) AS 'Average Food Price'
FROM Food
JOIN Restaurant ON Food.Restaurant_Rest_Id = Restaurant.Rest_Id
GROUP BY Rest_Name
GROUP BY AVG(Food_Price);
```

2.
This query helps us find the customers who are a distance of 3-4 miles away from the order/driver. This is important because it helps identify the customers in a certain distance radius to help deliver the food faster to them by a particular driver.

```
SELECT Customer_Id, Customer_Fname, Customer_Lname,  Order_Distance
FROM Customer
JOIN Orders ON Customer.Customer_Id = Orders.customerID
WHERE Order_Distance regexp '^3.';
```

3.
This query helps us find the customers who have orders whose delivery fee cost is greater than the average delivery cost for that customer. This is important because it helps identify the customer that is willing to spend more on delivery fees than usual for orders.

```
SELECT Customer_Fname, Customer_Lname, Order_Id, Order_DeliveryFee
FROM Orders
JOIN Customer ON Customer.Customer_Id = Orders.customerID
WHERE Order_DeliveryFee > (SELECT AVG(Order_DeliveryFee)
                           FROM Orders
                           WHERE Customer.Customer_Id = Orders.customerID);
```

4.
This query shows us the customers who have ordered more than 1 order from the app. This is important because it helps target the more frequent users of the app to more promotions, deals, etc.

```
SELECT Customer_Lname, Customer_Fname, count(Order_Id) AS 'Number of Orders'
FROM Customer
JOIN Orders ON Orders.customerID = Customer.Customer_Id
GROUP BY Customer_Id
HAVING COUNT(order_Id) > 1
ORDER BY Customer_Lname;
```

5.
This query shows us the food names and the names of the restaurant that food is from that have never been ordered by a customer. This is important as it identifies which food items sell poorly and which restaurants struggle to sell their food items.

```
SELECT Food_Name, Rest_Name
FROM Food
JOIN Restaurant ON Rest_ID = Restaurant_Rest_ID
WHERE NOT EXISTS (SELECT * FROM OrderDetails
                                    WHERE Food_ID = Food_Food_ID);
```

6.
This query shows us the tip amounts that have been given by more than 1 customer and how many customers have given that tip amount. This is important as with this information we can graph, without outliers, what the drivers typically receive in tips and get a clearer picture than we would get from simply querying the average.

```
SELECT Pmt_Tip, COUNT(Customer_ID)
FROM Customer
JOIN Orders ON Orders.customerID = Customer.Customer_ID
JOIN Payments on Payments.Customer_Customer_ID = Customer.Customer_ID
GROUP BY Pmt_Tip
HAVING COUNT(Customer_ID) > 1;
```

7.
This query gives us the average distance of a customer's order where a customer ordered a quantity greater than 2 in that order. This is important because it will allow us to determine if there is a relationship between quantity of items ordered and the distance of a customer to a restaurant. If there is a positive relationship then marketing strategies or promotions can be given to customers further away if they order a greater quantity of food.

```
SELECT Customer_Id, Customer_Fname, Customer_Lname, AVG(Order_Distance)
FROM Customer
```

```
JOIN Orders ON Customer.customer_Id = Orders.customerID
WHERE Customer_ID IN (SELECT customerID
                      FROM Orders
                      JOIN OrderDetails ON Orders.Order_ID =
OrderDetails.Orders_Order_Id
            WHERE OrderDetails_Qty > 2)
GROUP BY Customer_Id
Order BY AVG(Order_Distance);
```

8.

This query shows us the customers that made more than one payment ordered by whoever made the most payments. This is important as it identifies customers that have returned after their first order and who orders the most afterwards too.

```
SELECT Customer_Id, Customer_Fname, Customer_Lname, COUNT(Pmt_Id)
FROM Customer
JOIN Payments ON Customer.Customer_Id = Payments.Customer_Customer_Id
GROUP BY Customer_Id
HAVING COUNT(Pmt_Id) > 1
ORDER BY COUNT(Pmt_Id) DESC;
```

9.

This query shows us the restaurants in the order that had the most orders from top to bottom. This is important as it identifies restaurants that are drawing in more business than others and generating revenue.

```
SELECT Rest_Name, COUNT(DISTINCT(Orders.Order_Id))
FROM Restaurant
JOIN Food ON Restaurant.Rest_Id = Food.Restaurant_Rest_Id
JOIN OrderDetails ON Food.Food_Id = OrderDetails.Food_Food_Id
JOIN Orders ON OrderDetails.Orders_Order_Id = Orders.Order_Id
GROUP BY Rest_Name
ORDER BY COUNT(DISTINCT(Orders.Order_Id)) DESC;
```

10.

This query shows us the foods names and the total revenue generated for each food. This is important as it identifies foods that are generating the most revenue, and it allows for restaurants to focus more on why that is.

```
SELECT Food_Name, SUM(OrderDetails_Qty), Food_Price*SUM(OrderDetails_Qty) AS
'Sales'
FROM Food
JOIN OrderDetails ON Food.Food_Id = OrderDetails.Food_Food_Id
GROUP BY Food.Food_Id;
```