DoorDash Food Delivery Service

CS-6360:001: Database Design

Team Number: 31

Team Members:

1. Manan Dalal (MUD200000)
2. Vishesh Mehta (VJM190001)
3. Fenil Godhani (FKG210000)

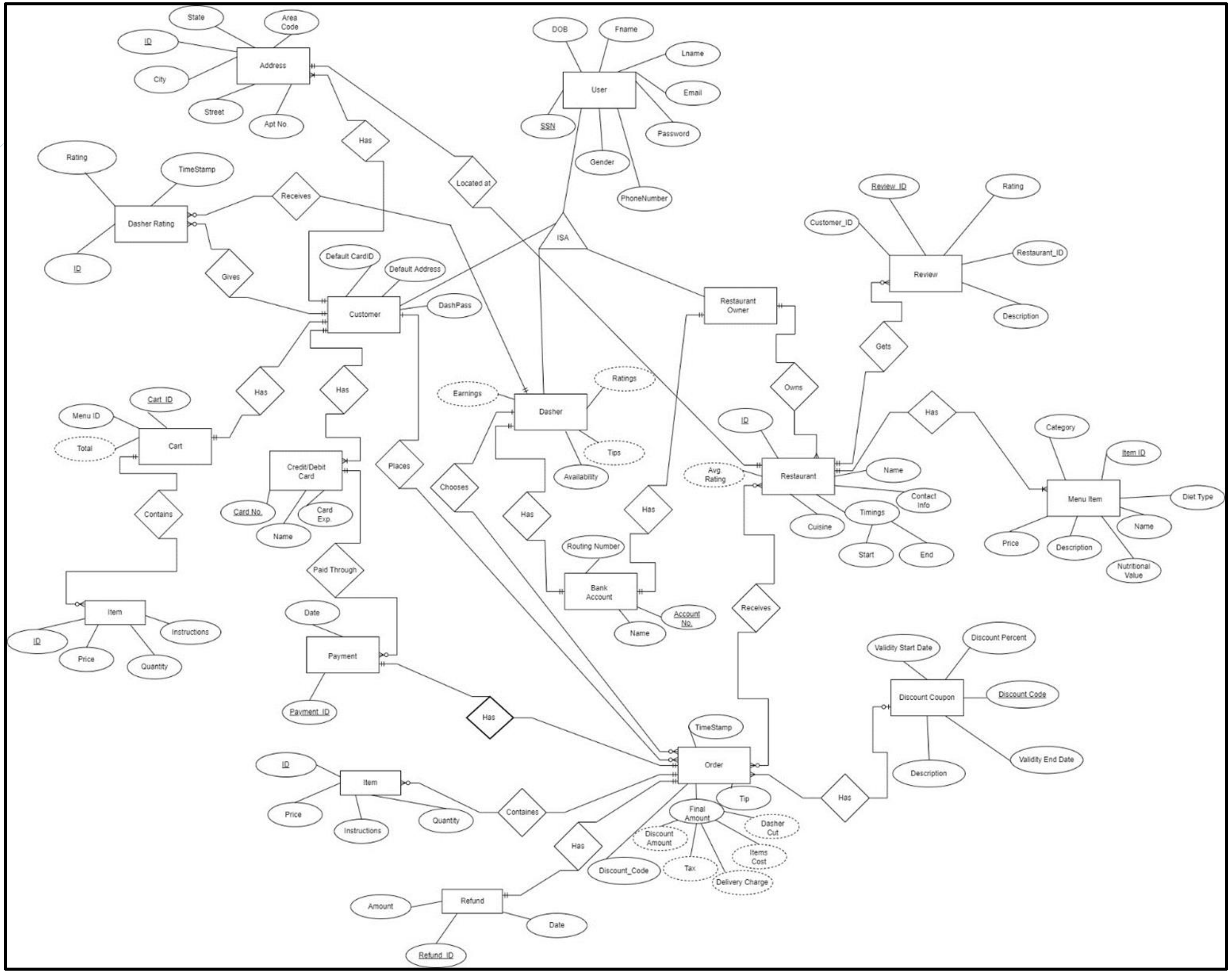
Data/Functional Requirements

- For Customers:
  - Create/Update Login and Contact Information
  - They can search for restaurants based on name, cuisine, menu items etc...
  - They can add items to their cart, place orders, specific requests within the order, updates and track the status of their order.
  - Payment, their order histories.
  - Can provide feedback/reviews for the order, restaurant or/and the dasher.
  - Can become a member and enjoy other benefits by buying DashPass.

Cont...

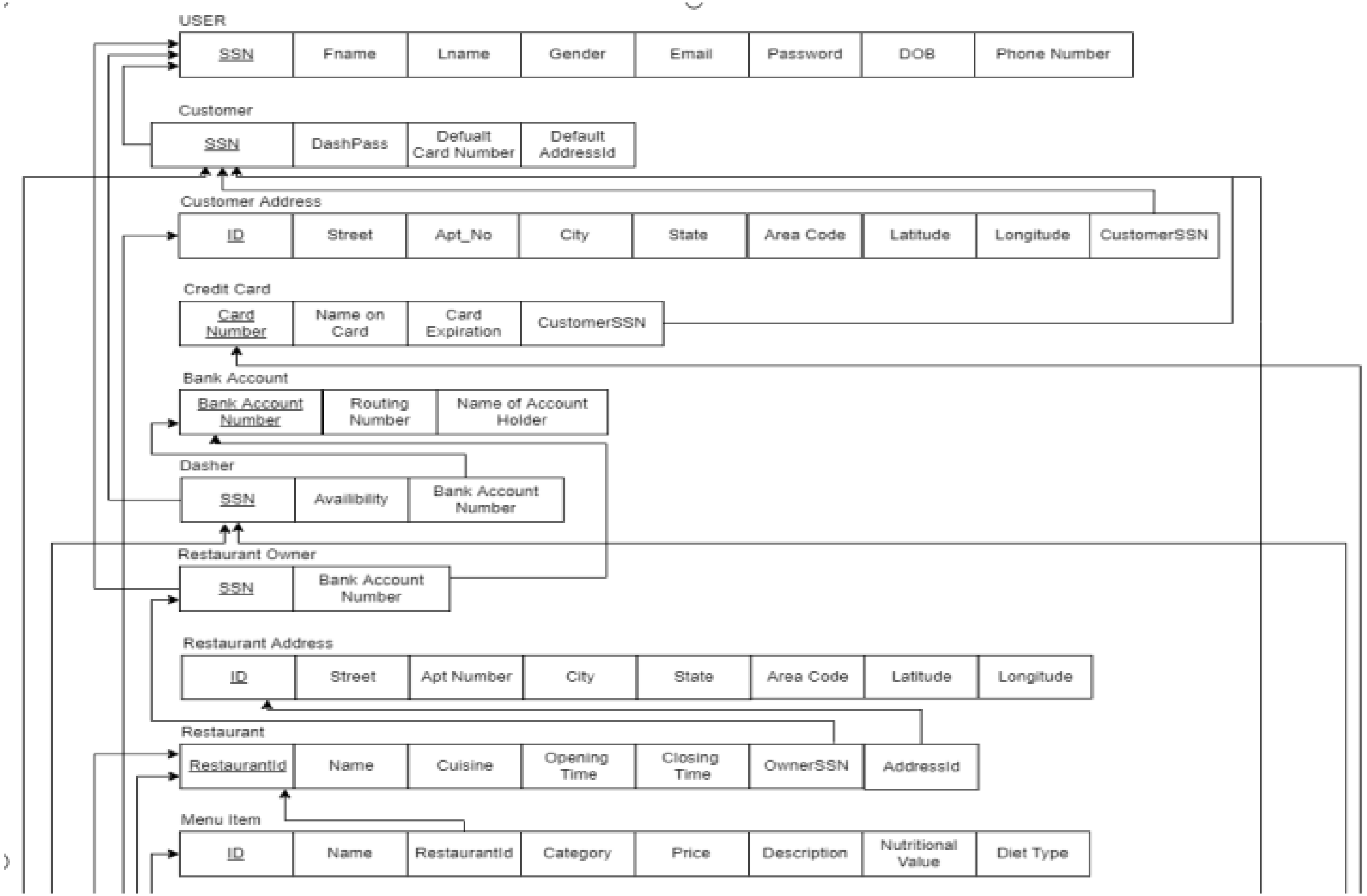
- For Restaurants:
  - Create a profile and can add new items/edit existing items to the menu.
  - They can receive orders, update status of the order and get information of the dasher.
  - Receive payments from the customer via DoorDash.
  - Can view insightful data such as most/least items ordered, best/least rated items, user feedbacks etc.…
  - Can add bank information to receive payments.
- For Dashers:
  - Choose from available orders to pick up around their area/current location.
  - Know when the order is available for pickup.
  - De-Register incase they want to discontinue.
  - Can receive payments/fees from DoorDash and tips from customers.
  - Can add bank/wallet information to receive payments.

ER/EER Diagram



Final Relational Model

After applying normalization, the below diagram is the normalized relational schema of our system.



USER

SSN	Fname	Lname	Gender	Email	Password	DOB	Phone Number

Customer

SSN	DashPass	Defualt Card Number	Default AddressId

Customer Address

ID	Street	Apt_No	City	State	Area Code	Latitude	Longitude	CustomerSSN

Credit Card

Card Number	Name on Card	Card Expiration	CustomerSSN

Bank Account

Bank Account Number	Routing Number	Name of Account Holder

Dasher

SSN	Availibility	Bank Account Number

Restaurant Owner

SSN	Bank Account Number

Restaurant Address

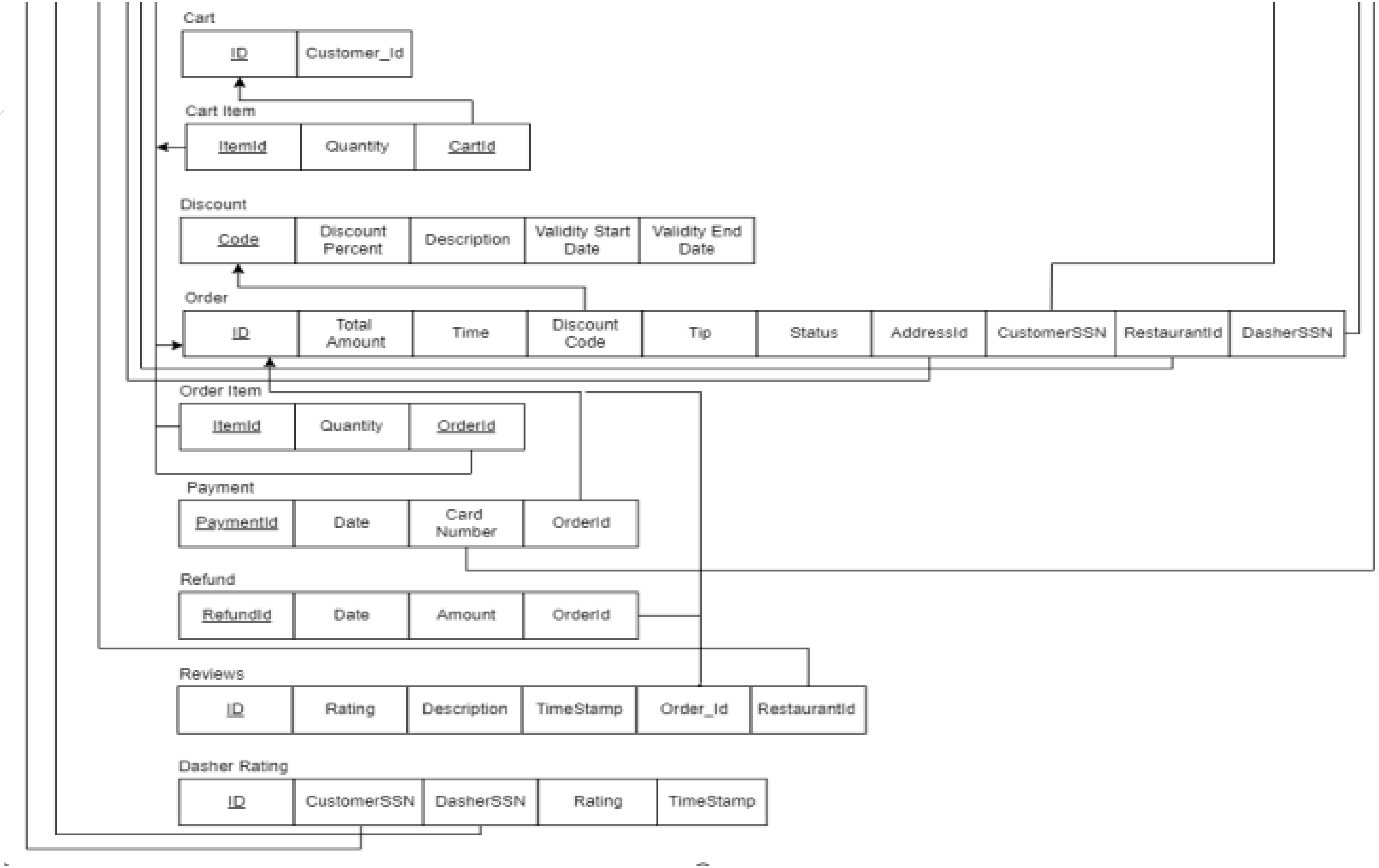
ID	Street	Apt Number	City	State	Area Code	Latitude	Longitude

Restaurant

RestaurantId	Name	Cuisine	Opening Time	Closing Time	OwnerSSN	AddressId

Menu Item

ID	Name	RestaurantId	Category	Price	Description	Nutritional Value	Diet Type



Cart

ID	Customer_Id

Cart Item

ItemId	Quantity	CartId

Discount

Code	Discount Percent	Description	Validity Start Date	Validity End Date

Order

ID	Total Amount	Time	Discount Code	Tip	Status	AddressId	CustomerSSN	RestaurantId	DasherSSN

Order Item

ItemId	Quantity	OrderId

Payment

PaymentId	Date	Card Number	OrderId

Refund

RefundId	Date	Amount	OrderId

Reviews

ID	Rating	Description	TimeStamp	Order_Id	RestaurantId

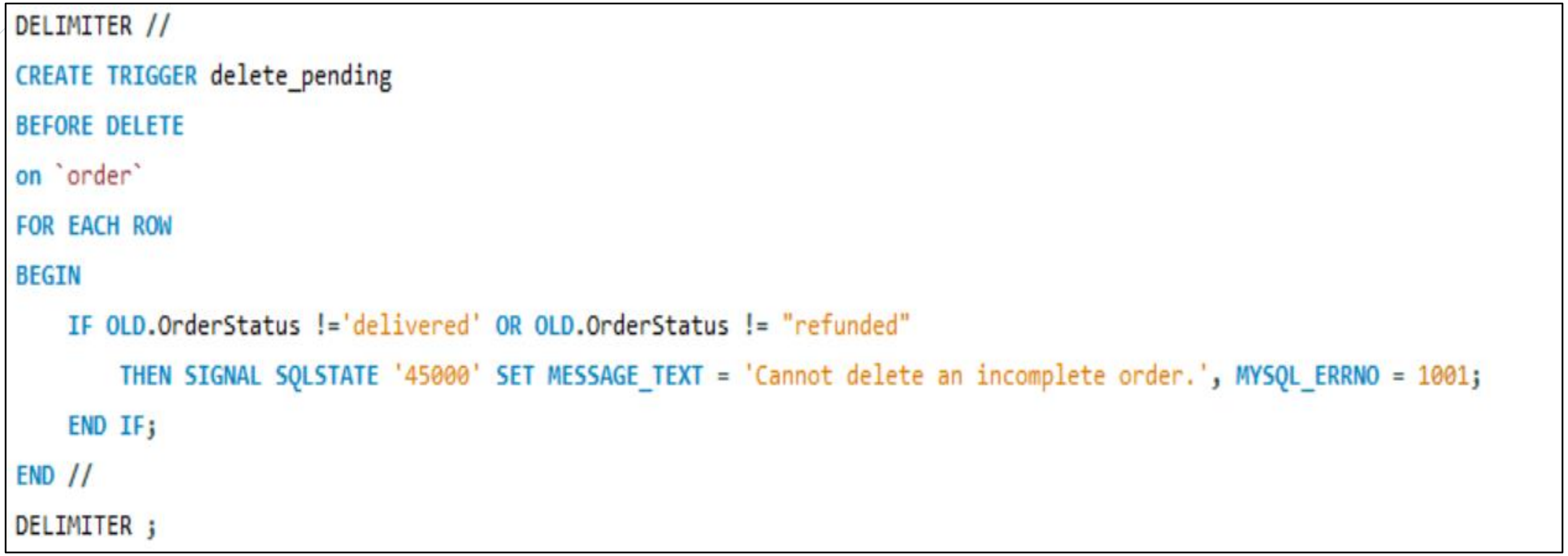
Dasher Rating

ID	CustomerSSN	DasherSSN	Rating	TimeStamp

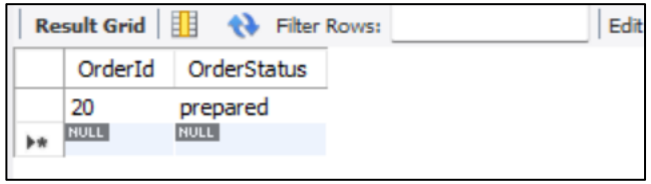
PL/SQL Statements

Triggers

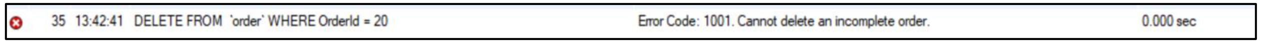
1.Creating a trigger that can satisfy the condition that “An order cannot be deleted if it has not been delivered or refunded”



```
DELIMITER //
CREATE TRIGGER delete_pending
BEFORE DELETE
on `order`
FOR EACH ROW
BEGIN
    IF OLD.OrderStatus !='delivered' OR OLD.OrderStatus != "refunded"
        THEN SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Cannot delete an incomplete order.', MYSQL_ERRNO = 1001;
    END IF;
END //
DELIMITER ;
```

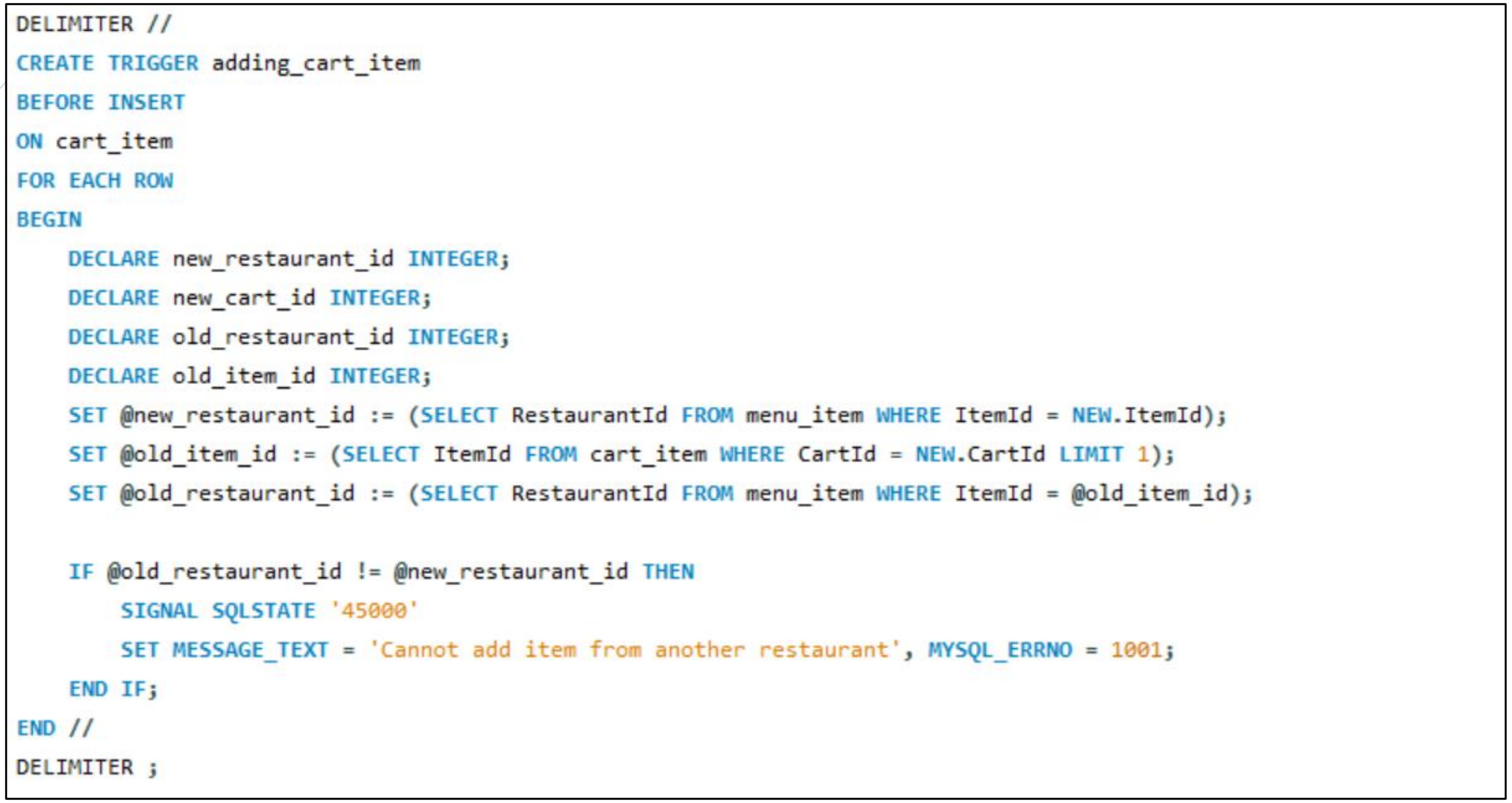


OrderId	OrderStatus
20	prepared
NULL	NULL

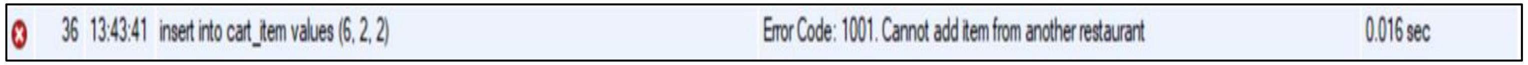


```
35  13:42:41  DELETE FROM `order` WHERE OrderId = 20        Error Code: 1001. Cannot delete an incomplete order.        0.000 sec
```

2.Creating a trigger that can satisfy the condition that “A customer can only have items from a single restaurant in his cart at a single time. If he adds an item from another restaurant, then the cart should be emptied first.”



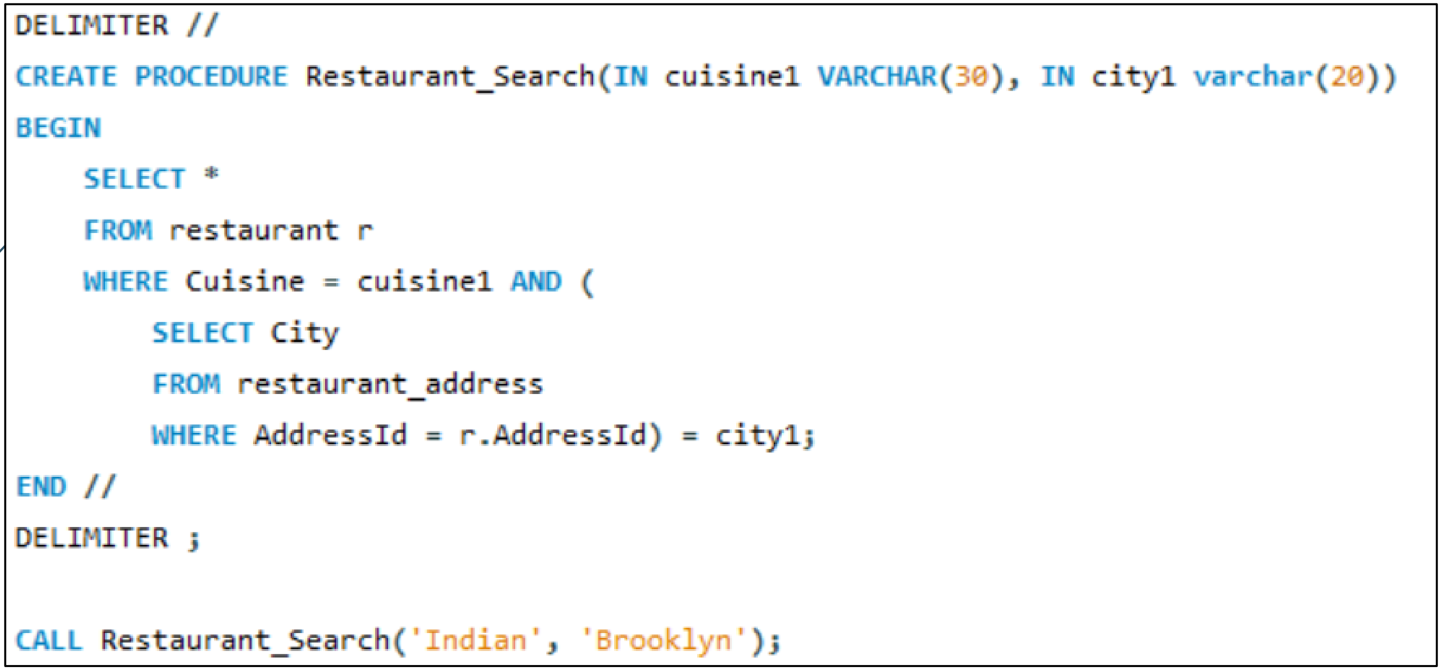
```
DELIMITER //
CREATE TRIGGER adding_cart_item
BEFORE INSERT
ON cart_item
FOR EACH ROW
BEGIN
    DECLARE new_restaurant_id INTEGER;
    DECLARE new_cart_id INTEGER;
    DECLARE old_restaurant_id INTEGER;
    DECLARE old_item_id INTEGER;
    SET @new_restaurant_id := (SELECT RestaurantId FROM menu_item WHERE ItemId = NEW.ItemId);
    SET @old_item_id := (SELECT ItemId FROM cart_item WHERE CartId = NEW.CartId LIMIT 1);
    SET @old_restaurant_id := (SELECT RestaurantId FROM menu_item WHERE ItemId = @old_item_id);

    IF @old_restaurant_id != @new_restaurant_id THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Cannot add item from another restaurant', MYSQL_ERRNO = 1001;
    END IF;
END //
DELIMITER ;
```



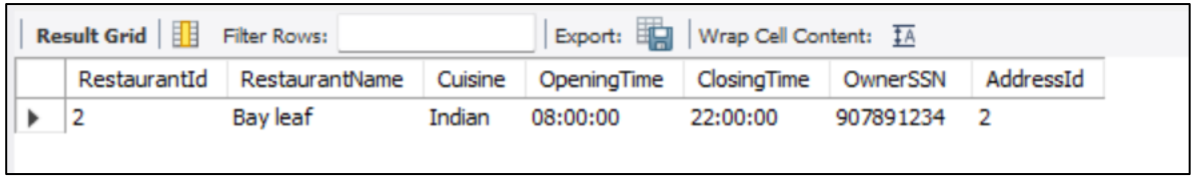
```
36  13:43:41  insert into cart_item values (6, 2, 2)    Error Code: 1001. Cannot add item from another restaurant    0.016 sec
```

Stored Procedures

1. A Customer can search for restaurants using filters like cuisine, city etc..

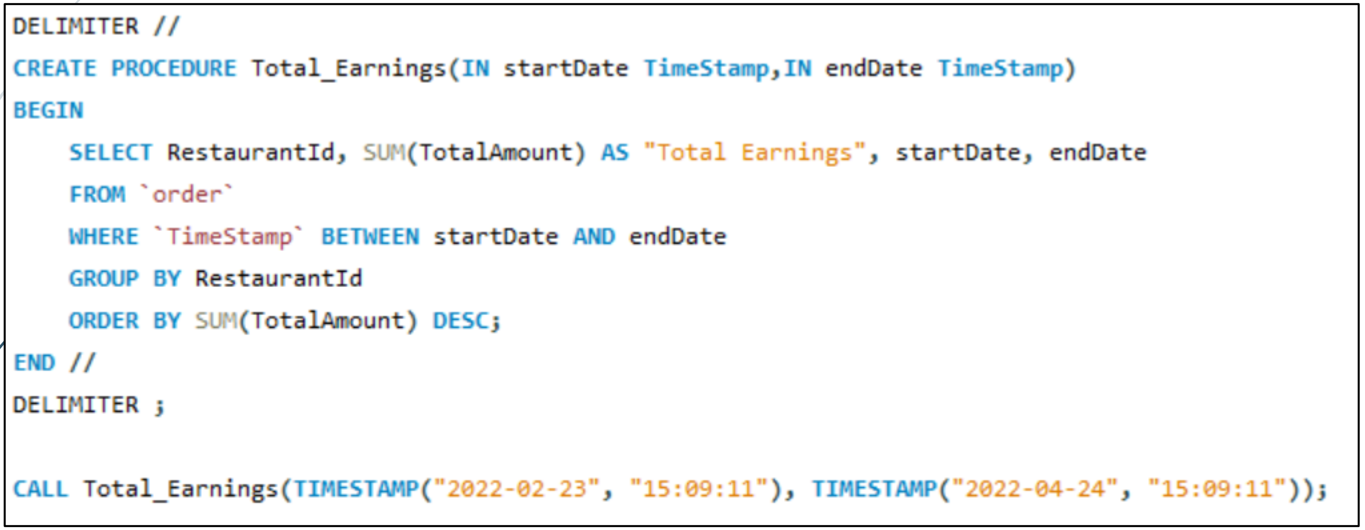


```
DELIMITER //
CREATE PROCEDURE Restaurant_Search(IN cuisine1 VARCHAR(30), IN city1 varchar(20))
BEGIN
    SELECT *
    FROM restaurant r
    WHERE Cuisine = cuisine1 AND (
        SELECT City
        FROM restaurant_address
        WHERE AddressId = r.AddressId) = city1;
END //
DELIMITER ;

CALL Restaurant_Search('Indian', 'Brooklyn');
```

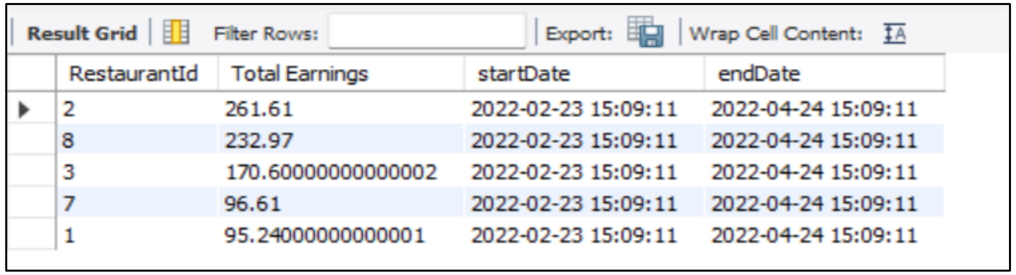


RestaurantId	RestaurantName	Cuisine	OpeningTime	ClosingTime	OwnerSSN	AddressId
2	Bay leaf	Indian	08:00:00	22:00:00	907891234	2

2. DoorDash can find out the earnings of each restaurant between a particular period.

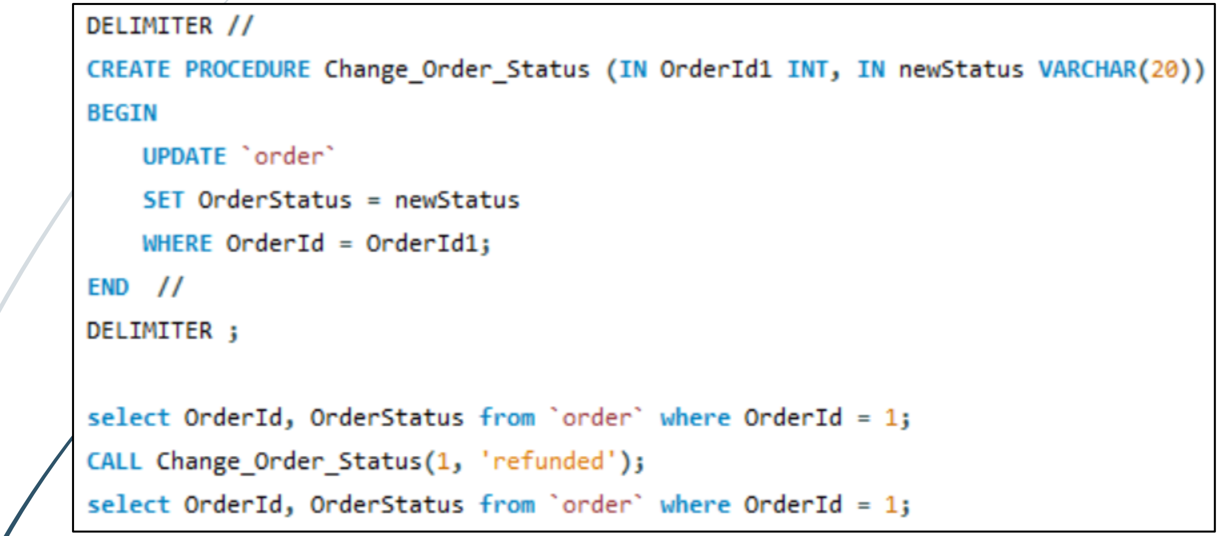


```
DELIMITER //
CREATE PROCEDURE Total_Earnings(IN startDate TimeStamp,IN endDate TimeStamp)
BEGIN
    SELECT RestaurantId, SUM(TotalAmount) AS "Total Earnings", startDate, endDate
    FROM `order`
    WHERE `TimeStamp` BETWEEN startDate AND endDate
    GROUP BY RestaurantId
    ORDER BY SUM(TotalAmount) DESC;
END //
DELIMITER ;

CALL Total_Earnings(TIMESTAMP("2022-02-23", "15:09:11"), TIMESTAMP("2022-04-24", "15:09:11"));
```

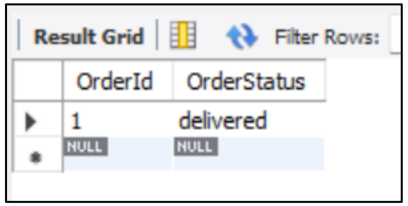


RestaurantId	Total Earnings	startDate	endDate
2	261.61	2022-02-23 15:09:11	2022-04-24 15:09:11
8	232.97	2022-02-23 15:09:11	2022-04-24 15:09:11
3	170.60000000000002	2022-02-23 15:09:11	2022-04-24 15:09:11
7	96.61	2022-02-23 15:09:11	2022-04-24 15:09:11
1	95.24000000000001	2022-02-23 15:09:11	2022-04-24 15:09:11

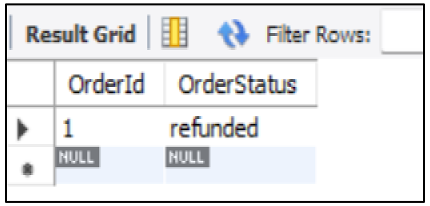
3. A Restaurant can change the status of an order as it gets processed and completed.



```
DELIMITER //
CREATE PROCEDURE Change_Order_Status (IN OrderId1 INT, IN newStatus VARCHAR(20))
BEGIN
    UPDATE `order`
    SET OrderStatus = newStatus
    WHERE OrderId = OrderId1;
END  //
DELIMITER ;

select OrderId, OrderStatus from `order` where OrderId = 1;
CALL Change_Order_Status(1, 'refunded');
select OrderId, OrderStatus from `order` where OrderId = 1;
```



OrderId	OrderStatus
1	delivered
NULL	NULL



OrderId	OrderStatus
1	refunded
NULL	NULL

Thank You.....