CS6360.001 DATABASE DESIGN
PROJECT TITLE: UberEats

Bhavya Thadiboina
NetID: BXT220009

PROJECT DESCRIPTION:

Uber Eats is an online food ordering and delivery platform. The guest could choose to order meals that are delivered by couriers using cars, scooters, bikes, or on foot according to their preferences and requirements listed by the hosts of the uber eats system. The guests and the hosts could interact via the uber eats in-app messaging system. The overall process is given a review or a rating for an improved experience. Due to its versatile availability of properties, Uber Eats is a very popular platform to book a stay or an experience.

Entities:

1. User
2. Delivery Address
3. All Categories
4. Order Item
5. Order
6. Payment
7. Delivery
8. Delivery Driver
9. Reviews
10. Promo Code

Data Requirements and Assumptions:

User: The system should store information about users, including their name, email address, phone number, password, and delivery address.

● One User can have one Delivery Address.

● One User can place many Orders.

● One User can write many Reviews.

All Categories: The system should store information about restaurants, including their name, location, cuisine type, menu items, and prices.

● All Categories can have many Menu Items.

● All Categories can receive many Orders.

Order: The system should store information about orders, including the date and time of the order, the restaurant and the user who placed the order, the items ordered, and the totalcost of the order.

● One Order can be placed by one User.

● One Order can be associated with one catagories.

● One Order can contain many Items.

● One Order can have one Payment.

● One Order can have one Delivery.

Delivery: The system should store information about deliveries, including the date and time of delivery, the delivery address, and the delivery status.

● One Delivery can be associated with one Order.

● One Delivery can be made by one Delivery Partner.

Payment: The system should store information about payments, including the payment method, the amount paid, payment Time and the payment status.

● One Payment can be associated with one Order.

Review: The system should allow users to rate and review restaurants should store this information in the database.

● One Review can be written by one User.

● One Review can be associated with one Restaurant

Delivery Driver: The system should store information about delivery partners, including their name, Vehicle Number, contact details, and availability.

● One Delivery Partner can make many Deliveries.

Promo Code: The system should store information about Promo Code including promo codes, discounts, Name and expiration dates.

● One Promotion can be used by many Users.

● One Promotion can be associated with many Orders or Users.

Delivery Address: This entity represents the delivery address associated with a user. Each delivery address has a unique ID and can be associated with only one user.

Order Item: This entity represents an item on a restaurant's menu. Each menu item has a unique ID and can be part of many orders.

Assumptions:

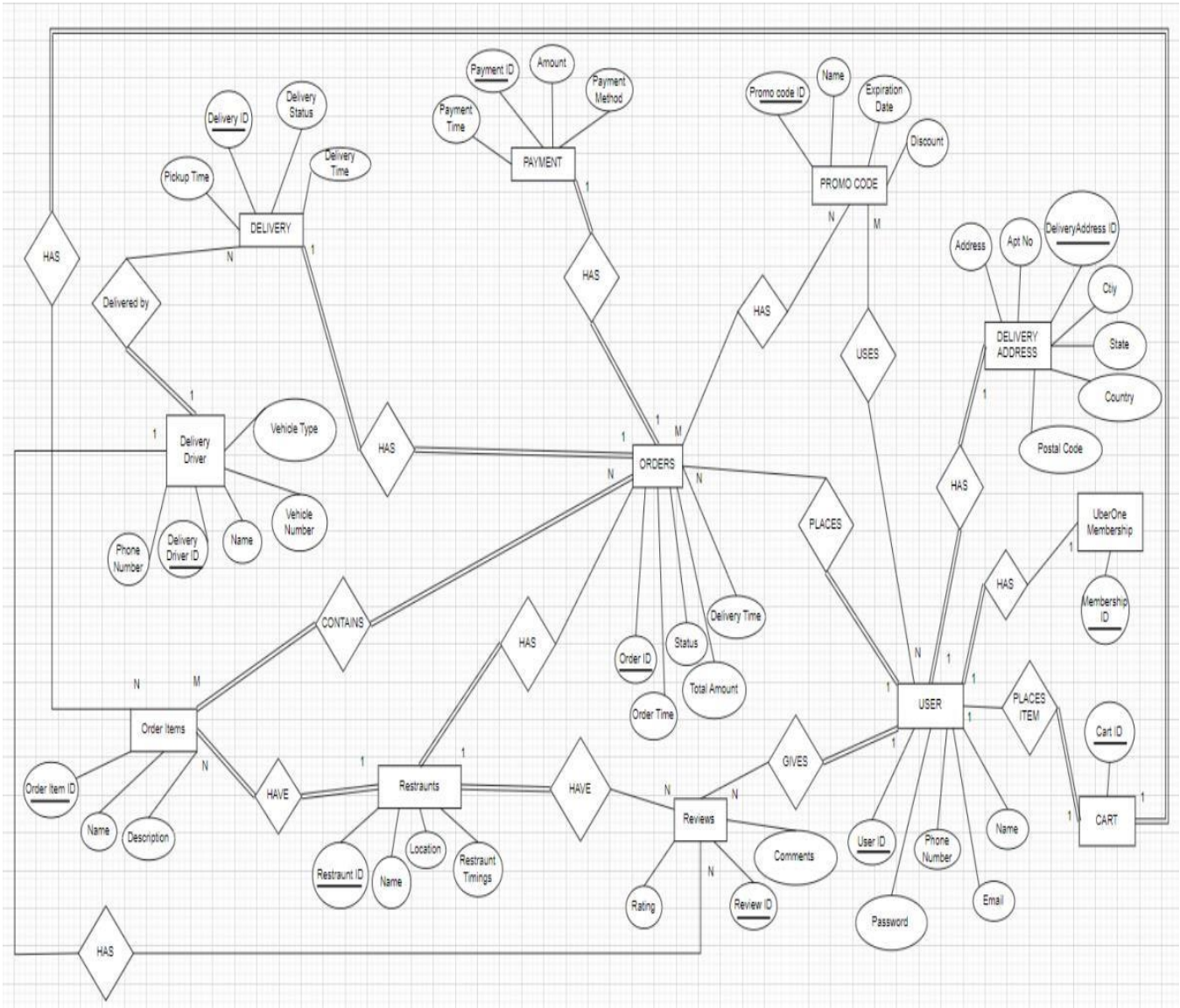
User: This entity represents the user who places the order. Each user has a unique ID and can place many orders and write many reviews. Each user can have one delivery address.

Delivery Address: This entity represents the delivery address associated with a user. Each delivery address has a unique ID and can be associated with only one user.

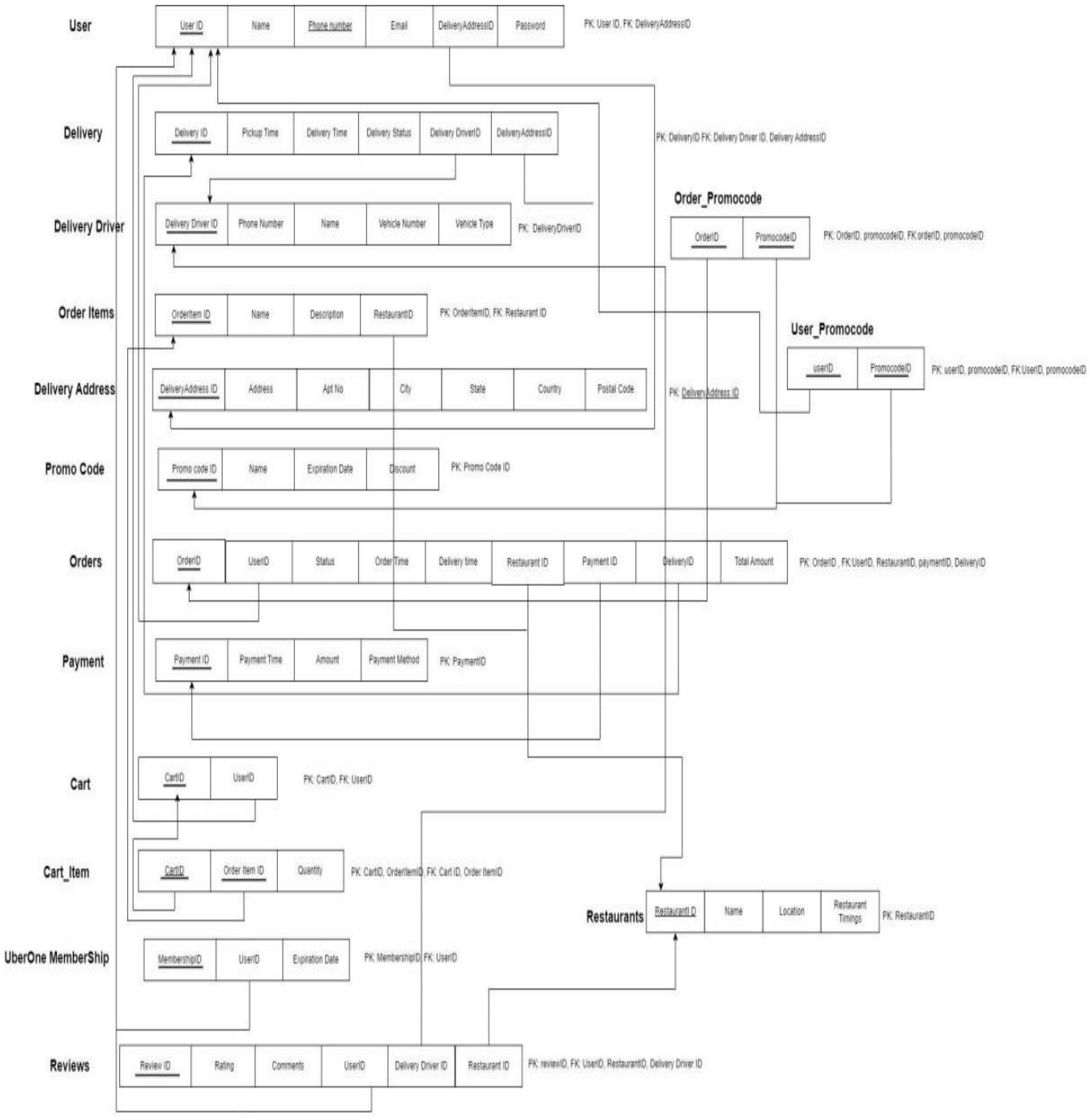
Order Item: This entity represents an item on the menu of a restaurant. Each menu item has a unique ID and can be part of many orders.

Order: This entity represents an order placed by a user. Each order has a unique ID and is associated with one user and one restaurant. Each order can have many items.

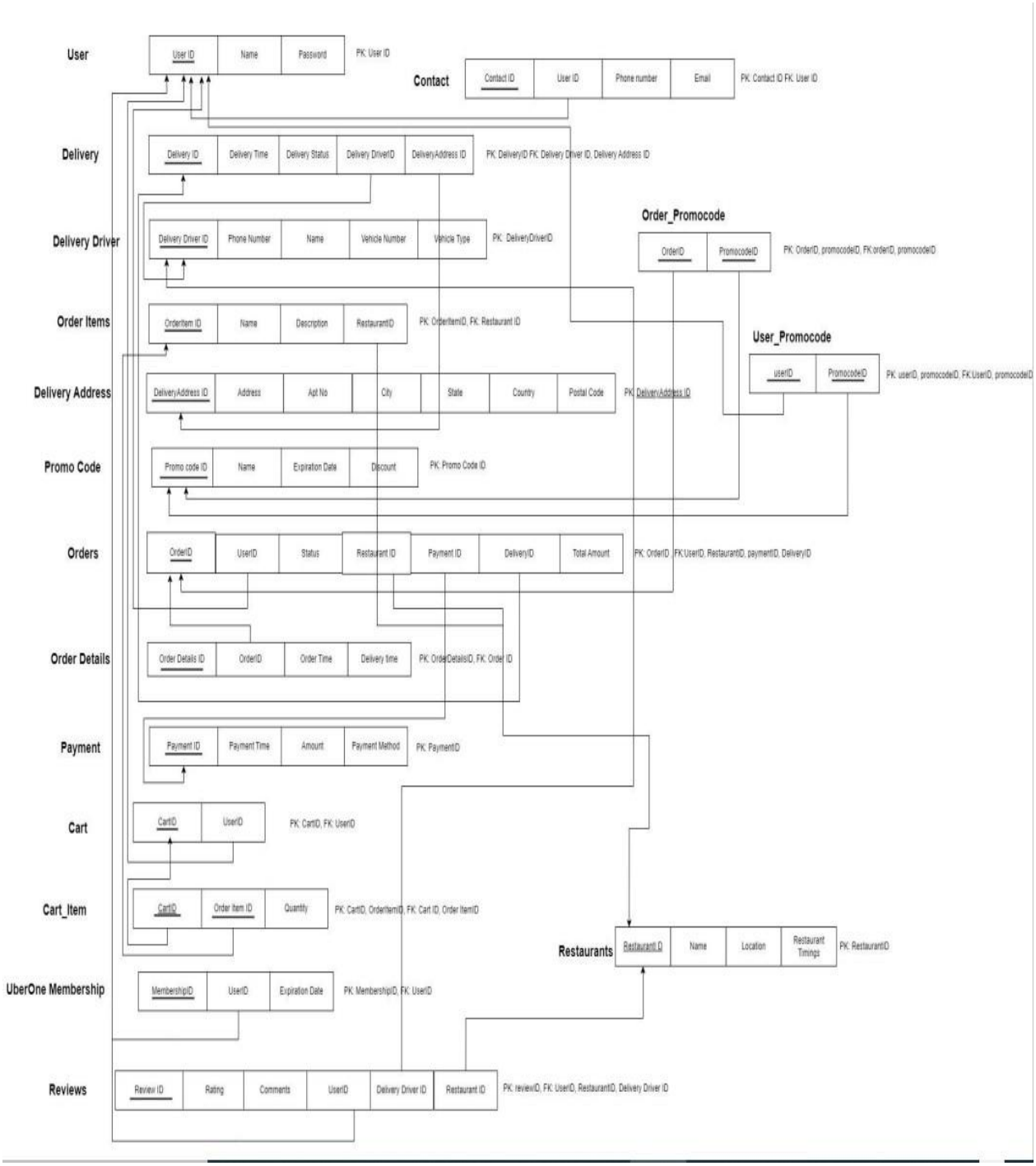
ER DIAGRAM:


DELIVERY
Delivery ID
Delivery Status
Pickup Time
Delivery Time
HAS
Delivered by
Delivery Driver
Vehicle Type
Vehicle Number
Name
Delivery Driver ID
Phone Number
PAYMENT
Payment ID
Amount
Payment Method
Payment Time
ORDERS
Order ID
Status
Delivery Time
Total Amount
Order Time
CONTAINS
Order Items
Order Item ID
Name
Description
HAVE
Restraunts
Restraunt ID
Name
Location
Restraunt Timings
Reviews
Rating
Comments
Review ID
GIVES
PROMO CODE
Promo code ID
Name
Expiration Date
Discount
USES
PLACES
USER
User ID
Phone Number
Name
Email
Password
DELIVERY ADDRESS
DeliveryAddress ID
Address
Apt No
City
State
Country
Postal Code
UberOne Membership
Membership ID
PLACES ITEM
CART
Cart ID

Relational Schema



Final relational schema after normalization.



SQL COMMANDS TO CREATE TABLES:

```
create schema Uber_Eats;
Use Uber_Eats;
Create table user(user_ID Varchar(60) not null,
 name Varchar(60) not null,
 password Varchar(60) not null,
 Primary key (user_ID)
 );

Create Table delivery_Driver(deliverydriver_ID varchar(60) not null,
Phone_number varchar(15) not null,
name varchar(30) not null,
Vehicle_number varchar(20) not null,
Vehicle_type varchar(30) not null,
primary key (deliverydriver_ID)
 );
Create table promo_code(promocode_ID int not null,
 name varchar(50) not null,
 expiration date not null,
 discount decimal(10,2) not null default 0,
 primary key (promocode_ID)
 );

Create table payment(payment_ID int not null,
Payment_time datetime not null,
Amount decimal(9,2) not null,
```

```
payment_method varchar(60)not null,
Primary key (payment_ID));

Create table restaurants(restaurant_ID int not null,
name varchar(60) not null,
Location varchar(40) not null,
Restaurant_timings varchar(50) not null,
Primary key (restaurant_ID)
);

Create table deliveryaddress(deliveryaddr_ID int not null,
Address varchar(50) not null,
AptNo varchar(20),
City varchar(50)not null,
state varchar(50)not null,
country char(40) not null,
postal_code int not null,
primary key (deliveryaddr_ID)
);

Create table contact(contact_ID varchar(60) not null,
phone_num varchar(15) not null,
Email varchar(100),
user_ID varchar(60) not null,
primary key (contact_ID));

Create table orderitems(orderitem_ID int not null,
```

```
Name varchar(40) not null ,
restaurant_ID int not null,
Description varchar(150) not null ,
Primary key (orderitem_ID),
Foreign key (restaurant_ID) REFERENCES restaurants(restaurant_ID) ON DELETE
CASCADE);
Create Table delivery(delivery_ID int not null,
Delivery_time datetime not null,
Delivery_Status Varchar(60) not null,
deliverydriver_ID varchar(60) not null,
deliveryaddr_ID int not null,
primary key (delivery_ID),
Foreign key (deliverydriver_ID) REFERENCES delivery_driver(deliverydriver_ID) ON
DELETE CASCADE,
Foreign key (deliveryaddr_ID) REFERENCES deliveryaddress(deliveryaddr_ID) ON DELETE
CASCADE
);

Create table orders(order_ID int not null,
Status varchar(50) not null,
restaurant_ID int not null,
user_ID varchar(60) not null,
payment_ID int not null,
delivery_ID int not null,
total_amount Decimal(10,2) not null,
primary key(order_ID),
FOREIGN KEY (user_ID) REFERENCES user(user_ID) ON DELETE CASCADE,
```

```
FOREIGN KEY (restaurant_ID) REFERENCES Restaurants(restaurant_ID) ON DELETE
CASCADE,
FOREIGN KEY (payment_ID) REFERENCES Payment(payment_ID) ON DELETE CASCADE,
FOREIGN KEY (delivery_ID) REFERENCES Delivery(delivery_ID) ON DELETE CASCADE
);


Create table orderdetails(orderdetails_ID int not null,
order_ID int not null,
ordertime datetime not null,
deliverytime datetime not null,
primary key (orderdetails_ID),
Foreign key (order_ID) REFERENCES orders(order_ID) ON DELETE CASCADE
);

Create table reviews(review_ID int not null,
Rating int not null,
comments varchar(500) not null,
user_ID varchar(60) not null,
restaurant_ID int not null,
deliverydriver_ID varchar(60) not null,
Primary key (review_ID),
Foreign key (deliverydriver_ID) REFERENCES delivery_driver(deliverydriver_ID) ON
DELETE CASCADE,
Foreign key (user_ID) REFERENCES user(user_ID) ON DELETE CASCADE,
Foreign key (restaurant_ID) REFERENCES restaurants(restaurant_ID) ON DELETE
CASCADE
```

```
);

Create table order_promocode(order_ID int not null,
promocode_ID int not null,
 primary key(order_ID , promocode_ID),
Foreign key (order_ID) REFERENCES orders(order_ID) ON DELETE CASCADE,
Foreign key (promocode_ID) REFERENCES promo_code(promocode_ID) ON DELETE
CASCADE
 );
Create table user_promocode(User_ID varchar(60) not null,
promocode_ID int not null,
primary key(user_ID, promocode_ID),
Foreign key (user_ID) REFERENCES user(user_ID) ON DELETE CASCADE,
Foreign key (promocode_ID) REFERENCES promo_code(promocode_ID) ON DELETE
CASCADE
 );
Create table cart(cart_ID int not null,
user_ID varchar(60) not null,
Primary key(Cart_ID),
Foreign key (user_ID) REFERENCES user(user_ID) ON DELETE CASCADE);

Create table cartitem(cart_ID int not null,
OrderItem_ID int not null,
quantity int not null,
 Primary key(cart_ID, OrderItem_ID),
Foreign key (cart_ID) REFERENCES cart(cart_ID) ON DELETE CASCADE,
Foreign key (orderitem_ID) REFERENCES orderitems(orderitem_ID) ON DELETE CASCADE
```

```
);

Create table uberone_membership(membership_ID Varchar(20) not null,
Expirationdate date not null,
user_ID varchar(60) not null,
Primary key(membership_ID),
Foreign key (user_ID) REFERENCES user(user_ID) ON DELETE CASCADE
);
```

Code for inserting the data into tables:

```
Use Uber_Eats;
INSERT INTO user (user_ID, name, password)
VALUES ('user1', 'John Doe', 'password1'),('user2', 'Jane Smith', 'password2'),('user3', 'Michael Johnson', 'password3');
Select * from user;

INSERT INTO delivery_driver (deliverydriver_ID, Phone_number, name, Vehicle_number, Vehicle_type)
VALUES ('driver1', '1234567890', 'John Driver', 'ABC123', 'Car'),('driver2', '9876543210', 'Jane Driver', 'XYZ789', 'Bike'), ('driver3', '4567891230', 'Michael Driver', 'DEF456', 'Car');
Select * from delivery_driver;

INSERT INTO promo_code (promocode_ID, name, expiration, discount)
VALUES (1, 'SUMMER10', '2023-08-31', 0.10),(2, 'WELCOME15', '2023-12-31', 0.15),(3, 'SAVEMORE20', '2023-10-15', 0.20);
Select * from promo_code;

INSERT INTO payment (payment_ID, Payment_time, Amount, payment_method)
VALUES (1, '2023-05-07 10:15:00', 25.99, 'Credit Card'),(2, '2023-05-06 15:30:00', 15.50, 'PayPal'),(3, '2023-05-05 12:45:00', 10.75, 'Cash');
Select * from payment;

INSERT INTO deliveryaddress (deliveryaddr_ID, Address, AptNo, City, state, country, postal_code)
VALUES (1, '123 First St', 'Apt 1', 'City A', 'State A', 'Country A', 12345),(2, '456 Second St', NULL, 'City B', 'State B', 'Country B', 67890),(3, '789 Third St', 'Apt 3', 'City C', 'State C', 'Country C',
```

```
54321);
Select * from deliveryaddress;

 INSERT INTO contact (contact_ID, phone_num, Email, user_ID)
VALUES ('contact1', '1234567890', 'john@example.com', 'user1'),('contact2', '9876543210',
'jane@example.com', 'user2'),('contact3', '4567891230', 'michael@example.com', 'user3');
Select * from contact;

 INSERT INTO orderitems (orderitem_ID, Name, restaurant_ID, Description)
VALUES (1, 'Item A', 1, 'Description for Item A'),(2, 'Item B', 2, 'Description for Item B'),(3, 'Item
C', 3, 'Description for Item C');
Select * from orderitems;

 INSERT INTO delivery (delivery_ID, Delivery_time, Delivery_Status, deliverydriver_ID,
deliveryaddr_ID)
VALUES (1, '2023-05-07 12:00:00', 'Delivered', 'driver1', 1),(2, '2023-05-08 15:30:00', 'In Transit',
'driver2', 2),(3, '2023-05-09 18:45:00', 'Pending', 'driver3', 3);
Select * from delivery;

 INSERT INTO orders (order_ID, Status, restaurant_ID, user_ID, payment_ID, delivery_ID,
total_amount)
VALUES (1, 'Pending', 1, 'user1', 1, 1, 20.50),(2, 'Delivered', 2, 'user2', 2, 2, 35.75),(3, 'Cancelled',
3, 'user3', 3, 3, 15.25);
Select * from orders;

 INSERT INTO orderdetails (orderdetails_ID, order_ID, ordertime, deliverytime)
VALUES (1, 1, '2023-05-07 11:45:00', '2023-05-07 12:30:00'),(2, 2, '2023-05-08 15:00:00', '2023-
05-08 16:00:00'),(3, 3, '2023-05-09 18:30:00', '2023-05-09 19:15:00');
Select * from orderdetails;

 INSERT INTO reviews (review_ID, Rating, comments, user_ID, restaurant_ID, deliverydriver_ID)
VALUES (1, 4, 'Great experience!', 'user1', 1, 'driver1'),(2, 3, 'Average service.', 'user2', 2,
'driver2'),(3, 5, 'Excellent food and delivery!', 'user3', 3, 'driver3');
Select * from reviews;

 INSERT INTO order_promocode (order_ID, promocode_ID)
VALUES (1, 1),(2, 2),(3, 3);
Select * from order_promocode;

 INSERT INTO user_promocode (user_ID, promocode_ID)
VALUES ('user1', 1),('user2', 2),('user3', 3);
Select * from user_promocode;
```

```
INSERT INTO cart (cart_ID, user_ID)
VALUES (1, 'user1'),(2, 'user2'),(3, 'user3');
Select * from cart;

INSERT INTO cartitem (cart_ID, OrderItem_ID, quantity)
VALUES (1, 1, 2),(2, 2, 1),(3, 3, 3);
Select * from cartitem;

INSERT INTO uberone_membership (membership_ID, Expirationdate, user_ID)
VALUES ('membership1', '2023-12-31', 'user1'),('membership2', '2023-11-30',
'user2'),('membership3', '2023-10-31', 'user3');
Select * from uberone_membership;
```

STORED PROCEDURES:

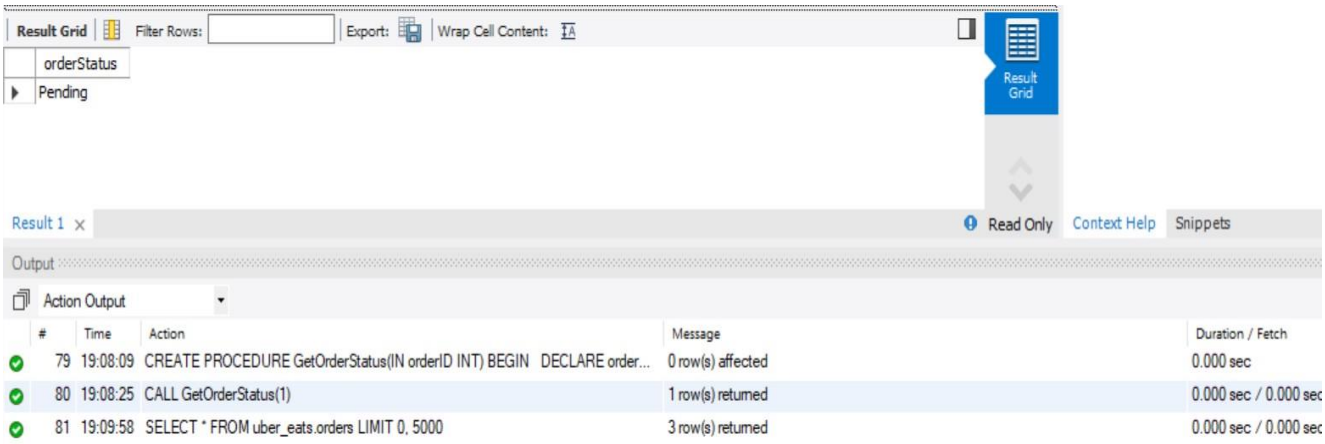
1. Stored procedure to GetOrderStatus

The first procedure we created is this GetOrderStatus

```
Use Uber_Eats;

DELIMITER //
CREATE PROCEDURE GetOrderStatus(IN orderID INT)
BEGIN
  DECLARE orderStatus VARCHAR(50);

  SELECT Status INTO orderStatus
  FROM orders
  WHERE order_ID = orderID;

  SELECT orderStatus;
END //
DELIMITER ;

CALL GetOrderStatus(1);
```

Output for the Stored procedure to GetOrderStatus:

We can see that the stored procedure has been executed successfully.



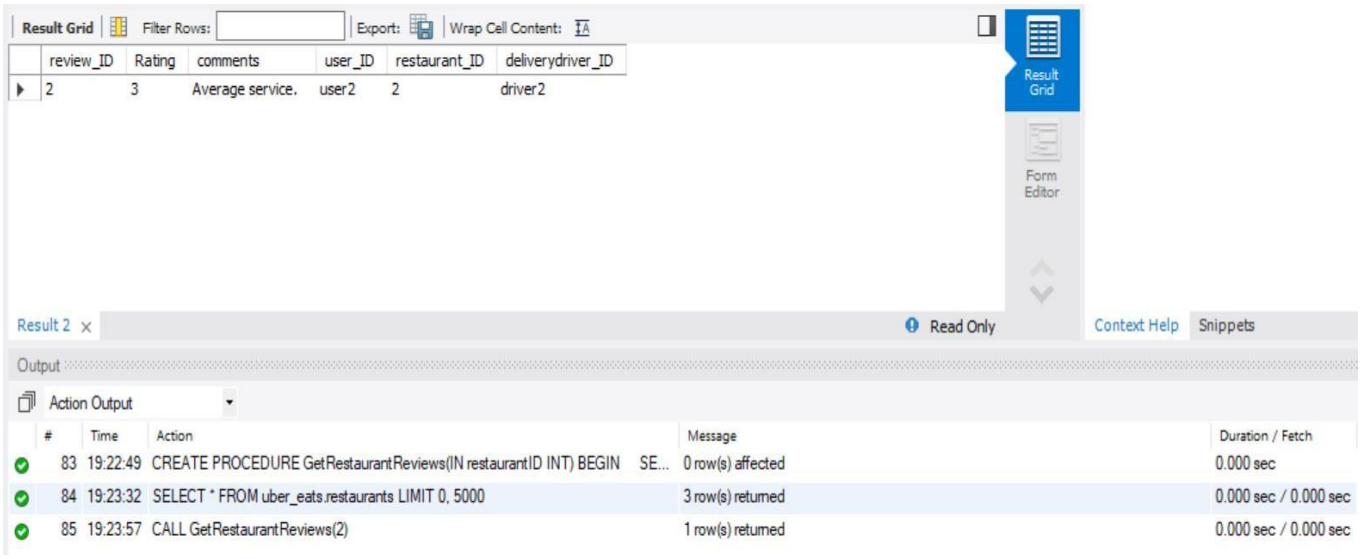
2. Stored procedure to GetRestaurantReviews

The second procedure we created is this GetRestaurantReviews

```
Use Uber_Eats;

 DELIMITER //

 CREATE PROCEDURE GetRestaurantReviews(IN restaurantID INT)
BEGIN
   SELECT *
   FROM reviews
   WHERE restaurant_ID = restaurantID;
END //

 DELIMITER ;

 CALL GetRestaurantReviews(2);
```

Output for the Stored procedure to GetRestaurantReviews:

We can see that the stored procedure has been executed successfully.



Triggers:

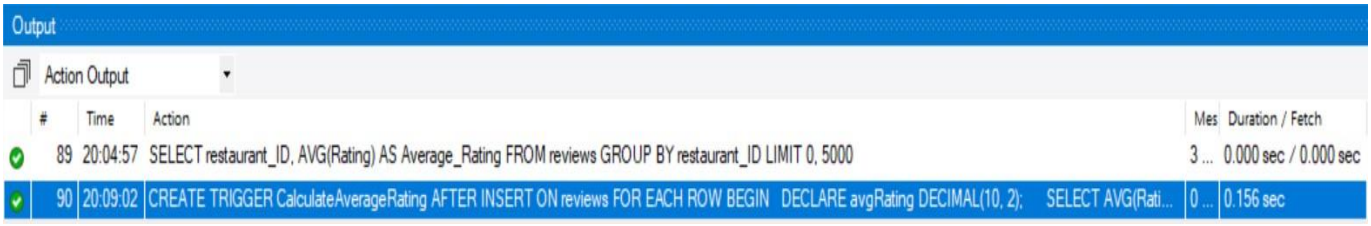
1. Trigger to CalculateAverageRating:

Trigger that we created is the CalculateAverageRating trigger

```
Use Uber_Eats;

DELIMITER //
CREATE TRIGGER CalculateAverageRating AFTER INSERT ON reviews
FOR EACH ROW
BEGIN
  DECLARE avgRating DECIMAL(10, 2);

  SELECT AVG(Rating) INTO avgRating
  FROM reviews
  WHERE restaurant_ID = NEW.restaurant_ID;

  UPDATE restaurants
  SET average_rating = avgRating
  WHERE  restaurant_ID  =  NEW.restaurant_ID;
END //
DELIMITER ;
```

Output for Trigger to CalculateAverageRating:

We can see that the trigger has been executed successfully



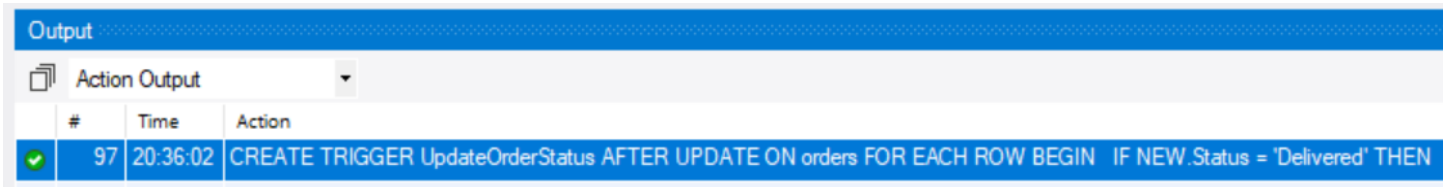
2. Trigger to UpdateOrderStatus:

Another trigger that we created is the UpdateOrderStatus trigger

```
Use Uber_Eats;

DELIMITER //
CREATE TRIGGER UpdateOrderStatus AFTER UPDATE ON orders
FOR EACH ROW
BEGIN
  IF NEW.Status = 'Delivered' THEN
    INSERT INTO reviews (Rating, comments, user_ID, restaurant_ID, deliverydriver_ID)
    VALUES (NULL, NULL, NEW.user_ID, NEW.restaurant_ID, NEW.delivery_ID);
  END IF;
END        //
DELIMITER ;
```

Output for Trigger to UpdateOrderStatus:

We can see that the trigger has been executed successfully



Conclusion:

We have completed all the requirements and the steps of the project in order to successfully represent the Uber Eats Database.