Naxel Santiago NAS180011
Vishnu Yarabarla SXY180042
CS 6360.001

Database Design - Final Project
Uber Eats

1. Data Requirements

Entities:

Person: id, first_name, last_name, date of birth, (age)

Driver: driver id, (^^Person)

Customer: customer id, customer_address, (^^Person)

Food_Order: order_id, customer id, restaurant id, driver_id, order_status, delivery_fee, total_price, requested_delivery_time, rating

Items_Ordered: order_id, {item_name, price, size}, quantity

Menu: menu_id, item_id

Menu_Item: item_id, item_name, price, size

Address: id, street, city, state, zip

Customer Address: (^^Address)

Restaurant Address: (^^Address)

Restaurant: restaurant_id, restaurant_name, restaurant_address

Menu_Item: item_id, item_name, price, size, restaurant id

Relationships:

Addresses can be classified as Customer_Add or Restaurant_Add

Customer has Customer_Add

Customer makes Food_Order which is delivered to Customer_Add

Food_Order contains list of Items_Ordered

Items_Ordered chosen from one Menu with many Menu_Item

Restaurant has one Menu with many Menu_Items

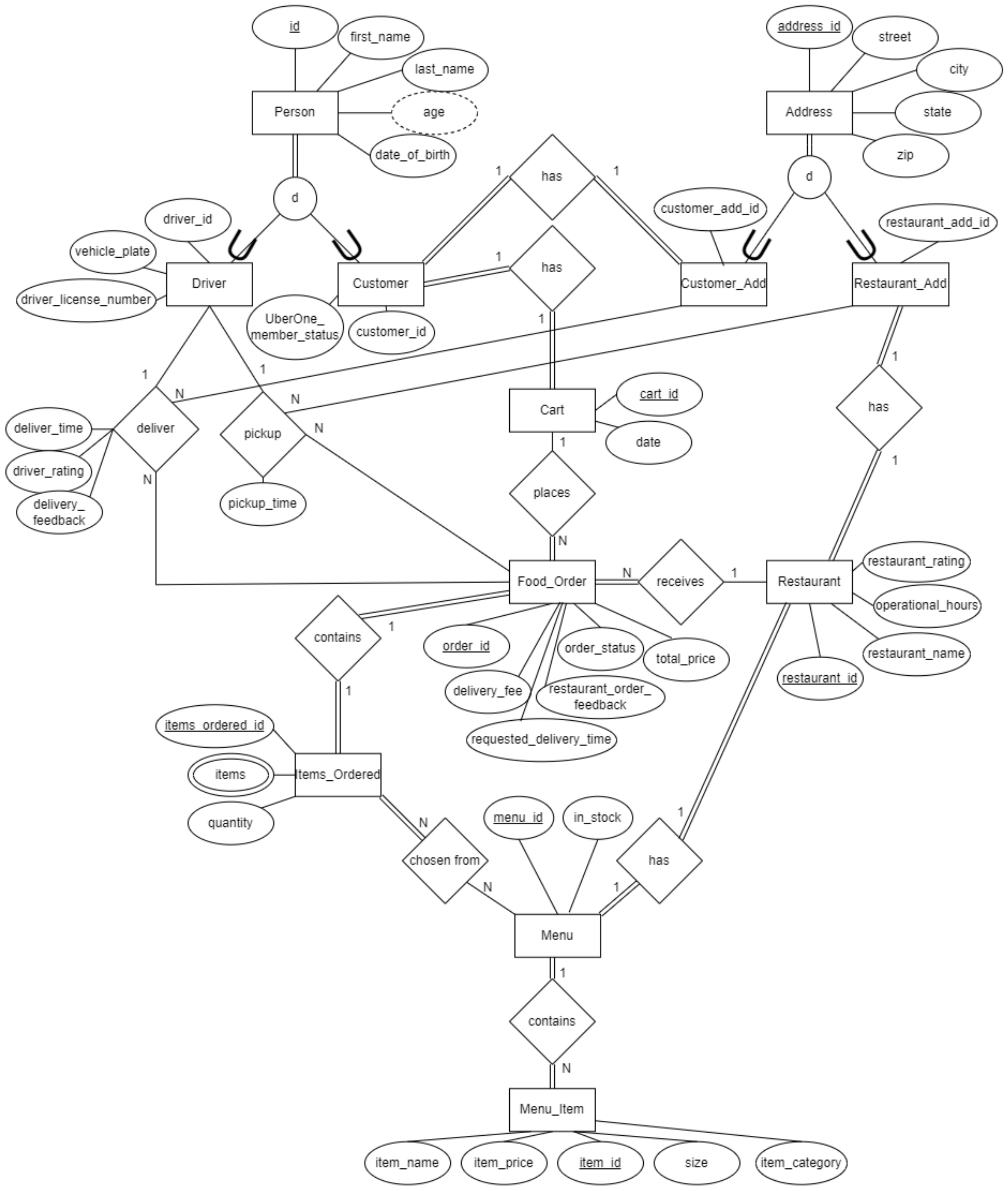
Restaurant receives Food_Order

Restaurant has Restaurant_Add where Food_Order is picked up from

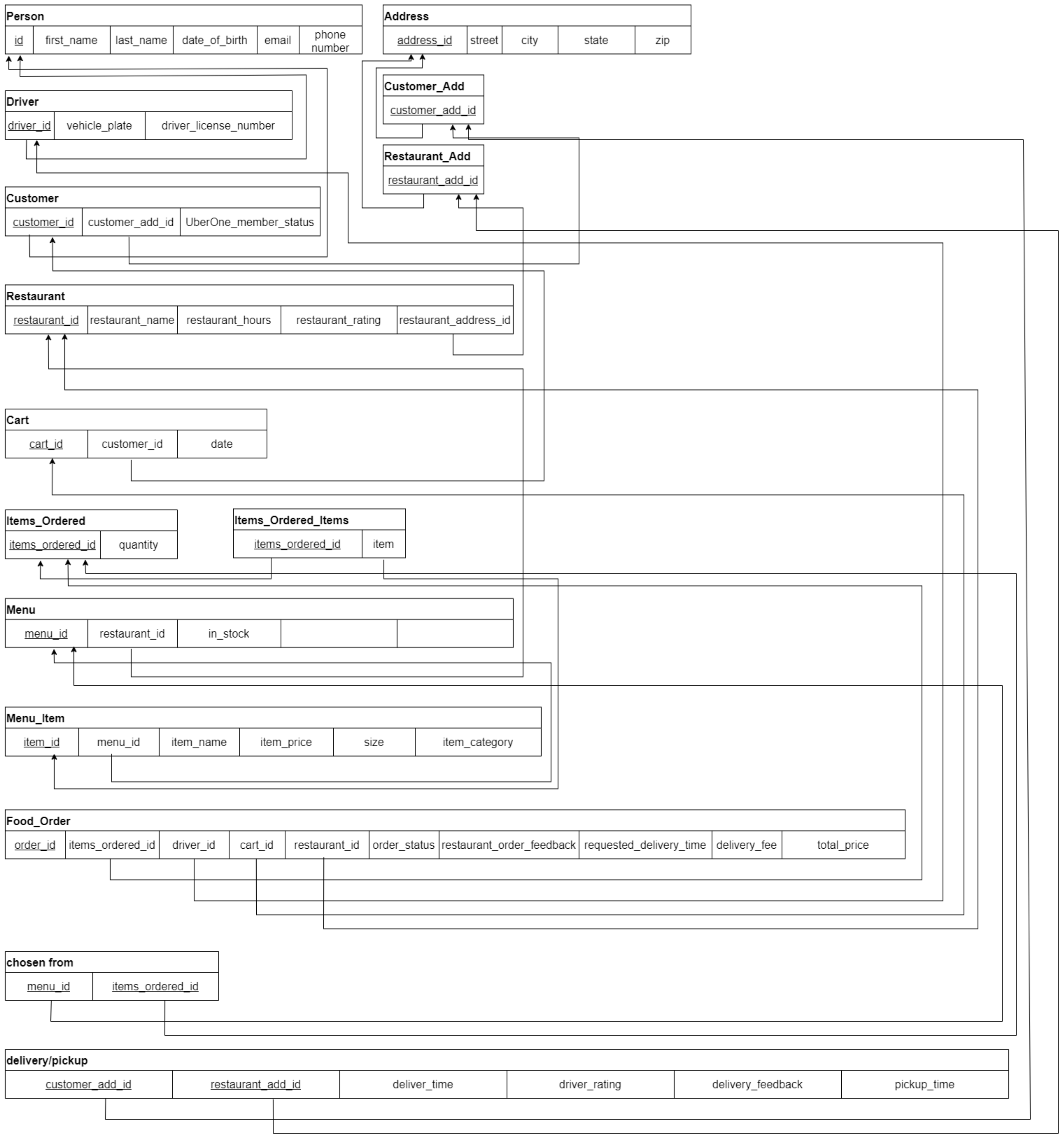
Driver does pickup of Food_Order from Restaurant_Add

Driver delivers Food_Order to Customer_Add

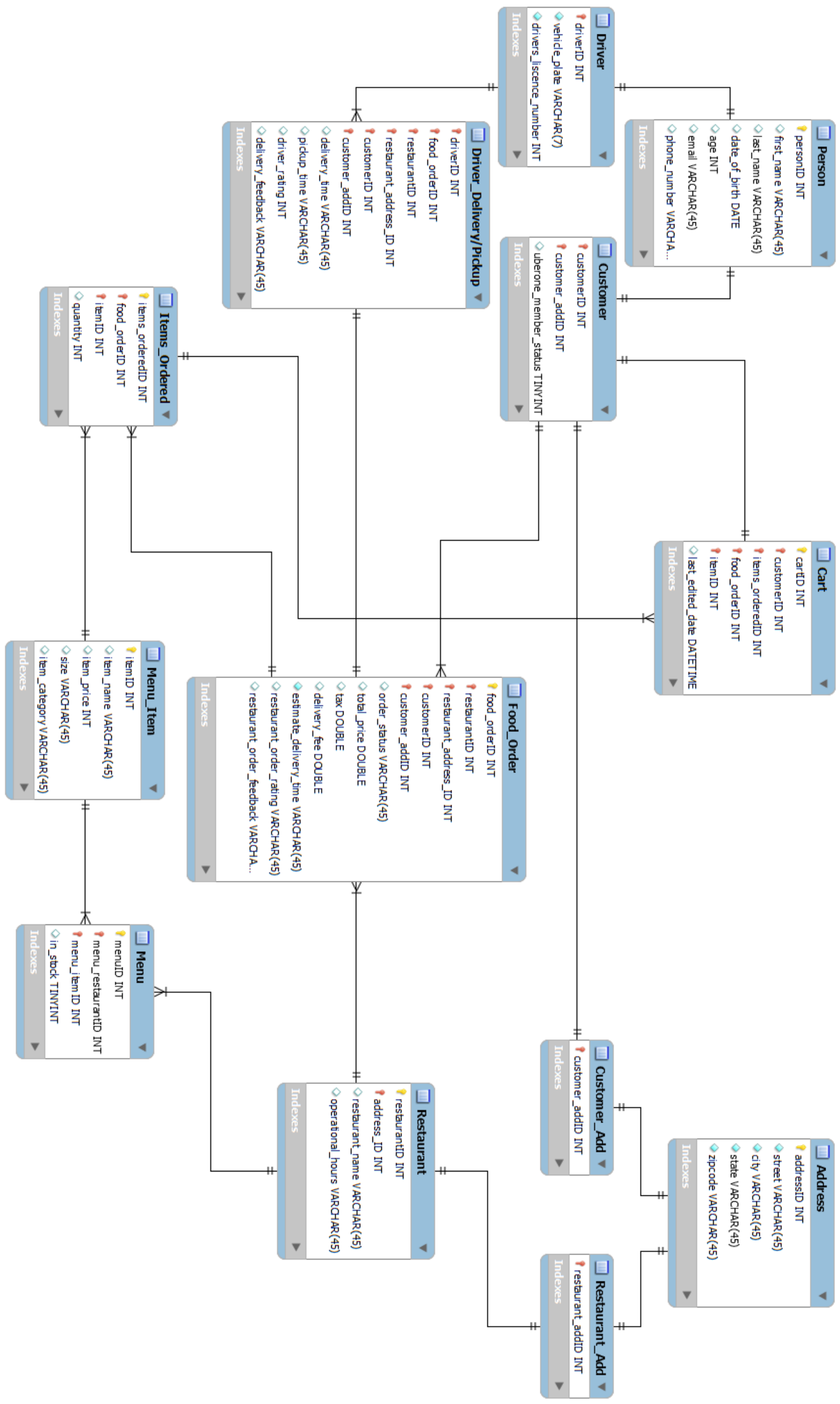
2. Entity Relation Diagram



3. Relational Schema Diagram



4. Relational Schema



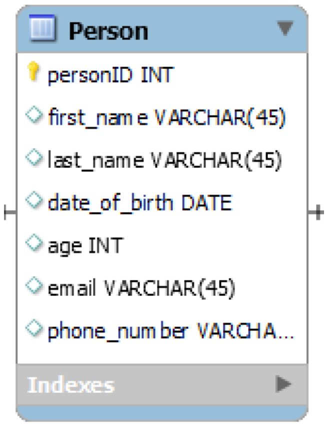
5. Database Normalization

1NF - All attributes depend on the key. Is in 1NF if it does not contain any composite or multi-valued attribute. All attributes are unique.

2NF - If every non-prime attribute A in R is fully functionally dependent on every key of R.

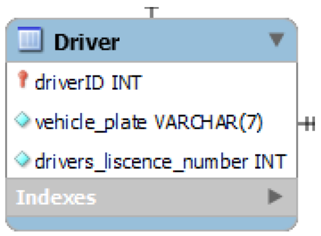
3NF - If it is in 2NF and no non-prime attribute A in R is transitively dependent on the primary key.

Note: Images are from before changes. Updated changes are at the end of this section.



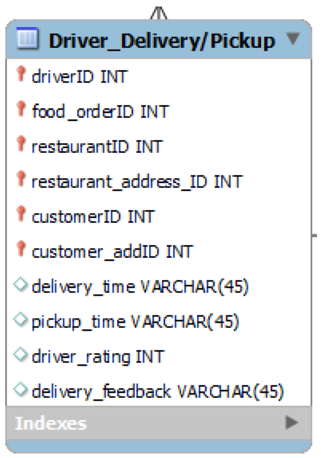
Person
personID INT
first_name VARCHAR(45)
last_name VARCHAR(45)
date_of_birth DATE
age INT
email VARCHAR(45)
phone_number VARCHA...
Indexes

1. Person:
   1NF - ✓
   2NF - ✓
   3NF - X
   Age is dependent on date_of_birth, therefore it is transitively dependent on personID. However, since it is a calculated attribute we believe it should stay in this table. All the attributes pass 3NF



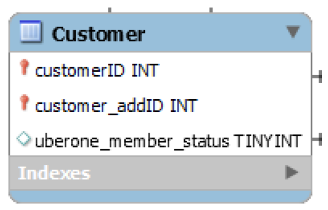
Driver
driverID INT
vehicle_plate VARCHAR(7)
drivers_liscence_number INT
Indexes

2. Driver:
   1NF - ✓
   2NF - ✓
   3NF - ✓

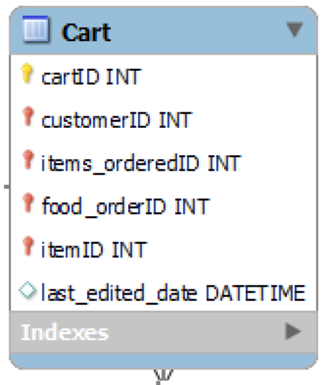


Driver_Delivery/Pickup
driverID INT
food_orderID INT
restaurantID INT
restaurant_address_ID INT
customerID INT
customer_addID INT
delivery_time VARCHAR(45)
pickup_time VARCHAR(45)
driver_rating INT
delivery_feedback VARCHAR(45)
Indexes

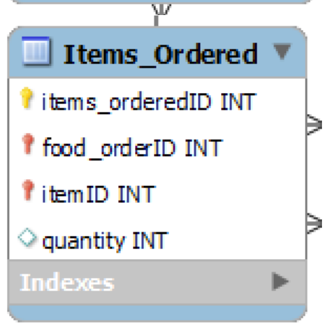
3. Driver_Delivery/Pickup:
   First we remove the two addresses since they are redundant.
   1NF - ✓
   2NF - X due to customerID -> pickup_time, since pickup should only be {driverID, food_orderID, restaurantID} -> pickup_time
   3NF - X
   Therefore we will split up delivery and pickup.
   After split both tables are in 3NF



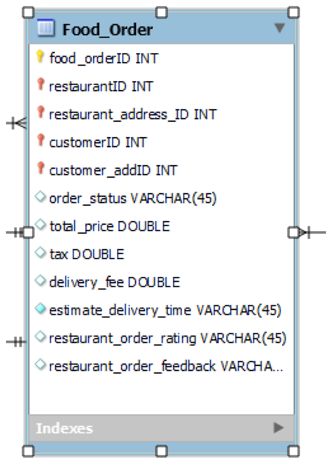
4. Customer:
   1NF - ✓
   2NF - X
   3NF - X
   Uberone_member is not dependent on the whole key.
   We will make customerID the only primary key, that way the table can be in 3NF.



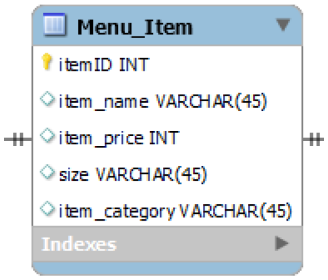
5. Cart:
   Removed cartID since it was not needed. Removed food_orderID and itemID since they were redundant. Added item_discount and checked_out attribute.
   1NF - ✓
   2NF - ✓
   3NF - ✓
   Remaining attributes are all dependent on customerID and items_orderedID.



6. Items_Ordered:
   1NF - ✓
   2NF - ✓
   3NF - ✓
   We kept items_orderedID in order to have it as a foreign key in Cart.



7. Food_Order:
   1NF - ✓
   2NF - X
   3NF - X
   First we reduce the primary keys to food_orderID, restaurantID and CustomerID. Since the two addresses shouldn't be primary keys.
   Now the remaining attributes are all dependent on food_orderID, restaurantID and customerID. note: the feedback attributes should be dependent on the food_order as well and not just customer and restaurant.



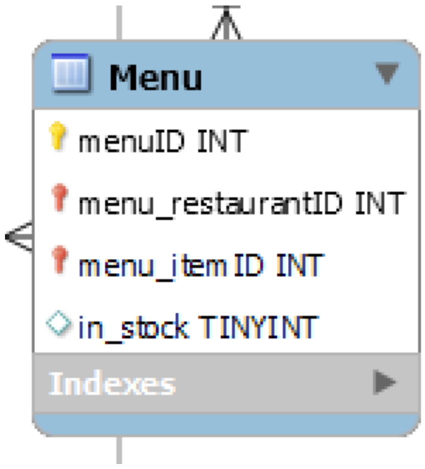
8. Menu_Item:

   1NF – ✓

   2NF – ✓

   3NF – ✓

   We keep itemID here since we also use it as a foreign key for Items_Ordered. All of our attributes are fully functionally dependent on our primary key with no partial dependencies. There are no transitive dependencies since none of our attributes depends on a non-primary key attribute.



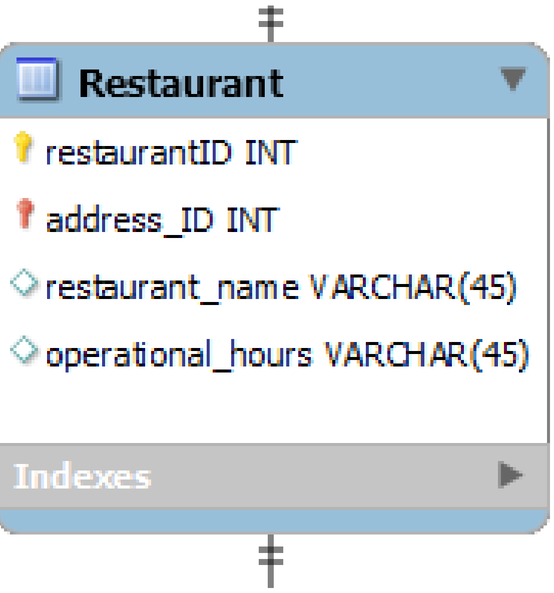
9. Menu:

   1NF – ✓

   2NF – ✓

   3NF – ✓

   Removed menuID since it was not needed. We can use menu_restaurantID and menu_itemID as our composite primary key since we need the specific restaurant and the item from that restaurant to determine the specific menu items. In_stock is our only non-primary key attribute and is fully functionally dependent on both primary keys we chose so there is no partial dependency and it is in 2NF. There are no transitive dependencies since in_stock depends directly on our primary keys and not another non-prime attribute.



10. Restaurant:

    1NF – ✓

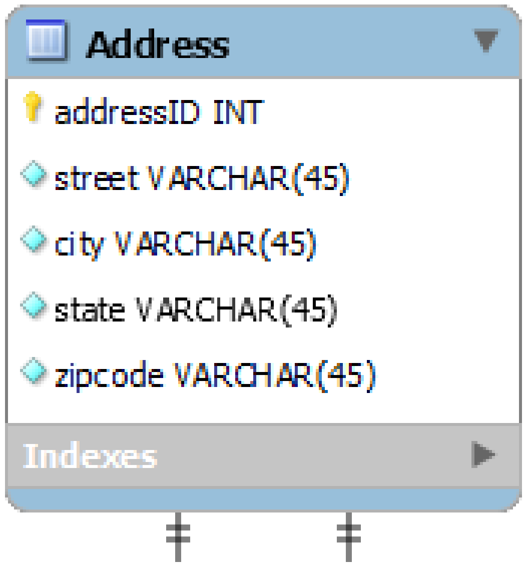
    2NF – ✓

    3NF – ✓

    restaurantID is kept since we need to differentiate between different chain locations of the same restaurant or different versions of that restaurant (Taco Bell vs Taco Bell Cantina). Restaurant does not have composite or multivalued attributes so 1NF is met. We use restaurantID as our primary key. We turn address_ID from a foreign key into a normal attribute.

address_ID, restaurant_name and operational_hours all fully depend on restaurantID. There are no transitive dependencies since none of our nonprime attributes depend on another nonprime attribute.

11. Address:



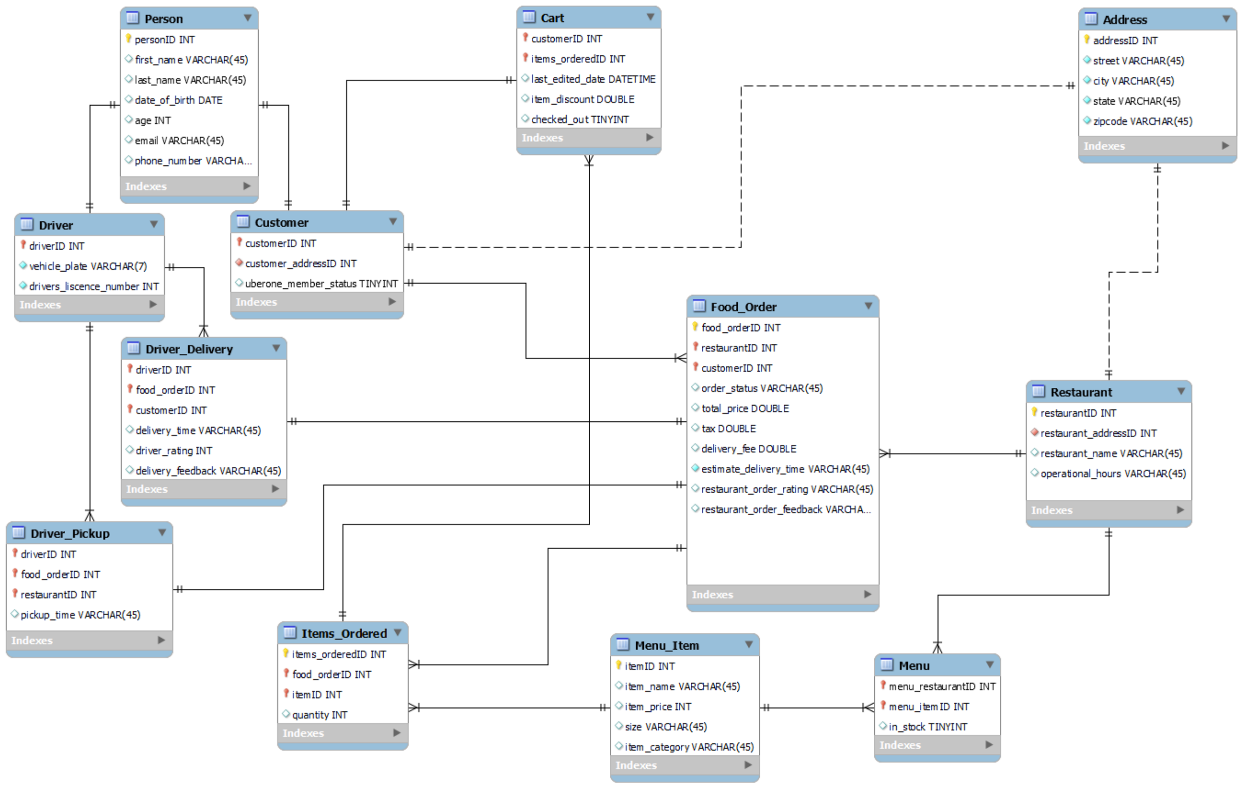
1NF – ✓

2NF – ✓

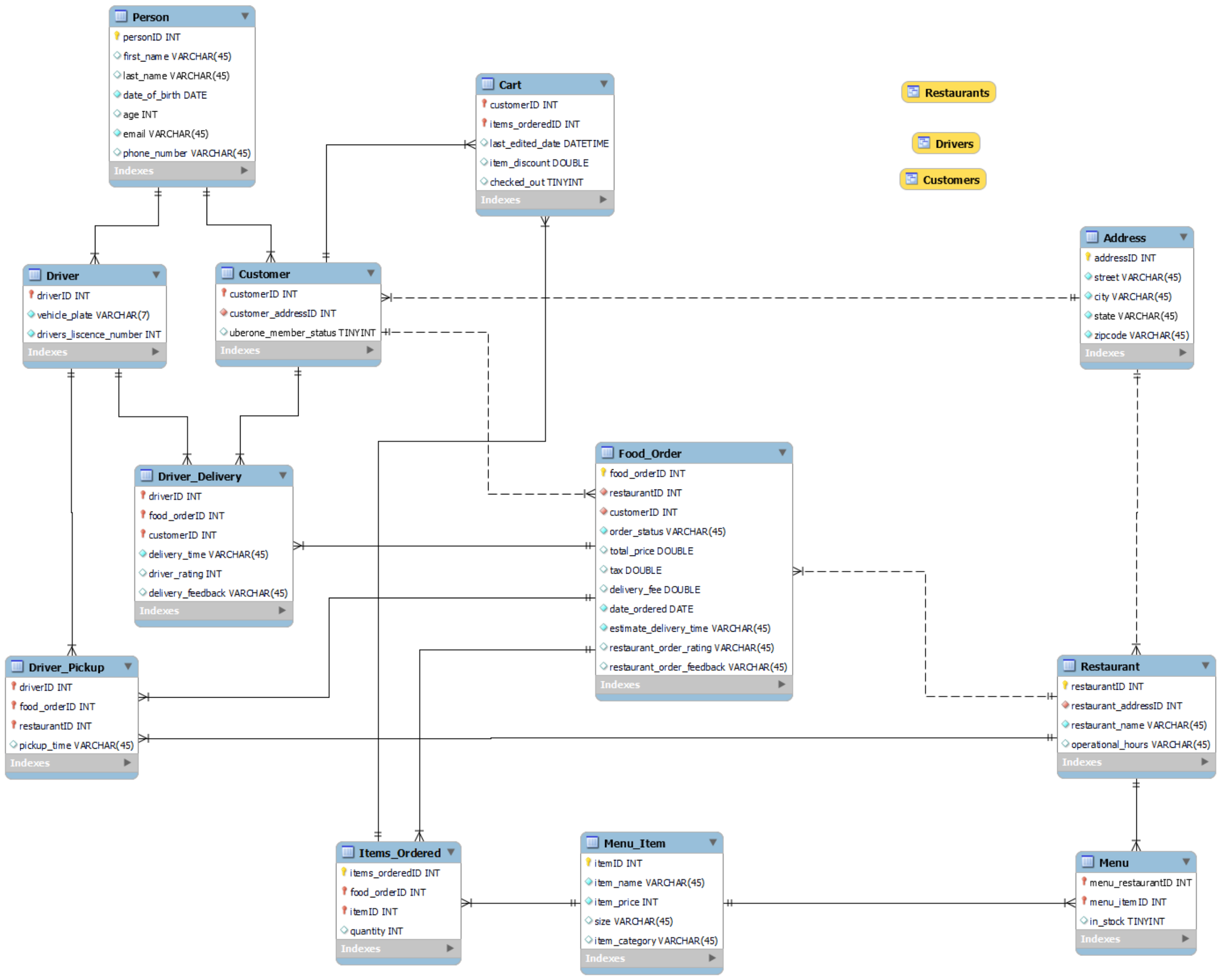
3NF – ✓

There are no composite/multi-valued attributes. addressID remains as our primary key and all attributes have a complete dependency on addressID. All the other attributes are non-prime attributes and all rely on our primary key and not another non-prime attribute so 3NF is met.

This is the final Relational Schema for UberEats:



6. SQL Tables

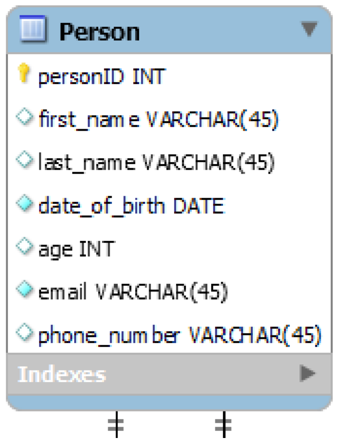


SQL Script:

```
-- MySQL Script generated by MySQL Workbench
-- Sun Apr 23 01:48:11 2023
-- Model: New Model    Version: 1.0
-- MySQL Workbench Forward Engineering

SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_
ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION';

-- -----------------------------------------------------
-- Schema UberEats
-- -----------------------------------------------------
-- -----------------------------------------------------
-- Schema UberEats
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `UberEats` DEFAULT CHARACTER SET utf8 ;
USE `UberEats` ;
```

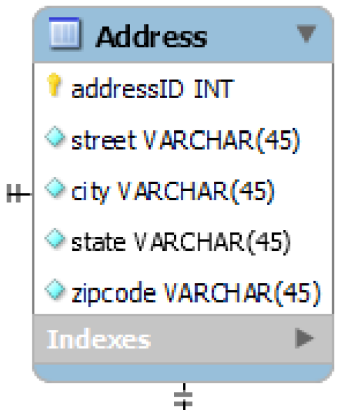
-- ---
-- Table `Person`
-- ---

```
DROP TABLE IF EXISTS `Person` ;

CREATE TABLE IF NOT EXISTS `Person` (
  `personID` INT NOT NULL AUTO_INCREMENT,
  `first_name` VARCHAR(45) NULL,
  `last_name` VARCHAR(45) NULL,
  `date_of_birth` DATE NOT NULL,
  `age` INT GENERATED ALWAYS AS (0) VIRTUAL,
  `email` VARCHAR(45) NOT NULL,
  `phone_number` VARCHAR(45) NULL,
  PRIMARY KEY (`personID`),
  UNIQUE INDEX `PersonID_UNIQUE` (`personID` ASC) VISIBLE)
ENGINE = InnoDB;
-- SELECT timestampdiff(YEAR, date_of_birth, CURDATE()) as age;
```

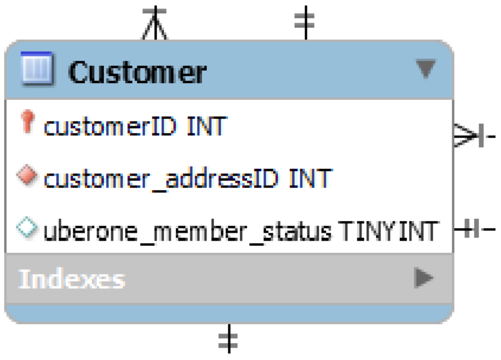


-- ---
-- Table `Address`
-- ---

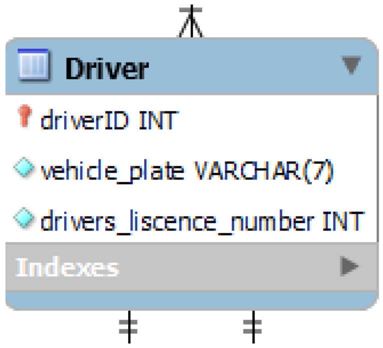
```
DROP TABLE IF EXISTS `Address` ;

CREATE TABLE IF NOT EXISTS `Address` (
  `addressID` INT NOT NULL AUTO_INCREMENT,
  `street` VARCHAR(45) NOT NULL,
  `city` VARCHAR(45) NOT NULL,
  `state` VARCHAR(45) NOT NULL,
  `zipcode` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`addressID`),
  UNIQUE INDEX `addressID_UNIQUE` (`addressID` ASC) VISIBLE)
ENGINE = InnoDB;
```



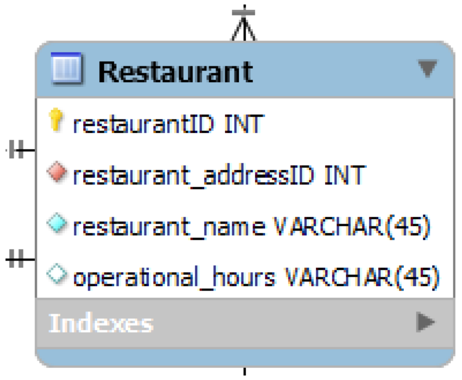
```
-- -----------------------------------------------------
-- Table `Customer`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `Customer` ;

CREATE TABLE IF NOT EXISTS `Customer` (
  `customerID` INT NOT NULL,
  `customer_addressID` INT NOT NULL,
  `uberone_member_status` TINYINT NULL DEFAULT 0,
  PRIMARY KEY (`customerID`),
  INDEX `fk_Customer_Person_idx` (`customerID` ASC) VISIBLE,
  INDEX `fk_Customer_Address1_idx` (`customer_addressID` ASC) VISIBLE,
  UNIQUE INDEX `customerID_UNIQUE` (`customerID` ASC) VISIBLE,
  UNIQUE INDEX `customer_addressID_UNIQUE` (`customer_addressID` ASC) VISIBLE,
  CONSTRAINT `fk_Customer_Person`
    FOREIGN KEY (`customerID`)
    REFERENCES `Person` (`personID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Customer_Address1`
    FOREIGN KEY (`customer_addressID`)
    REFERENCES `Address` (`addressID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```



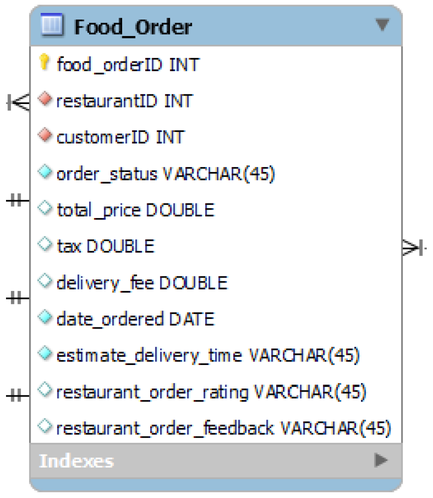
```
-- -----------------------------------------------------
-- Table `Driver`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `Driver` ;

CREATE TABLE IF NOT EXISTS `Driver` (
  `driverID` INT NOT NULL,
  `vehicle_plate` VARCHAR(7) NOT NULL,
  `drivers_liscence_number` INT NOT NULL,
  PRIMARY KEY (`driverID`),
  UNIQUE INDEX `vehicle_plate_UNIQUE` (`vehicle_plate` ASC) VISIBLE,
  UNIQUE INDEX `drivers_liscence_UNIQUE` (`drivers_liscence_number` ASC) VISIBLE,
  UNIQUE INDEX `driverID_UNIQUE` (`driverID` ASC) VISIBLE,
  CONSTRAINT `fk_Driver_Person1`
    FOREIGN KEY (`driverID`)
    REFERENCES `Person` (`personID`)
    ON DELETE NO ACTION
```



```
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `Restaurant`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `Restaurant` ;

CREATE TABLE IF NOT EXISTS `Restaurant` (
  `restaurantID` INT NOT NULL AUTO_INCREMENT,
  `restaurant_addressID` INT NOT NULL,
  `restaurant_name` VARCHAR(45) NOT NULL,
  `operational_hours` VARCHAR(45) NULL DEFAULT '24/7',
  PRIMARY KEY (`restaurantID`),
  UNIQUE INDEX `restaurantID_UNIQUE` (`restaurantID` ASC) VISIBLE,
  INDEX `fk_Restaurant_Address1_idx` (`restaurant_addressID` ASC) VISIBLE,
  UNIQUE INDEX `restaurant_addressID_UNIQUE` (`restaurant_addressID` ASC) VISIBLE,
  CONSTRAINT `fk_Restaurant_Address1`
    FOREIGN KEY (`restaurant_addressID`)
    REFERENCES `Address` (`addressID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

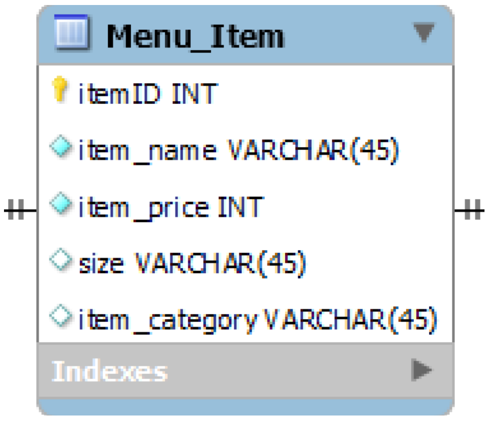


```
-- -----------------------------------------------------
-- Table `Food_Order`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `Food_Order` ;

CREATE TABLE IF NOT EXISTS `Food_Order` (
  `food_orderID` INT NOT NULL AUTO_INCREMENT,
  `restaurantID` INT NOT NULL,
  `customerID` INT NOT NULL,
  `order_status` VARCHAR(45) NOT NULL DEFAULT 'Processing Payment',
  `total_price` DOUBLE NULL,
  `tax` DOUBLE NULL,
  `delivery_fee` DOUBLE NULL,
  `date_ordered` date NOT NULL DEFAULT (curdate()),
  `estimate_delivery_time` VARCHAR(45) NOT NULL,
  `restaurant_order_rating` VARCHAR(45) NULL,
  `restaurant_order_feedback` VARCHAR(45) NULL,
  PRIMARY KEY (`food_orderID`),
```

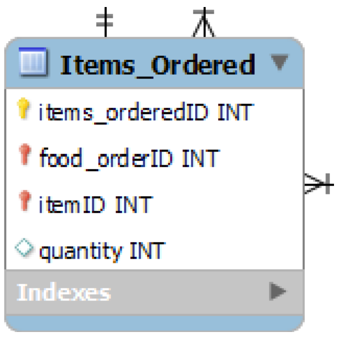


```
  UNIQUE INDEX `food_orderID_UNIQUE` (`food_orderID` ASC) VISIBLE,
  INDEX `fk_Food_Order_Restaurant1_idx` (`restaurantID` ASC) VISIBLE,
  INDEX `fk_Food_Order_Customer1_idx` (`customerID` ASC) VISIBLE,
  UNIQUE INDEX `restaurantID_UNIQUE` (`restaurantID` ASC) VISIBLE,
  UNIQUE INDEX `customerID_UNIQUE` (`customerID` ASC) VISIBLE,
  CONSTRAINT `fk_Food_Order_Restaurant1`
    FOREIGN KEY (`restaurantID`)
    REFERENCES `Restaurant` (`restaurantID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Food_Order_Customer1`
    FOREIGN KEY (`customerID`)
    REFERENCES `Customer` (`customerID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

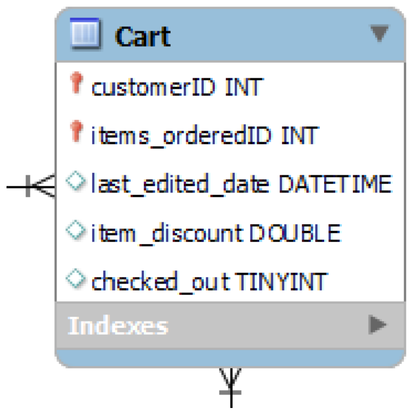
```
-- -----------------------------------------------------
-- Table `Menu_Item`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `Menu_Item` ;

CREATE TABLE IF NOT EXISTS `Menu_Item` (
  `itemID` INT NOT NULL AUTO_INCREMENT,
  `item_name` VARCHAR(45) NOT NULL,
  `item_price` INT NOT NULL,
  `size` VARCHAR(45) NULL DEFAULT 'undetermined',
  `item_category` VARCHAR(45) NULL,
  PRIMARY KEY (`itemID`),
  UNIQUE INDEX `idMenu_Item_UNIQUE` (`itemID` ASC) VISIBLE)
ENGINE = InnoDB;
```



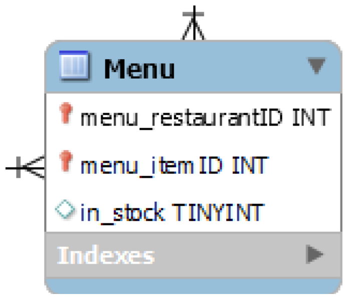
```
-- -----------------------------------------------------
-- Table `Items_Ordered`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `Items_Ordered` ;

CREATE TABLE IF NOT EXISTS `Items_Ordered` (
  `items_orderedID` INT NOT NULL AUTO_INCREMENT,
  `food_orderID` INT NOT NULL,
  `itemID` INT NOT NULL,
  `quantity` INT NULL DEFAULT 1,
```



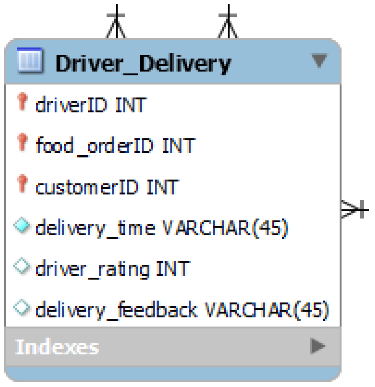
```
  PRIMARY KEY (`items_orderedID`, `food_orderID`, `itemID`),
  INDEX `fk_Food_Order_has_Menu_Item_Menu_Item1_idx` (`itemID` ASC) VISIBLE,
  INDEX `fk_Food_Order_has_Menu_Item_Food_Order1_idx` (`food_orderID` ASC) VISIBLE,
  UNIQUE INDEX `items_orderedID_UNIQUE` (`items_orderedID` ASC) VISIBLE,
  CONSTRAINT `fk_Food_Order_has_Menu_Item_Food_Order1`
    FOREIGN KEY (`food_orderID`)
    REFERENCES `Food_Order` (`food_orderID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Food_Order_has_Menu_Item_Menu_Item1`
    FOREIGN KEY (`itemID`)
    REFERENCES `Menu_Item` (`itemID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `Cart`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `Cart` ;

CREATE TABLE IF NOT EXISTS `Cart` (
  `customerID` INT NOT NULL,
  `items_orderedID` INT NOT NULL,
  `last_edited_date` DATETIME NULL DEFAULT (curdate()),
  `item_discount` DOUBLE NULL,
  `checked_out` TINYINT NULL DEFAULT 0,
  PRIMARY KEY (`items_orderedID`, `customerID`),
  INDEX `fk_Cart_Items_Ordered1_idx` (`items_orderedID` ASC) VISIBLE,
  INDEX `fk_Cart_Customer1_idx` (`customerID` ASC) VISIBLE,
  CONSTRAINT `fk_Cart_Items_Ordered1`
    FOREIGN KEY (`items_orderedID`)
    REFERENCES `Items_Ordered` (`items_orderedID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Cart_Customer1`
    FOREIGN KEY (`customerID`)
    REFERENCES `Customer` (`customerID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```



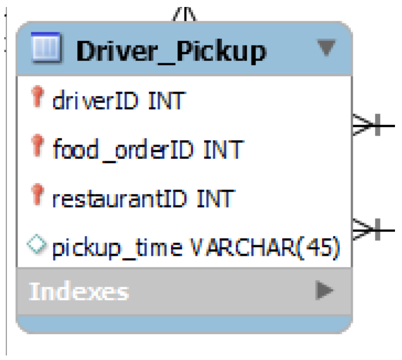
```
-- -----------------------------------------------------
-- Table `Menu`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `Menu` ;

CREATE TABLE IF NOT EXISTS `Menu` (
  `menu_restaurantID` INT NOT NULL,
  `menu_itemID` INT NOT NULL,
  `in_stock` TINYINT NULL DEFAULT 0,
  PRIMARY KEY (`menu_restaurantID`, `menu_itemID`),
  INDEX `fk_Menu_Menu_Item1_idx` (`menu_itemID` ASC) VISIBLE,
  CONSTRAINT `fk_Menu_Restaurant1`
    FOREIGN KEY (`menu_restaurantID`)
    REFERENCES `Restaurant` (`restaurantID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Menu_Menu_Item1`
    FOREIGN KEY (`menu_itemID`)
    REFERENCES `Menu_Item` (`itemID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```



```
-- -----------------------------------------------------
-- Table `Driver_Delivery`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `Driver_Delivery` ;

CREATE TABLE IF NOT EXISTS `Driver_Delivery` (
  `driverID` INT NOT NULL,
  `food_orderID` INT NOT NULL,
  `customerID` INT NOT NULL,
  `delivery_time` VARCHAR(45) NOT NULL,
  `driver_rating` INT NULL,
  `delivery_feedback` VARCHAR(45) NULL,
  PRIMARY KEY (`driverID`, `food_orderID`, `customerID`),
  UNIQUE INDEX `driverID_UNIQUE` (`driverID` ASC) VISIBLE,
  INDEX `fk_Driver_Delivery_Food_Order1_idx` (`food_orderID` ASC) VISIBLE,
  INDEX `fk_Driver_Delivery_Customer1_idx` (`customerID` ASC) VISIBLE,
  CONSTRAINT `fk_Driver_Delivery_Driver1`
    FOREIGN KEY (`driverID`)
    REFERENCES `Driver` (`driverID`)
    ON DELETE NO ACTION
```



```
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Driver_Delivery_Food_Order1`
    FOREIGN KEY (`food_orderID`)
    REFERENCES `Food_Order` (`food_orderID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Driver_Delivery_Customer1`
    FOREIGN KEY (`customerID`)
    REFERENCES `Customer` (`customerID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```



```
-- -----------------------------------------------------
-- Table `Driver_Pickup`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `Driver_Pickup` ;

CREATE TABLE IF NOT EXISTS `Driver_Pickup` (
  `driverID` INT NOT NULL,
  `food_orderID` INT NOT NULL,
  `restaurantID` INT NOT NULL,
  `pickup_time` VARCHAR(45) NULL,
  PRIMARY KEY (`driverID`, `food_orderID`, `restaurantID`),
  INDEX `fk_Driver_Pickup_Food_Order1_idx` (`food_orderID` ASC) VISIBLE,
  INDEX `fk_Driver_Pickup_Restaurant1_idx` (`restaurantID` ASC) VISIBLE,
  CONSTRAINT `fk_Driver_Delivery_Driver10`
    FOREIGN KEY (`driverID`)
    REFERENCES `Driver` (`driverID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Driver_Pickup_Food_Order1`
    FOREIGN KEY (`food_orderID`)
    REFERENCES `Food_Order` (`food_orderID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_Driver_Pickup_Restaurant1`
    FOREIGN KEY (`restaurantID`)
    REFERENCES `Restaurant` (`restaurantID`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;
```

```
-- -----------------------------------------------------
-- View `customer`
-- -----------------------------------------------------
DROP VIEW IF EXISTS Customers;

CREATE VIEW Customers AS
SELECT personID, first_name, last_name, date_of_birth, age, email, phone_number,
uberone_member_status, street, city, state, zipcode
FROM person, customer, address
WHERE personID = customerID AND customer_addressID = addressID;

-- -----------------------------------------------------
-- View `drivers`
-- -----------------------------------------------------
DROP VIEW IF EXISTS Drivers;

CREATE VIEW Drivers AS
SELECT personID, first_name, last_name, date_of_birth, age, email, phone_number,
vehicle_plate, drivers_liscence_number
FROM person, driver
WHERE personID = driverID;

-- -----------------------------------------------------
-- View `restaurants`
-- -----------------------------------------------------
DROP VIEW IF EXISTS Restaurants;

CREATE VIEW Restaurants AS
SELECT restaurantID, restaurant_name, operational_hours, `addressID`, `street`, `city`, `state`,
`zipcode`
FROM restaurant, address
WHERE restaurant_addressID = addressID;


SET FOREIGN_KEY_CHECKS = 1;
-- SET SQL_MODE=@OLD_SQL_MODE;
-- SET FOREIGN_KEY_CHECKS=1;
-- SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

Procedures:

The addCustomer procedure will allow for the insert of a customer with their respective address and person tables being auto filled.

```
-- -----------------------------------------------------
-- Procedure `addCustomer`
-- -----------------------------------------------------
DROP PROCEDURE IF EXISTS addCustomer;

delimiter //
CREATE PROCEDURE addCustomer (IN varFN varchar(45), IN varLN varchar(45), IN varDoB
date, IN varEmail varchar(45), IN varPN varchar(45),
                                        IN varSt varchar(45), IN varCity varchar(45),
IN varState varchar(45), IN varZip varchar(45), IN varMem tinyint)
BEGIN
      INSERT INTO person(`first_name`,  `last_name`, `date_of_birth`, `email`,
`phone_number`) VALUES(varFN, varLN, varDoB, varEmail, varPN);
  INSERT INTO address(`street`, `city`, `state`, `zipcode`) VALUES(varSt, varCity, varState,
varZip);
  SELECT personID INTO @returnedCustomerID FROM person WHERE person.email =
varEmail;
  SELECT addressID INTO @returnedAddressID FROM address WHERE address.street =
varSt AND address.zipcode = varZip;
  INSERT INTO customer(`customerID`, `customer_addressID`, `uberone_member_status`)
VALUES(@returnedCustomerID, @returnedAddressID, varMem);
END//
delimiter ;
```

The addDriver procedure will allow for the insert of a driver with their respective person table being auto filled.

```
-- -----------------------------------------------------
-- Procedure `addDriver`
-- -----------------------------------------------------
DROP PROCEDURE IF EXISTS addDriver;

delimiter //
CREATE PROCEDURE addDriver (IN varFN varchar(45), IN varLN varchar(45), IN varDoB
date, IN varEmail varchar(45), IN varPN varchar(45),
                                        IN varPlate varchar(7), IN varLisc int)
BEGIN
      INSERT INTO person(`first_name`,  `last_name`, `date_of_birth`, `email`,
`phone_number`) VALUES(varFN, varLN, varDoB, varEmail, varPN);
  SELECT personID INTO @returnedDriverID FROM person WHERE person.email = varEmail;
```

```
    INSERT INTO driver(`driverID`, `vehicle_plate`, `drivers_liscence_number`)
VALUES(@returnedDriverID, varPlate, varLisc);
END//
delimiter ;
```

Triggers:

The calculateAge trigger will calculate and update the person’s age based on their date of birth

```
-- ---------------------------------------------------
-- Trigger `calculateAge` triggers after customer instert
-- ---------------------------------------------------
DROP TRIGGER IF EXISTS calculateAge;

delimiter //
CREATE TRIGGER calculateAge
AFTER INSERT ON customer
FOR EACH ROW
BEGIN
   SELECT date_of_birth INTO @DoB FROM person WHERE personID = NEW.customerID;
   SELECT CAST(TIMESTAMPDIFF(YEAR, @DoB, CURDATE()) AS unsigned) INTO @age;
   UPDATE person SET age = (@age) WHERE personID = NEW.customerID;
END //
delimiter ;
```

The updateOrderStatusDelivered trigger will set the food_order to “delivered” when the driver delivers the food.

```
-- ---------------------------------------------------
-- Trigger `updateOrderStatusDelivered`
-- ---------------------------------------------------
DROP TRIGGER IF EXISTS updateOrderStatusDelivered;

delimiter //
CREATE TRIGGER updateOrderStatusDelivered
AFTER UPDATE ON driver_delivery
FOR EACH ROW
BEGIN
        IF NOT(NEW.delivery_time <=> OLD.delivery_time) THEN
                UPDATE food_order SET order_status = 'Delivered' WHERE
food_order.food_orderID = NEW.food_orderID;
        END IF;
END //
delimiter ;
```