Restaurant Management Database

Portfolio

Christopher Shields

Professor Edward E. Burns

CS04430

2/29/2023

# Table Of Contents

[**Table Of Contents 2**](#_6vpggunsnmwq)

[**1. Requirements Collection & Analysis 3**](#_ynwftttdomj6)

[**2. Conceptual Design 5**](#_t6ruy1bhr90f)

[**3. Logical Design 8**](#_4paikzb8a5t4)

[**4. Physical Design 11**](#_ua8eh8isc0cv)

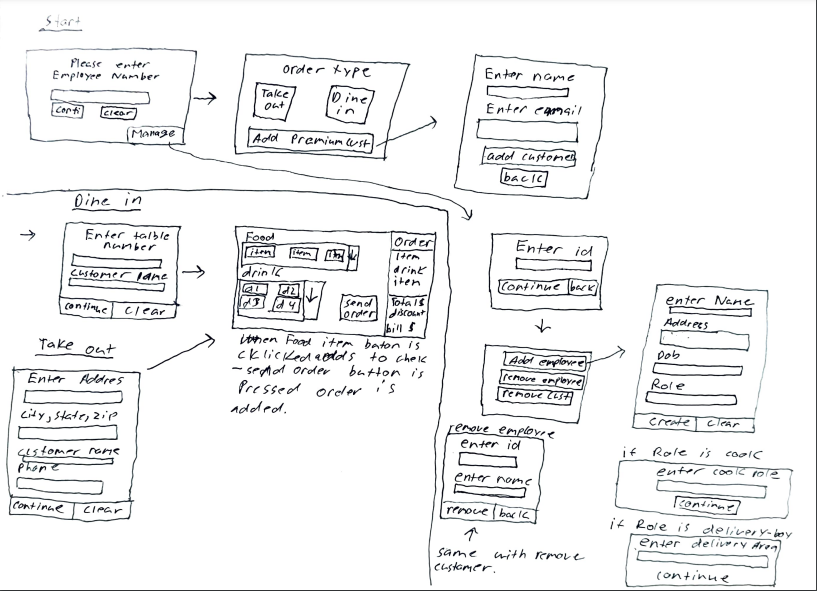
# 1. Requirements Collection & Analysis

The restaurant management database will store all the details necessary to operate a restaurant that takes dine-in and takeout orders. There are four main entities we will be storing, a list of employees, a list of orders, a catalog for food and beverage items, and a list of customers. Details about these actors will also be included in the database. I have also included a diagram of what the possible front-end application could look like. Below are my requirements for this solution:

Restaurant Management System Requirements List:

* There are two types of orders, Dine-in and delivery.
* Delivery orders include an area code, address, and phone number.
* Dine-in includes a table number.
* Orders must be either a delivery or dine-in.
* Orders consist of one or more food items.
* Orders also include the food items, the cost of the order, and an identifying order number.
* An order must always be assigned to one customer
* Customers can request one or more orders
* If a customer has a premium record, then a discount will be assigned to their order
* Dine-in orders are assigned to tables.
* All employees have a name, address, date of birth, and unique identifying employee number.
* There are four types of employees, Servers, Cooks, delivery boys, and managers.
* Delivery boys have an assigned area code(for delivery), cooks have a role (grill, prep, backup), and managers have an id,
* Customers have a name, email customer id, and a customer record.
* Food items will include a name, food cost, identifying item number, and price.

Below I have included a diagram of what a possible front-end application could look like:



# 

# 2. Conceptual Design

For this project, I will create a simple Restaurant Management Database system that will provide all the essential details necessary to operate a restaurant. In this part, I will construct the conceptual design for this database based on the requirements provided by part 1 of the project.

First I will describe the details of any entities, their attributes, and any relationships between different entities. All of this is information that I extracted from the requirements documents. First, there are four primary entities.

* First, any details for food orders will be organized in the Order(Order) relation. Each Order will include a unique order number (order\_num), the cost of the order(order\_cost), any discounts for the order(discount), one or more food items(item\_id), and the id of the customer who requested the order (cust\_id). There will also be two types(sub-tables) of orders:
  + Dine-in orders(Dine\_in\_order), will also include a table number(table\_num)
  + Delivery orders(Delivery\_order), will include an address(order\_address), area code(area\_code), and phone number(phone\_number)
* A Customer (Customer) will include a name (cust\_fname)/(cust\_lname), email address(cust\_email), a unique customer id(cust\_id), and a record of the purchases for the customer(cust\_rec). Along with this, Customers will be able to place any number of orders. Also, if a customer has a good record then a discount will be added to their order.
* The database will also keep track of different beverage or food items(MenuItem). Menu Items will include a name(item\_name), cost(item\_cost), and a unique item id(item\_id). As stated above, one order can have any number of menu items.
* An employee (Employee) will contain a name(emp\_fname)/(emp\_lname), address(emp\_adress), a phone number(emp\_phone) and unique employee number(emp\_num). An employee must be one of four different types:
  + Servers(Server), which will include a table number(table\_num)
  + Deliver boys(DeliveryBoy), will include an area code(area\_code)
  + Cooks (Cook) will have a type(cook\_type), grill, prep, or backup.
  + Managers(Managers) will have a unique manager id(manager\_id) and a role(manager\_role), Front of the house or back of the house.

In terms of the relationships between the different entities:

* Customers can request any number of orders.
* An order will consist of one or more menu items.
* A dine-in order will be assigned to a server through the table number
* A delivery order will be assigned to a delivery boy through an area code

In terms of constraints:

* Orders must be either dine-in or delivery.
* An employee must be one of the four employee types.
* If a customer has a good record, a discount must be applied to an order.
* An order cost will consist of all of the menu item costs added together along with a 7% sales tax.

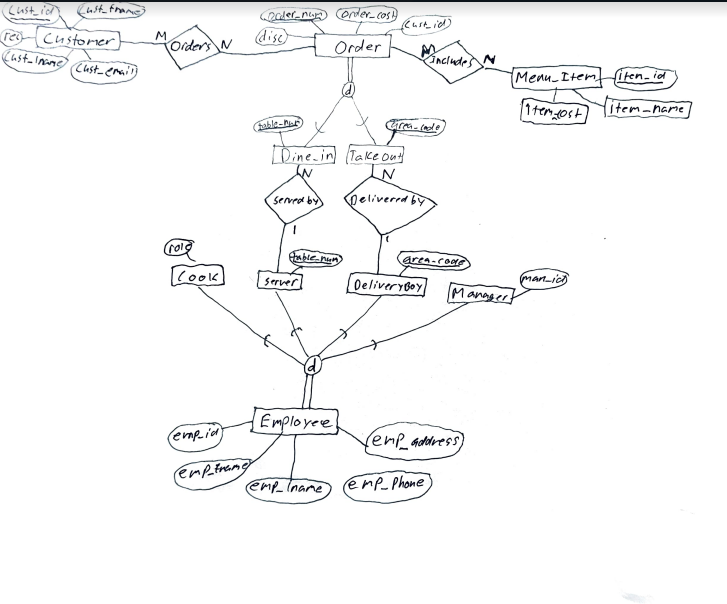
Below is an overview of the different main entities:

* Order(order\_num, order\_cost, discount, cust\_id) Pk = order\_num
* Customer(cust\_id, cust\_fname,cust\_lname,cust\_email,cust\_rec) Pk = cust\_id
* MenuItem(item\_id,item\_name, item\_cost) Pk = item\_id
* Employee(emp\_num,emp\_fname,emp\_lname,emp\_adress, emp\_phone) Pk = emp\_num

Subclasses:

* Server(table\_num)
* Cook(role)
* DeliveryBoy(area\_code)
* Manager(man\_id)
* Dine-in(table\_num)
* TakeOut(area\_code)

Below is the EER diagram that I drew for this project:



# 3. Logical Design

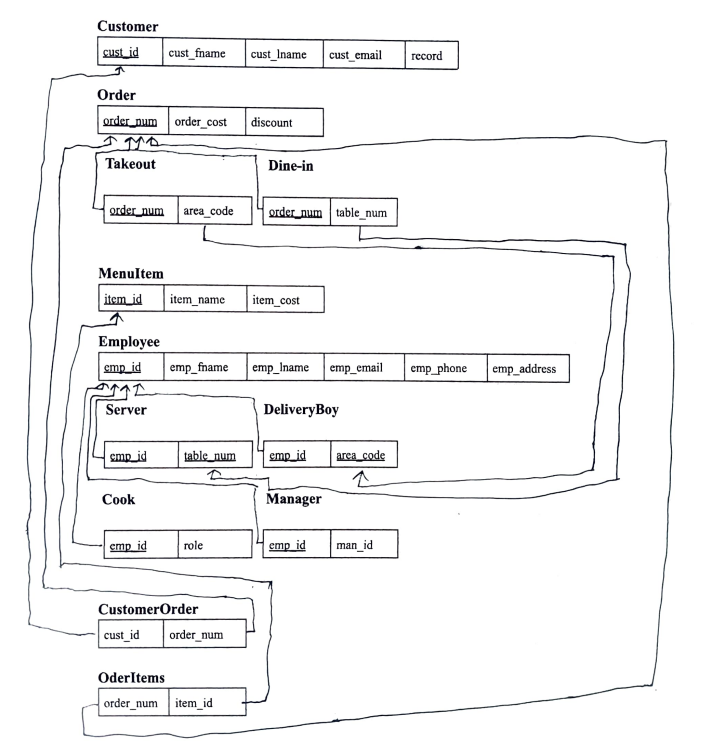
For this part of the semester-long project, I will take the EER diagram I created and provided in part two and translate it into a relational database schema. To do this I will be using the nine-step algorithm to convert EER models into relationship types

1. Mapping of regular entity types: I will create relations for Customer, Order, MenuItem, and Employee as these are regular entity types for my database. I will also be including all of the attributes for these entities.
2. Mapping of weak entity types: I have no weak entity types in the EER so I will be skipping this step.
3. Mapping of Binary 1:1 Relationship types: I have no 1:1 relationship types in the EER so I will be skipping this step.
4. Mapping of Binary 1:N Relationship types: For this step I do have two Binary 1:N relationships but they are subclasses through specialization. I will skip this step for now but come back later after I have mapped specilization.
5. Mapping of Binary M:N Relationship types: I have two Binary M:N Relationship types that I will be mapping from the EER. These are the orders relationship between Customer and Order, and the Includes relationship between Order and MenuItem. I will be mapping both of these by creating a new relation that includes the primary keys of both sides of the relationship as foreign key attributes in the new relation. This means the new Order relation will include cust\_id and order\_num as foreign key attributes; and the new Includes relation will include order\_num and item\_id as foreign key attributes.
6. Mapping of multivalued Atributes: I have no multivalued attributes in the EER so as such I will be skipping this step.
7. Mapping of N-ary Relationship types: I have no N-ary Relationship types in the EER so as such I will be skipping this step.
8. Mapping of Specilzation or Generlization: For this project I have included specilization for two entities, Order and Employee. Order has two subclasses Dine-in and TakeOut; whereas, Employee has four subclasses: Cook, Server, DeliverBoy, and Manager. To map this, I will create a relation for each subclass that includes the primary key of the superclass as the primary key of the new relation. For example, The new Dine-in relation will include Order\_num as its primary key.

Note: Now that I have mapped specilization, I am ready to go back to step 4 and mapp all binary 1:N relationship types. There are two of these types in the EER. ServedBy between server and Dine-in and Delivered By between DeliverBoy and TakeOut. To mapp this, I will be including in the Dine-in relation the primary key of Server(table\_num) as a foreign key atribute. Likewise, I will be including in the Take-out relation the primary key of DeliveryBoy(area\_code) as a foreign key atribute

1. Maping of Union Types (Catagories): I have no union types in the EER so I will be skipping this step.

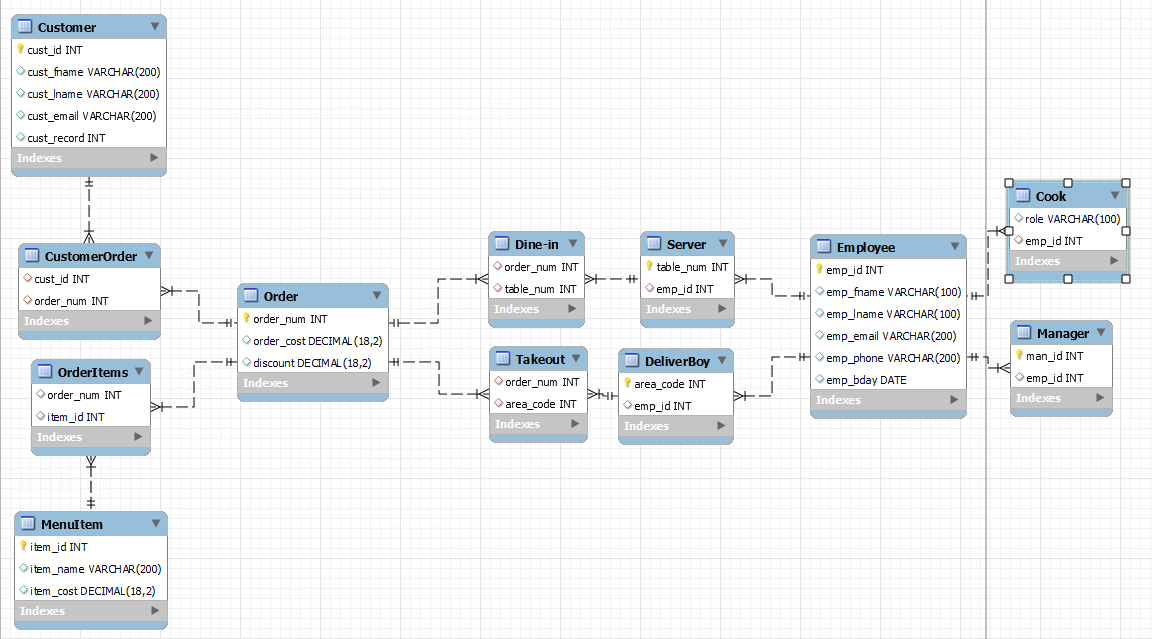
Now that I have completed all of the steps for mapping an EER to a relational schema, I have included a picture of my relational schema on the next page:



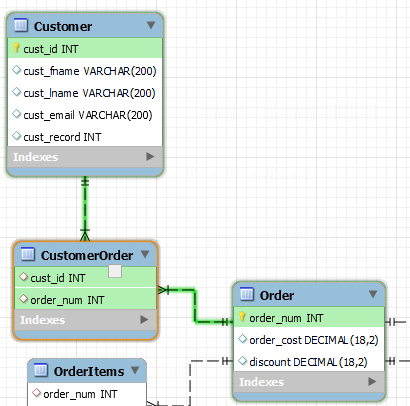
# 4. Physical Design

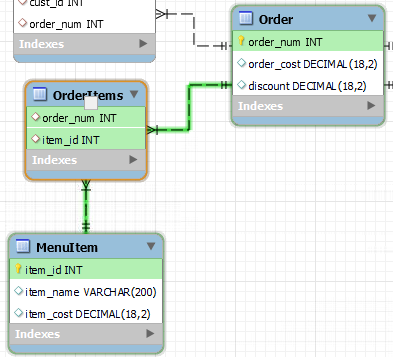
For the final part of this project, I will be taking the relational diagram I constructed in part 3 and building the actual physical tables in the database.

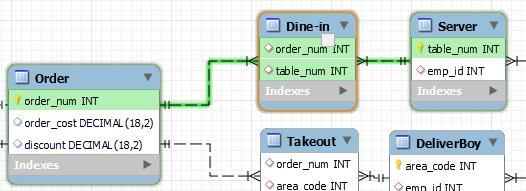
First, I will be utilizing MySQL Workbench to build a diagram of the tables based on my relational diagram. I have included A screenshot of this diagram below:

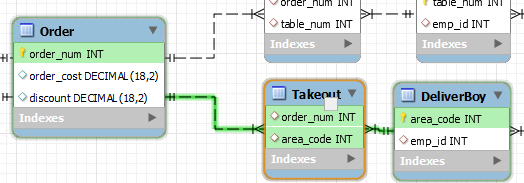


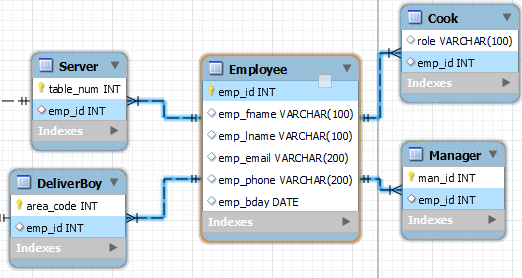
Here are some screenshots showing the relationships between each table:











Next, I will forward engineer this model into actual tables in the database. I have included the SQL statements that will complete this task:

-- 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 shield74

-- -----------------------------------------------------

-- -----------------------------------------------------

-- Schema shield74

-- -----------------------------------------------------

CREATE SCHEMA IF NOT EXISTS `shield74` DEFAULT CHARACTER SET utf8 ;

USE `shield74` ;

-- -----------------------------------------------------

-- Table `shield74`.`Customer`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `shield74`.`Customer` (

`cust\_id` INT NOT NULL AUTO\_INCREMENT,

`cust\_fname` VARCHAR(200) NULL,

`cust\_lname` VARCHAR(200) NULL,

`cust\_email` VARCHAR(200) NULL,

`cust\_record` INT NULL,

PRIMARY KEY (`cust\_id`),

UNIQUE INDEX `cst\_id\_UNIQUE` (`cust\_id` ASC) VISIBLE)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `shield74`.`Order`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `shield74`.`Order` (

`order\_num` INT NOT NULL AUTO\_INCREMENT,

`order\_cost` DECIMAL(18,2) NULL,

`discount` DECIMAL(18,2) NULL,

PRIMARY KEY (`order\_num`),

UNIQUE INDEX `order\_num\_UNIQUE` (`order\_num` ASC) VISIBLE)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `shield74`.`MenuItem`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `shield74`.`MenuItem` (

`item\_id` INT NOT NULL AUTO\_INCREMENT,

`item\_name` VARCHAR(200) NULL,

`item\_cost` DECIMAL(18,2) NULL,

PRIMARY KEY (`item\_id`),

UNIQUE INDEX `item\_id\_UNIQUE` (`item\_id` ASC) VISIBLE)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `shield74`.`Employee`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `shield74`.`Employee` (

`emp\_id` INT NOT NULL AUTO\_INCREMENT,

`emp\_fname` VARCHAR(100) NULL,

`emp\_lname` VARCHAR(100) NULL,

`emp\_email` VARCHAR(200) NULL,

`emp\_phone` VARCHAR(200) NULL,

`emp\_bday` DATE NULL,

PRIMARY KEY (`emp\_id`),

UNIQUE INDEX `emp\_id\_UNIQUE` (`emp\_id` ASC) VISIBLE)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `shield74`.`CustomerOrder`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `shield74`.`CustomerOrder` (

`cust\_id` INT NULL,

`order\_num` INT NULL,

INDEX `fk\_cust\_id\_idx` (`cust\_id` ASC) VISIBLE,

INDEX `fk\_order\_num\_idx` (`order\_num` ASC) VISIBLE,

CONSTRAINT `fk\_customer\_id\_to\_cust`

FOREIGN KEY (`cust\_id`)

REFERENCES `shield74`.`Customer` (`cust\_id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_order\_num\_to\_order`

FOREIGN KEY (`order\_num`)

REFERENCES `shield74`.`Order` (`order\_num`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `shield74`.`OrderItems`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `shield74`.`OrderItems` (

`order\_num` INT NULL,

`item\_id` INT NULL,

INDEX `fk\_order\_num\_idx` (`order\_num` ASC) VISIBLE,

INDEX `fk\_item\_id\_idx` (`item\_id` ASC) VISIBLE,

CONSTRAINT `fk\_order\_items\_`

FOREIGN KEY (`order\_num`)

REFERENCES `shield74`.`Order` (`order\_num`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_item\_id\_to\_item`

FOREIGN KEY (`item\_id`)

REFERENCES `shield74`.`MenuItem` (`item\_id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `shield74`.`Server`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `shield74`.`Server` (

`table\_num` INT NOT NULL,

`emp\_id` INT NULL,

PRIMARY KEY (`table\_num`),

UNIQUE INDEX `table\_num\_UNIQUE` (`table\_num` ASC) VISIBLE,

INDEX `fk\_server\_id\_idx` (`emp\_id` ASC) VISIBLE,

CONSTRAINT `fk\_server\_id`

FOREIGN KEY (`emp\_id`)

REFERENCES `shield74`.`Employee` (`emp\_id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `shield74`.`DeliverBoy`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `shield74`.`DeliverBoy` (

`area\_code` INT NOT NULL,

`emp\_id` INT NULL,

PRIMARY KEY (`area\_code`),

UNIQUE INDEX `area\_code\_UNIQUE` (`area\_code` ASC) VISIBLE,

INDEX `fr\_emp\_id\_idx` (`emp\_id` ASC) VISIBLE,

CONSTRAINT `fr\_delivery\_boy\_id`

FOREIGN KEY (`emp\_id`)

REFERENCES `shield74`.`Employee` (`emp\_id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `shield74`.`Cook`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `shield74`.`Cook` (

`role` VARCHAR(100) NULL,

`emp\_id` INT NULL,

INDEX `fk\_emp\_id\_idx` (`emp\_id` ASC) VISIBLE,

CONSTRAINT `fk\_cook\_id`

FOREIGN KEY (`emp\_id`)

REFERENCES `shield74`.`Employee` (`emp\_id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `shield74`.`Manager`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `shield74`.`Manager` (

`man\_id` INT NOT NULL AUTO\_INCREMENT,

`emp\_id` INT NULL,

PRIMARY KEY (`man\_id`),

UNIQUE INDEX `man\_id\_UNIQUE` (`man\_id` ASC) VISIBLE,

INDEX `fk\_emp\_id\_idx` (`emp\_id` ASC) VISIBLE,

CONSTRAINT `fk\_emp\_id\_of\_man`

FOREIGN KEY (`emp\_id`)

REFERENCES `shield74`.`Employee` (`emp\_id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `shield74`.`Takeout`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `shield74`.`Takeout` (

`order\_num` INT NULL,

`area\_code` INT NULL,

INDEX `fk\_order\_num\_idx` (`order\_num` ASC) VISIBLE,

INDEX `fk\_are\_code\_idx` (`area\_code` ASC) VISIBLE,

CONSTRAINT `fk\_order\_num\_from\_delivery`

FOREIGN KEY (`order\_num`)

REFERENCES `shield74`.`Order` (`order\_num`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_delivered\_by`

FOREIGN KEY (`area\_code`)

REFERENCES `shield74`.`DeliverBoy` (`area\_code`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `shield74`.`Dine-in`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `shield74`.`Dine-in` (

`order\_num` INT NULL,

`table\_num` INT NULL,

INDEX `fk\_order\_num\_idx` (`order\_num` ASC) VISIBLE,

INDEX `fk\_table\_num\_idx` (`table\_num` ASC) VISIBLE,

CONSTRAINT `fk\_order\_num\_from\_Dine`

FOREIGN KEY (`order\_num`)

REFERENCES `shield74`.`Order` (`order\_num`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_served\_by`

FOREIGN KEY (`table\_num`)

REFERENCES `shield74`.`Server` (`table\_num`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

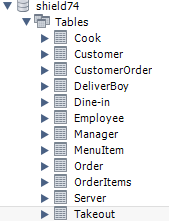
ENGINE = InnoDB;

SET SQL\_MODE=@OLD\_SQL\_MODE;

SET FOREIGN\_KEY\_CHECKS=@OLD\_FOREIGN\_KEY\_CHECKS;

SET UNIQUE\_CHECKS=@OLD\_UNIQUE\_CHECKS;

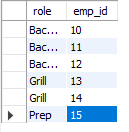
Now that the Forward engineer is done I will include a screenshot of the tables in the database:



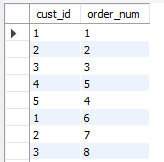
Next, I initialized each table to have at least 5 rows.

I have included pictures o these below:

Cook:



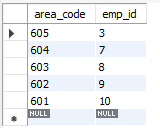
CustomerOrder:



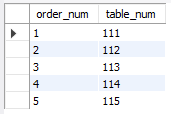
Customer:



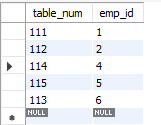
DeliverBoy:



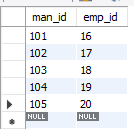
DineIn:



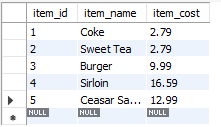
Server:



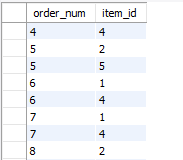
Manager:



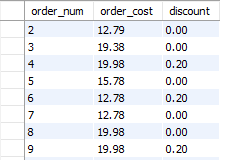
MenuItem:



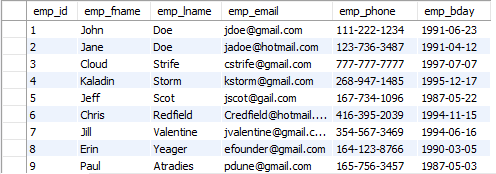
OrderItems:



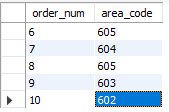
Order:



Employee:



TakeOut:



I also included screenshots of a few of my Inserts:





Finally, I will perform a join query to test the data in my table. This query displays the Info for each employee who is a server. A screenshot of the query along with the result is shown below:

