**Yummy+**

Guanxiaoxiong Liu

<https://github.com/lgxxabc/50900_Project.git>

3/26/2022

**Revision History**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Date** | **Description** | **Author** | **Reviewer** | **Version** |
| 2022/03/18 | Initial release of this document | G.Liu |  | V1.0.0 |

**Table of Contents**

[1. Introduction 4](#_Toc101792030)

[**1.1** **Overview** 4](#_Toc101792031)

[2. Database design 5](#_Toc101792032)

[**2.1** **Relational Database Design** 5](#_Toc101792033)

[**2.2** **Entity Relationship Diagram (ERD) Model** 6](#_Toc101792034)

[3. Database Implementation 8](#_Toc101792035)

[**3.1** **Create Database Tables** 8](#_Toc101792038)

[**3.2** **Load Data Files** 10](#_Toc101792039)

[**3.3** **Define PKs and FKs** 12](#_Toc101792040)

[4. Database Queries Practices 14](#_Toc101792041)

[**4.1** **Create twelve Queries** 14](#_Toc101792045)

[5. Practices on Indexes and Views 17](#_Toc101792046)

[**5.1** **Practices on Indexes** 17](#_Toc101792048)

[**5.2** **Practices on Views** 18](#_Toc101792049)

# Introduction

* 1. **Overview**

I have created an online food ordering platform called Yummy+. I need to create a database solution that keeps track of customer data, online shopping cart data, restaurant data, menu item data, and order item data. I’m interested in this data because I need to know who (customer info) is using which cart (cart ID) and which dishes have been ordered (order items) from which restaurant's menu (menu items) (restaurant info). I will manually enter restaurant data and menu item data according to the information provided by the restaurants. While customer data, shopping cart data, and order item data are generated when a customer produces an order, as the business owner, I need to see all the data to understand how my business is doing. The sales team needs to view customer data and order item data to provide the right deals and incentives to encourage customers to use our online ordering platform. They need to look at customer data and order item data to understand which foods are most popular at which restaurants and which are ordered in smaller quantities. Armed with this information, we can appropriately adjust the selection of restaurants and food on the platform or change the price of food to stimulate sales. Website administrators need to understand the behavior of users visiting the website based on customer data to provide support to better meet the needs of users. For example, we can count which users have ordered the most meals from which restaurant on our website, which may indicate that they like this restaurant the most. Then we can recommend new menus to users in a more targeted manner to improve their user experience. In addition, back-end engineers need user data to count which day of the week the website has the most traffic and the highest number of concurrent users, so they can optimize the performance of the website during that period, such as adding servers, doing load balancing, doing caching, and so on. Front-end engineers need restaurant data and menu item data to analyze which web pages are visited the most to determine which pages need to be optimized.

# Database design

* 1. **Relational Database Design**

For my business, I will have the following entities:

1. Customer Info – describes the Email, First Name, Last Name, Password, and Cart ID(FK)
2. Cart Info – describes Cart ID, Date, and Total Price
3. Order Item Info – describes Order Item ID, Quantity, Price, Menu Item ID(FK), and Cart ID(FK)
4. Menu Item Info – describes Menu Item ID, Name, Price, Item’s description, and Restaurant ID(FK)
5. Restaurant Info – describes Restaurant ID, Address, Name, Phone# and Image URL

|  |  |
| --- | --- |
| **Customer Info** | |
| Email (PK) | Varchar (30) |
| First Name | Varchar (20) |
| Last Name | Varchar (20) |
| Password | Varchar (20) |
| Cart ID(FK) | Integer |

|  |  |
| --- | --- |
| **Cart Info** | |
| Cart ID(PK) | Integer |
| Date | Date |
| Total Price | Float (7, 2) |

|  |  |
| --- | --- |
| **Order Item Info** | |
| ID(PK) | Integer |
| Quantity | Integer |
| Price | Float (7, 2) |
| Menu Item ID(FK) | Integer |
| Cart ID(FK) | Integer |

|  |  |
| --- | --- |
| **Menu Item Info** | |
| ID(PK) | Integer |
| Name | Varchar (40) |
| Price | Float (7, 2) |
| Description | Blob |
| Restaurant ID(FK) | Integer |

|  |  |
| --- | --- |
| **Restaurant Info** | |
| ID(PK) | Integer |
| Address | Varchar (60) |
| Name | Varchar (40) |
| Phone | Varchar (20) |
| Image URL | Blob |

* 1. **Entity Relationship Diagram (ERD) Model**

ERD Diagram using Unified Modeling Language (UML):

Diagram

Description automatically generated

ERD Diagram using Crow’s Foot notation:

Diagram

Description automatically generated

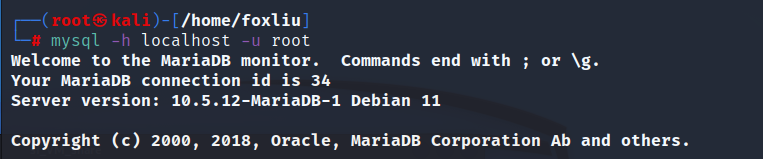
# Database Implementation

2. 1. **Create Database Tables**

To create the Database Tables, the following steps should be executed:

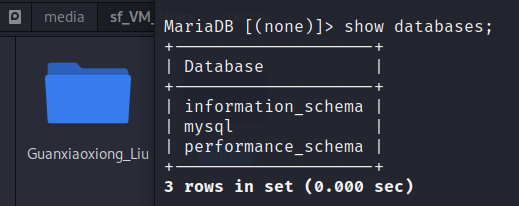
1. Connect to the MySQL server:  
   # mysql -h localhost -u root

// -h: host

// -u: username

1. Check the tables in the database:

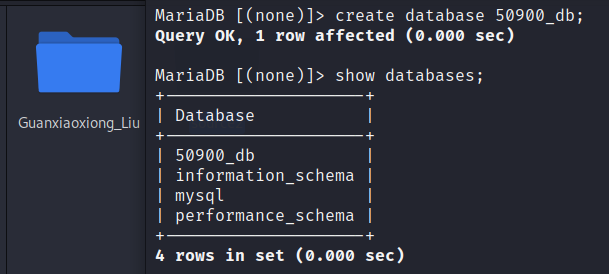
> show databases

// To show the contents in a database

1. Create a database:

> create database 50900\_db;

// To create a database named “50900\_db”

> show databases

1. Choose the database that has been created:

> use 50900\_db;

// Enter the database 50900\_db

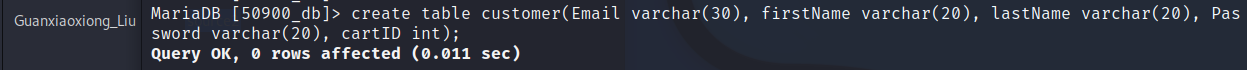
1. Create the first table:

> create table cart(ID int, Date date, Price float);

// Create a table named “cart” with three columns: ID, Date and Price.

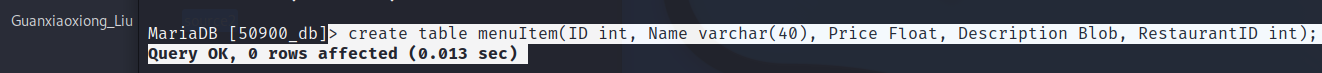
1. Create the second table:

> create table customer(Email varchar(30), firstName varchar(20), lastName varchar(20), Password varchar(20), cartID int);

// Create a table named “customer” with five columns: Email, firstName, lastName, Password and cartID.

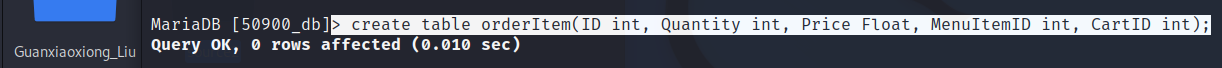
1. Create the third table:

> create table menuItem(ID int, Name varchar(40), Price Float, Description Blob, RestaurantID int);

// Create a table named “menuItem” with five columns: ID, Name, Price, Description and RestaurantID.

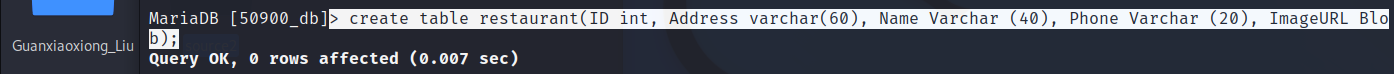
1. Create the fourth table:

> create table orderItem(ID int, Quantity int, Price Float, MenuItemID int, CartID int);

// Create a table named “orderItem” with five columns: ID, Name, Price, Description and RestaurantID.

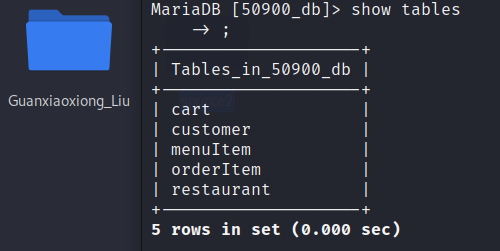
1. Create the fifth table:

> create table restaurant(ID int, Address varchar(60), Name Varchar (40), Phone Varchar (20), ImageURL Blob);

// Create a table named “restaurant” with five columns: ID, Address, Name, Phone and ImageURL.

1. Check the tables in the database:

> show tables;

 // Show the tables in the current database.

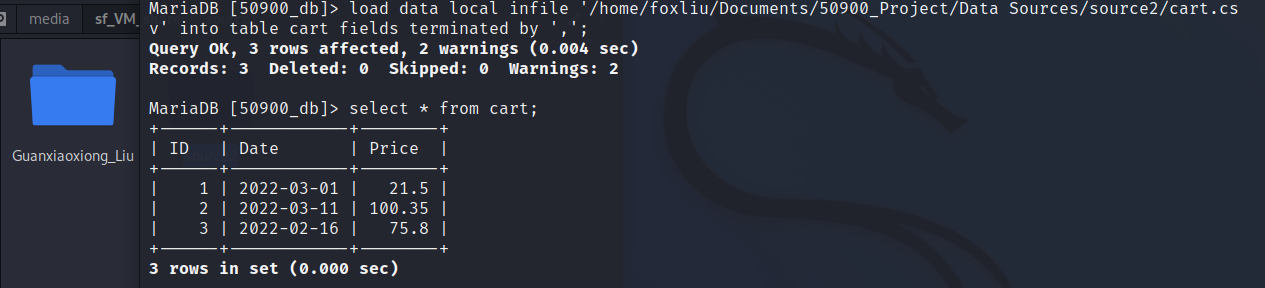
* 1. **Load Data Files**

Next, load the data files into the corresponding tables:

1. Load the first data file:  
   > load data local infile '/home/foxliu/Documents/50900\_Project/Data Sources/source2/cart.csv' into table cart fields terminated by ',';

// Load the data from cart.csv into the cart table. Include the absolute path to the data file.

> select \* from cart;

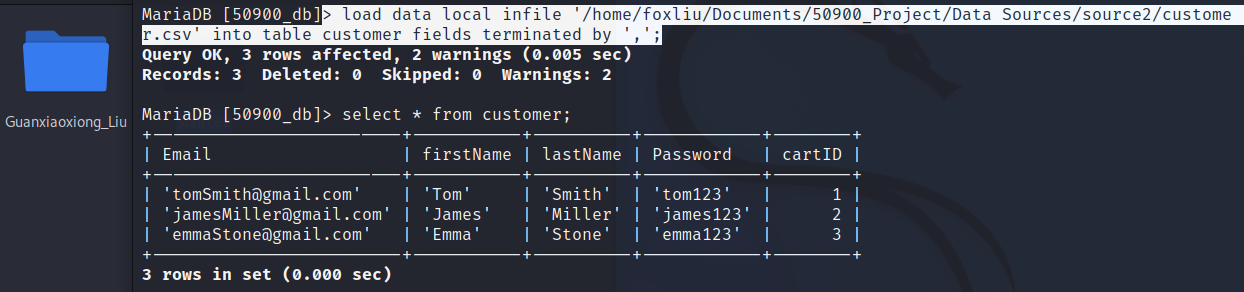
// Check the data in the table

1. Load the second data file:

> load data local infile '/home/foxliu/Documents/50900\_Project/Data Sources/source2/customer.csv' into table customer fields terminated by ',';

// Load the data from customer.csv into the customer table. Include the absolute path to the data file.

> select \* from customer;

// Check the data in the table

1. Load the third data file:

> load data local infile '/home/foxliu/Documents/50900\_Project/Data Sources/source2/menuItem.csv' into table menuItem fields terminated by ',';

// Load the data from menuItem.csv into the menuItem table. Include the absolute path to the data file.

> select \* from menuItem;

// Check the data in the table

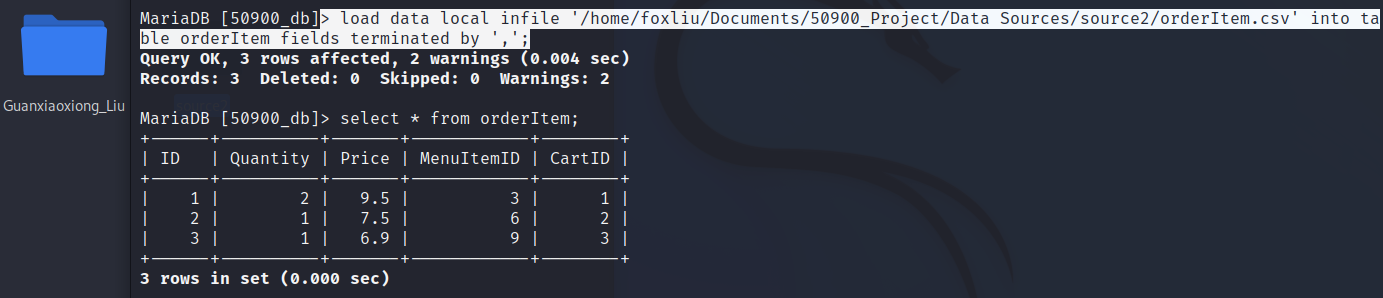
1. Load the fourth data file:

> load data local infile '/home/foxliu/Documents/50900\_Project/Data Sources/source2/orderItem.csv' into table orderItem fields terminated by ',';

// Load the data from orderItem.csv into the orderItem table. Include the absolute path to the data file.

> select \* from orderItem;

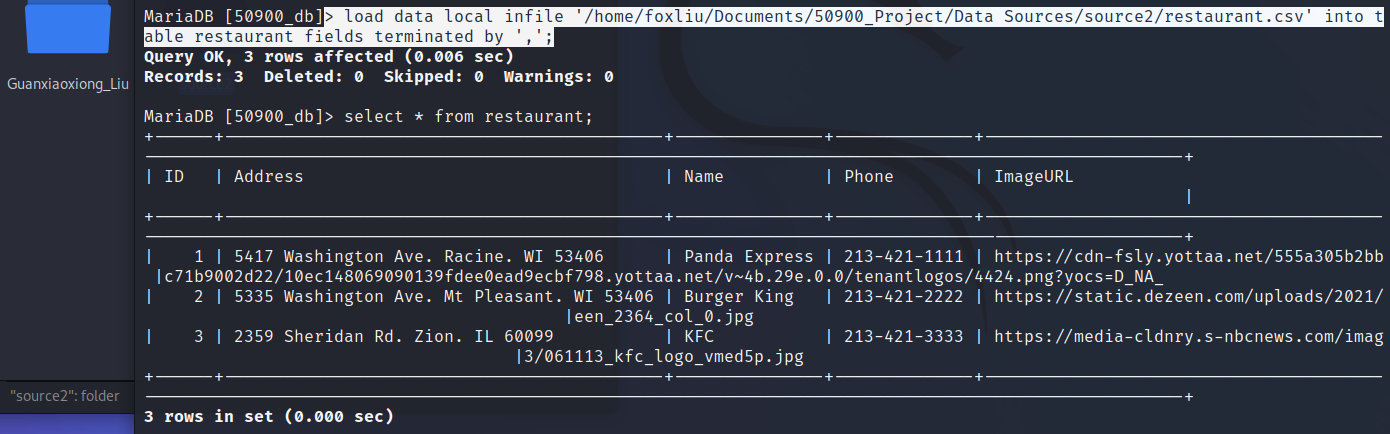
// Check the data in the table

1. Load the fifth data file:

> load data local infile '/home/foxliu/Documents/50900\_Project/Data Sources/source2/restaurant.csv' into table restaurant fields terminated by ',';

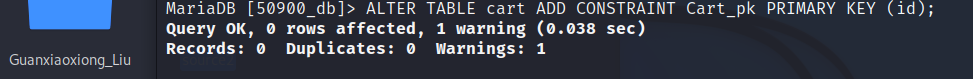
// // Load the data from restaurant.csv into the restaurant table. Include the absolute path to the data file.

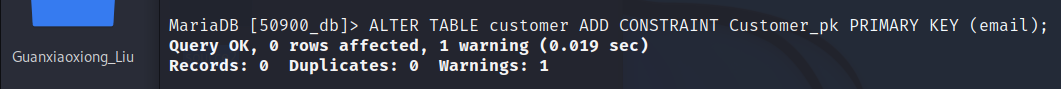
> select \* from restaurant;

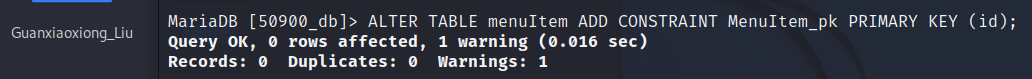
// Check the data in the table

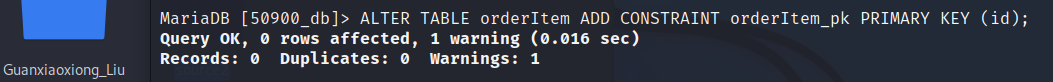
* 1. **Define PKs and FKs**

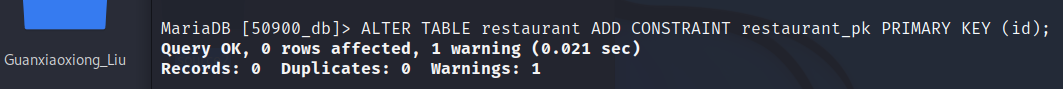
Define the Primary Keys and Foreign Keys in each table as needed:

1. Add PKs in each table:  
   > ALTER TABLE cart ADD CONSTRAINT Cart\_pk PRIMARY KEY (id);

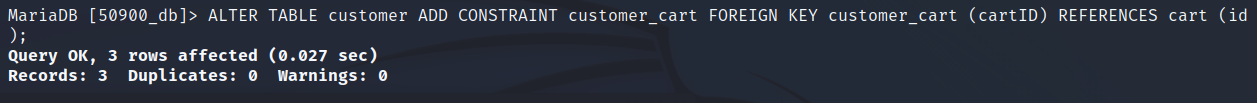
> ALTER TABLE customer ADD CONSTRAINT Customer\_pk PRIMARY KEY (email);

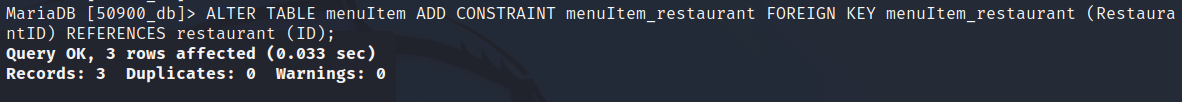
> ALTER TABLE menuItem ADD CONSTRAINT MenuItem\_pk PRIMARY KEY (id);

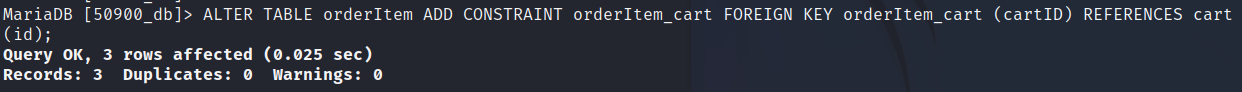
> ALTER TABLE orderItem ADD CONSTRAINT orderItem\_pk PRIMARY KEY (id);

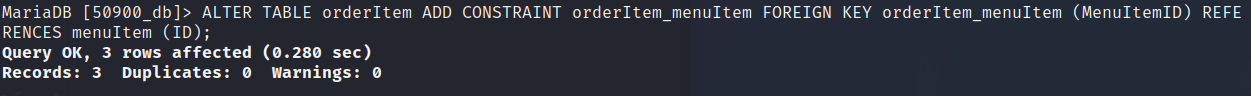
> ALTER TABLE restaurant ADD CONSTRAINT restaurant\_pk PRIMARY KEY (id);

1. Add FKs in each table:

> ALTER TABLE customer ADD CONSTRAINT customer\_cart FOREIGN KEY customer\_cart (cartID) REFERENCES cart (id);

> ALTER TABLE menuItem ADD CONSTRAINT menuItem\_restaurant FOREIGN KEY menuItem\_restaurant (RestaurantID) REFERENCES restaurant (ID);

> ALTER TABLE orderItem ADD CONSTRAINT orderItem\_cart FOREIGN KEY orderItem\_cart (cartID) REFERENCES cart (id);

> ALTER TABLE orderItem ADD CONSTRAINT orderItem\_menuItem FOREIGN KEY orderItem\_menuItem (MenuItemID) REFERENCES menuItem (ID);

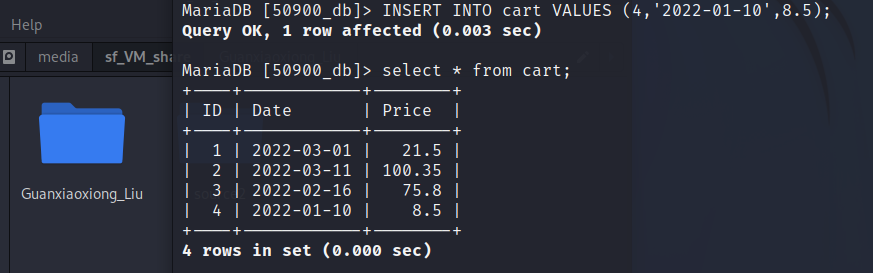
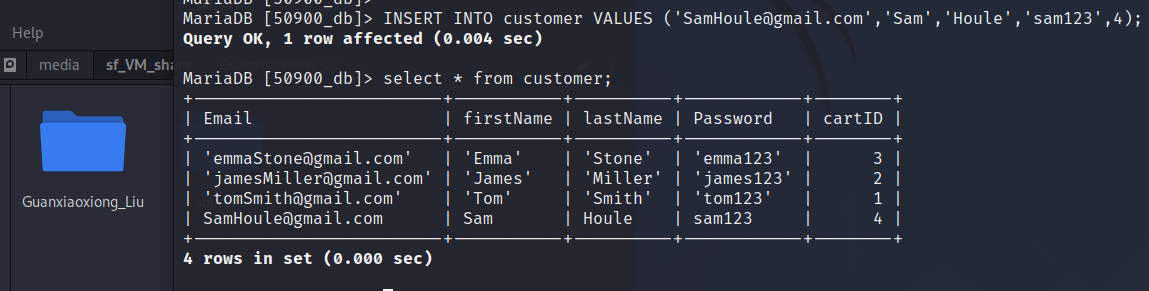
# Database Queries Practices

3. 1. **Create twelve Queries**

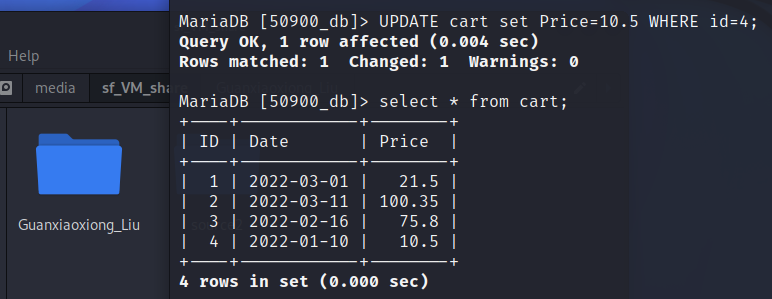
Write twelve queries using SQL commands:

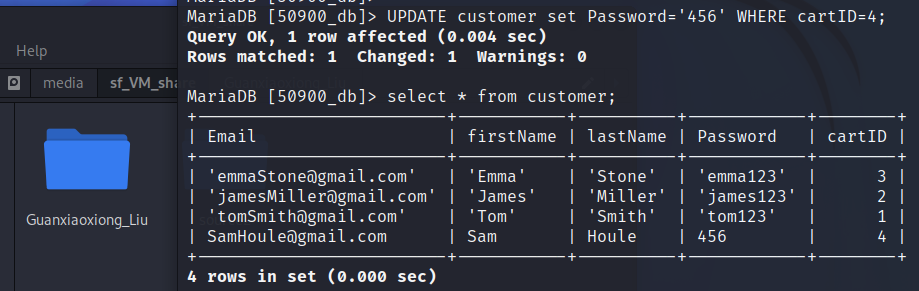
1. Two queries of Insert statements.

> INSERT INTO cart VALUES (4,'2022-01-10',8.5);

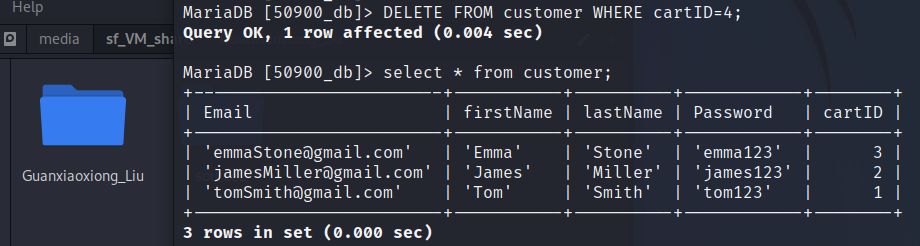
> INSERT INTO customer VALUES ('SamHoule@gmail.com','Sam','Houle','sam123',4);

1. Two queries of Update statements.

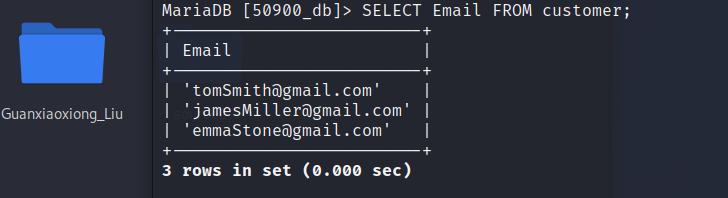
> UPDATE cart set Price=10.5 WHERE id=4;

> UPDATE customer set Password='456' WHERE cartID=4;

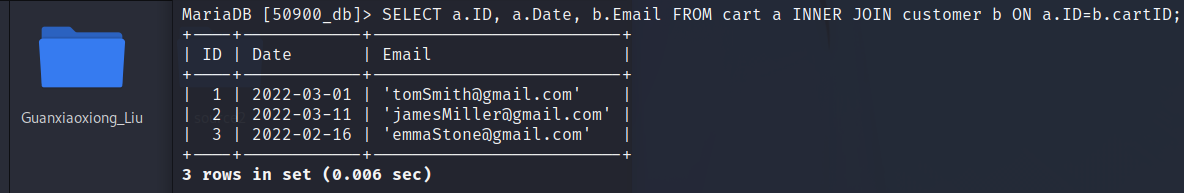
1. One query of Delete statement.

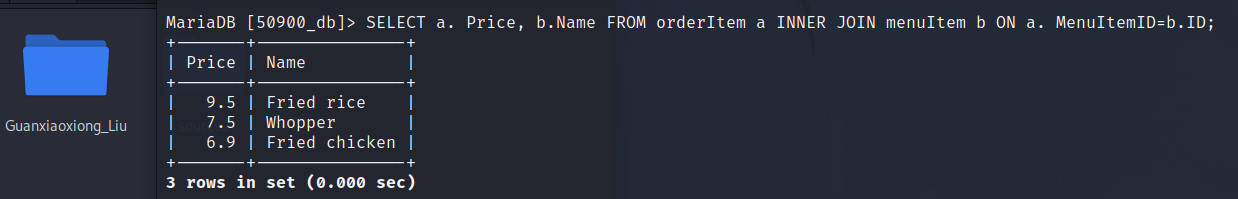
> DELETE FROM customer WHERE cartID=4;

1. One Select statement that selects a subset of the rows and columns from a table.

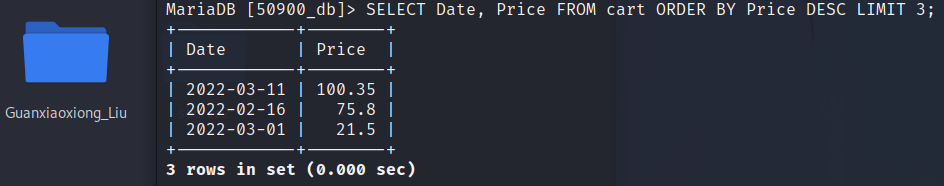
> SELECT Email FROM customer;

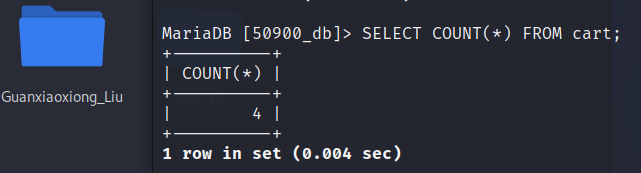
1. Two queries of Select statements that select data from a joining of two tables.

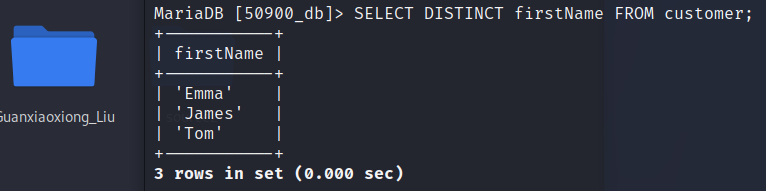
> SELECT a.ID, a.Date, b.Email FROM cart a INNER JOIN customer b ON a.ID=b.cartID;

> SELECT a. Price, b.Name FROM orderItem a INNER JOIN menuItem b ON a. MenuItemID=b.ID;

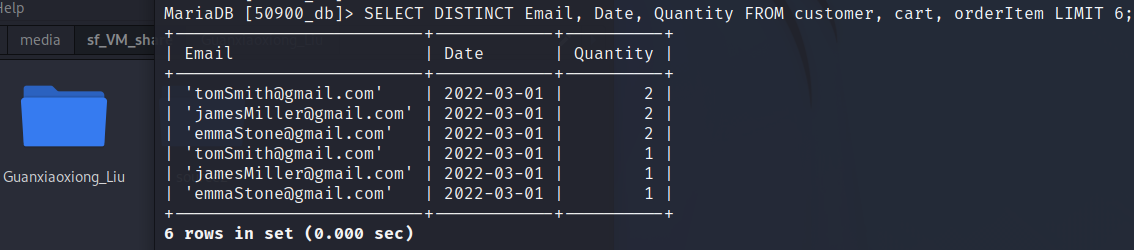
1. Three queries to use summary functions to generate statistics about the data.

> SELECT Date, Price FROM cart ORDER BY Price DESC LIMIT 3;

> SELECT COUNT(\*) FROM cart;

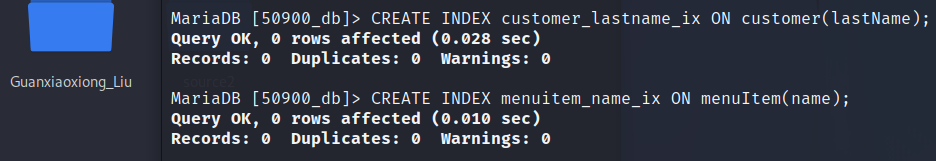
> SELECT DISTINCT firstName FROM customer;

1. One multi-table query.

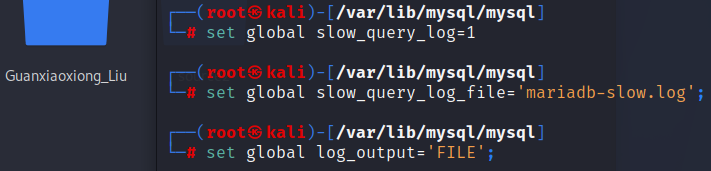
> SELECT DISTINCT Email, Date, Quantity FROM customer, cart, orderItem LIMIT 6;

# Practices on Indexes and Views

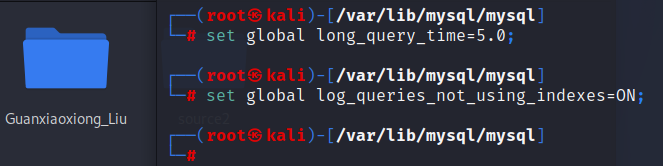
1. 1. **Practices on Indexes**

I added an index to the menuItem table. This is the table that database administrators use when managing restaurant databases or searching for items from a restaurant. An index can help administrators find the item faster, especially when a restaurant's menu offers a wide variety of foods. Likewise, I added another index to the Customer table. This table can be used when the sales team is trying to find out information about a customer. Given that the number of customers may be large, indexing will improve the efficiency of searching for information in the database.

Activity Monitor and Query Store are tools to identity slow queries in SQL server. While the Slow Query Log is used to record the SQL queries that took a long time to perform in the MariaDB in Kali Linux. And mysqldumpslow is a tool to examine the slow query log.

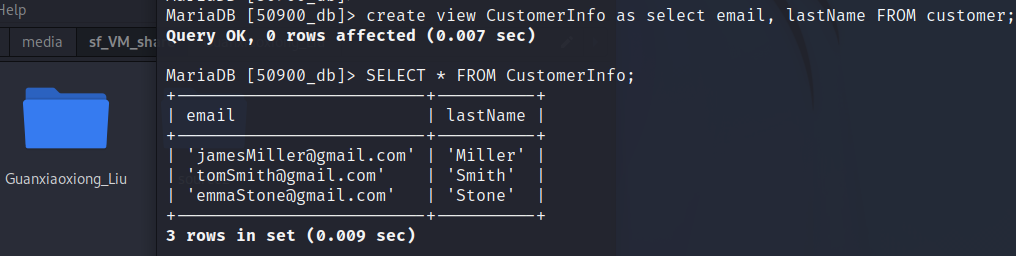
The slow query log is disabled by default. To enable the slow query log, set slow\_query\_log system variable to 1. Then configure the Slow Query Log filename to ‘mariadb-slow.log’. The slow query log can either be written to a file or a table. It can be explicitly chosen by setting the log\_output system variable.

The default slow query log time is 10 seconds. The time can be configured by setting the long\_query\_time system variable. In addition, the slow query log can be configured to log queries that do not use indexes regardless of their execution time by setting the log\_queries\_not\_using\_indexes system variable.

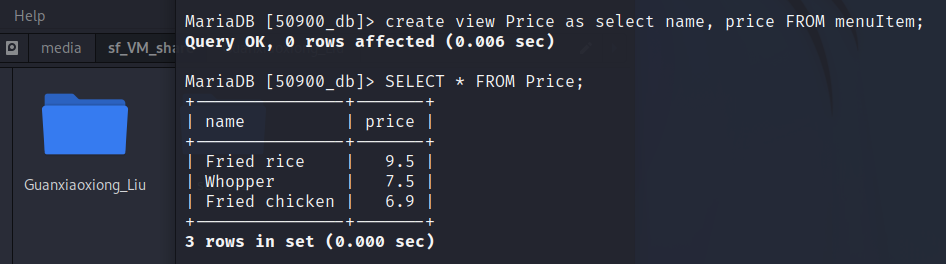
Currently, the design is not big enough to show improvements with an index, and the queries are not slow enough to be captured by the slow query logs.

* 1. **Practices on Views**

A view is a virtual table which contents are defined by a query. Like a real table, a view contains a series of named rows and columns of data, but a view is not a table that the database actually stores.

I created a view named CustomerInfo.

This view is valuable addition to my database because it hides sensitive information such as the passwords of the customer. When the restaurant requires us to provide user information, we can provide these data after desensitization. The only information the restaurant can see is the user's last name and their email address.

Likewise, I created another view named Price.

The value of this view is that it eliminates redundant information. Sometimes customers don't care about the raw materials and ingredients of the food. In this case, we only need to provide the name and price of the food to the customer. So that customers can quickly understand the price of the food and make choices.