Code Documentation for Restaurant Chain Database System

Table of Contents

List of Figures

1 Environment setup

We need XAMPP Control Panel for our process. This implementation made on Apporto platform.

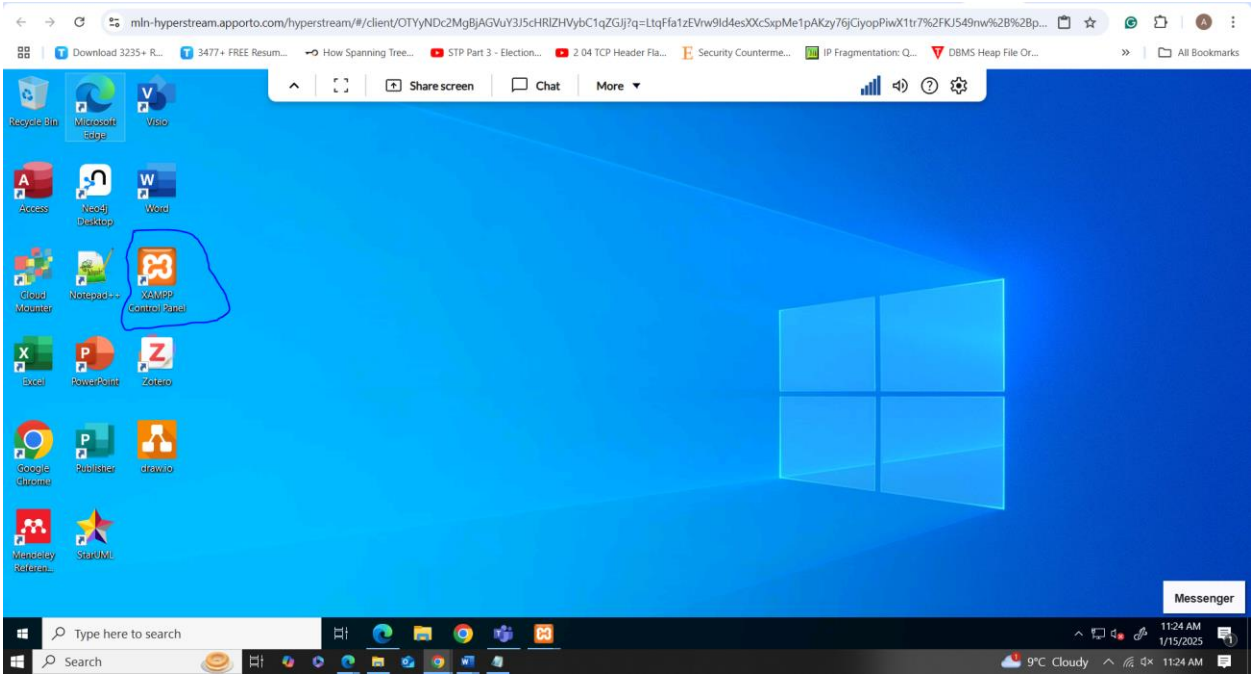


Figure 1: XAMPP Control Panel application

Inside that Apache and MySQL need to be start first.

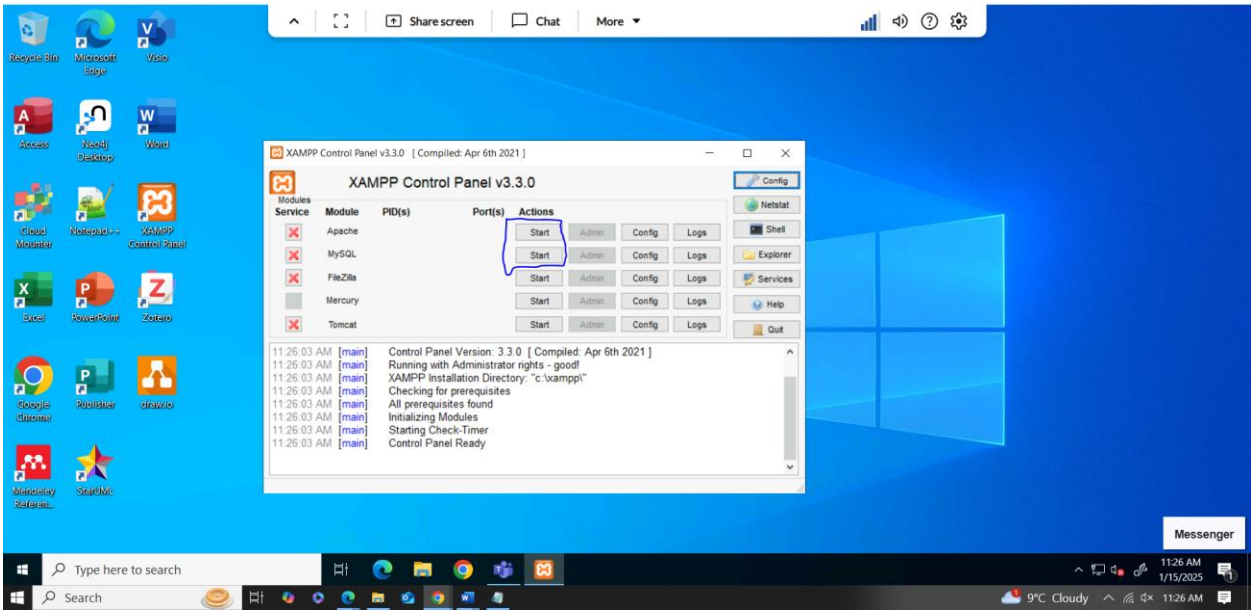


Figure 2: Apache and MySQL starting position

After the start, we can see Apache and MySQL were prompt green.

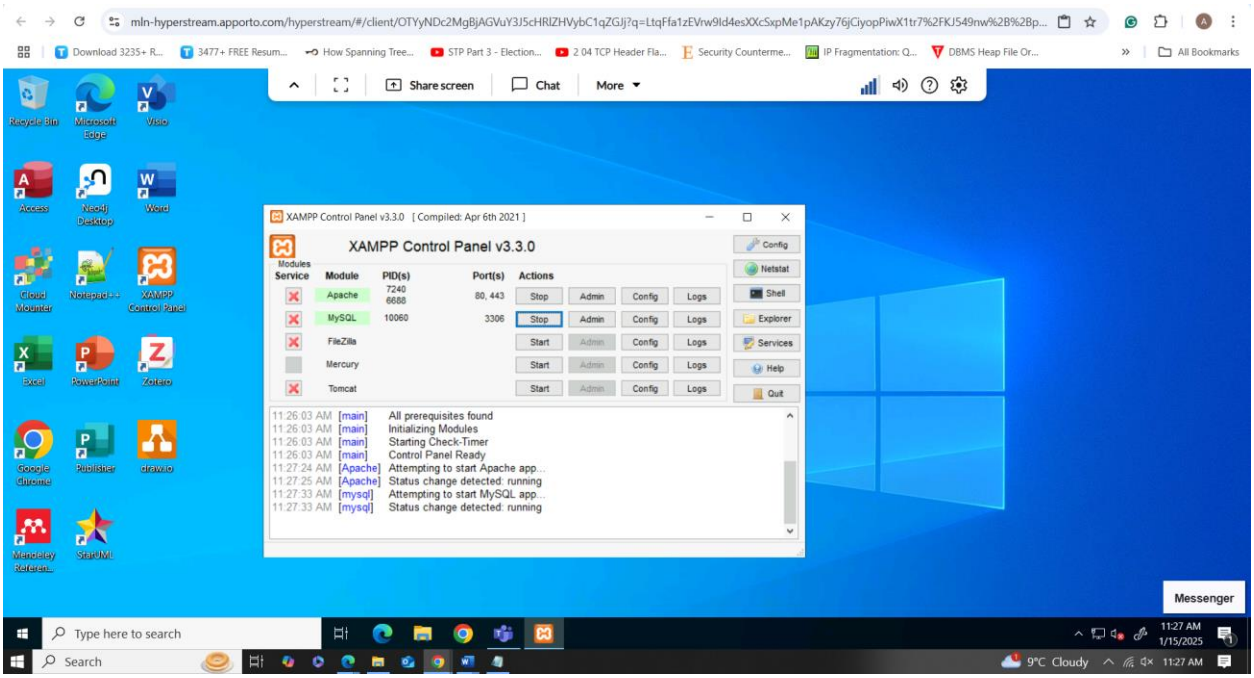


Figure 3: Initial start of apache and MySQL

After we need to click MySQL admin button to see the GUI of the phpMyAdmin.

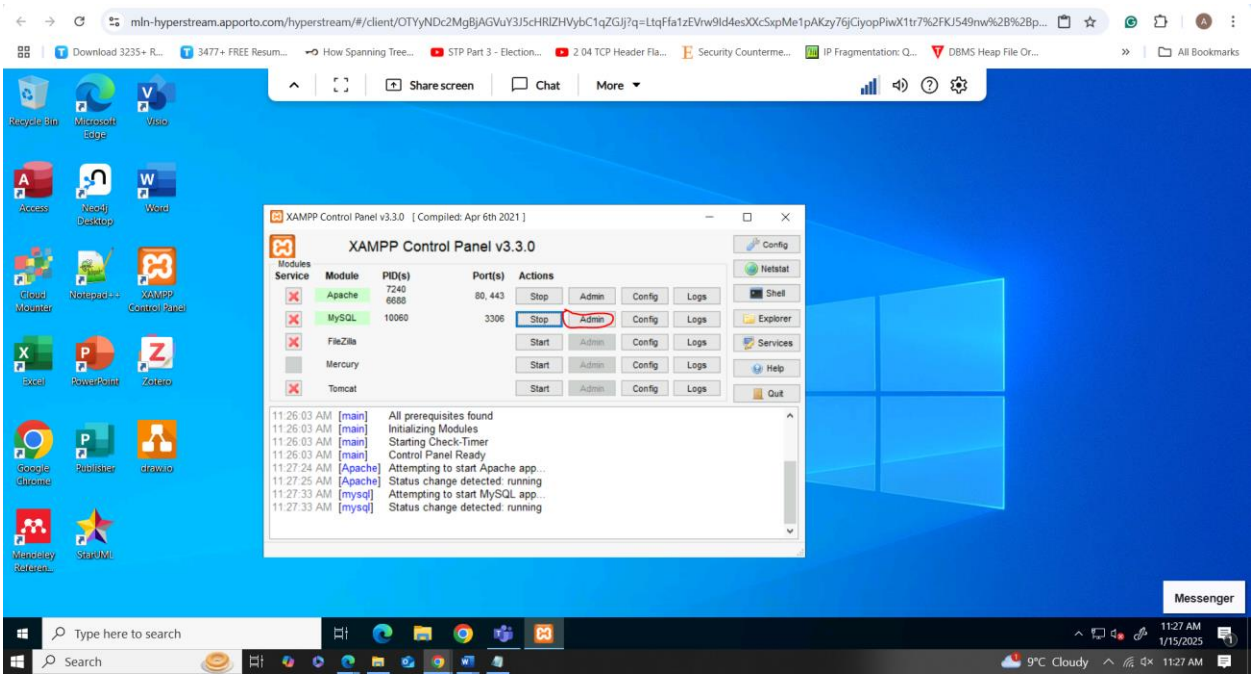


Figure 4: MYSQL admin button

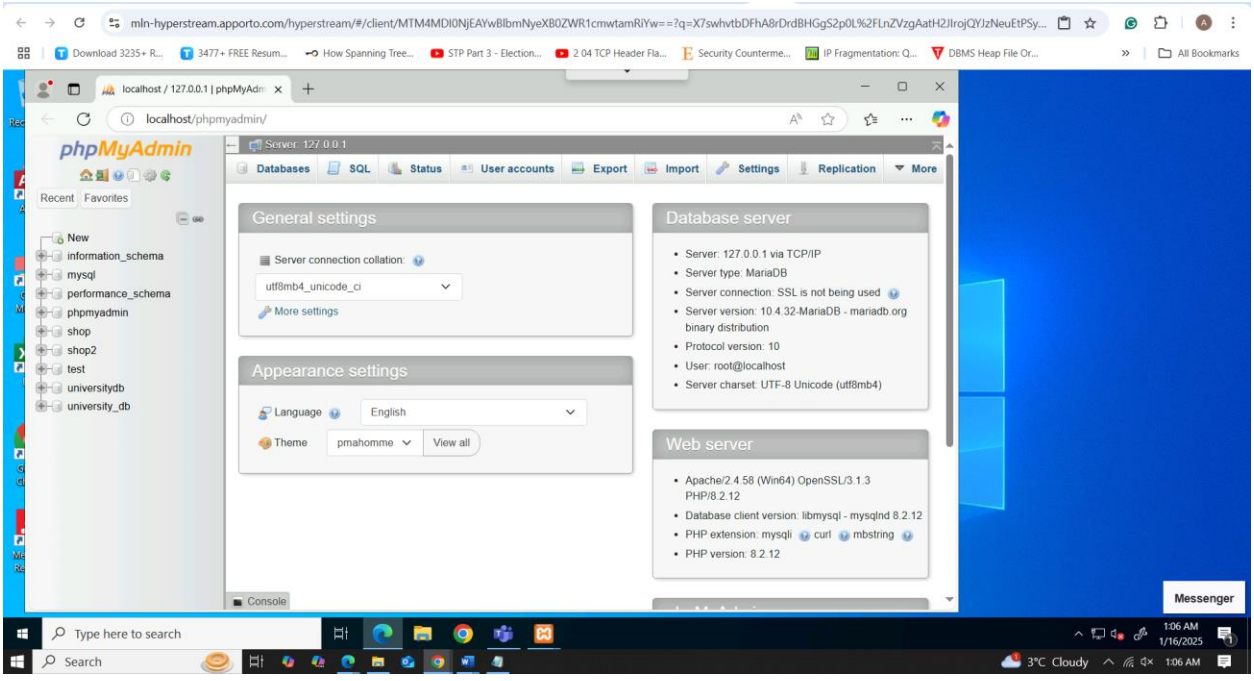


Figure 5: phpMyAdmin localhost interface

We can use shell also for entire process. For better view and understanding the queries, we used Admin interface (GUI).

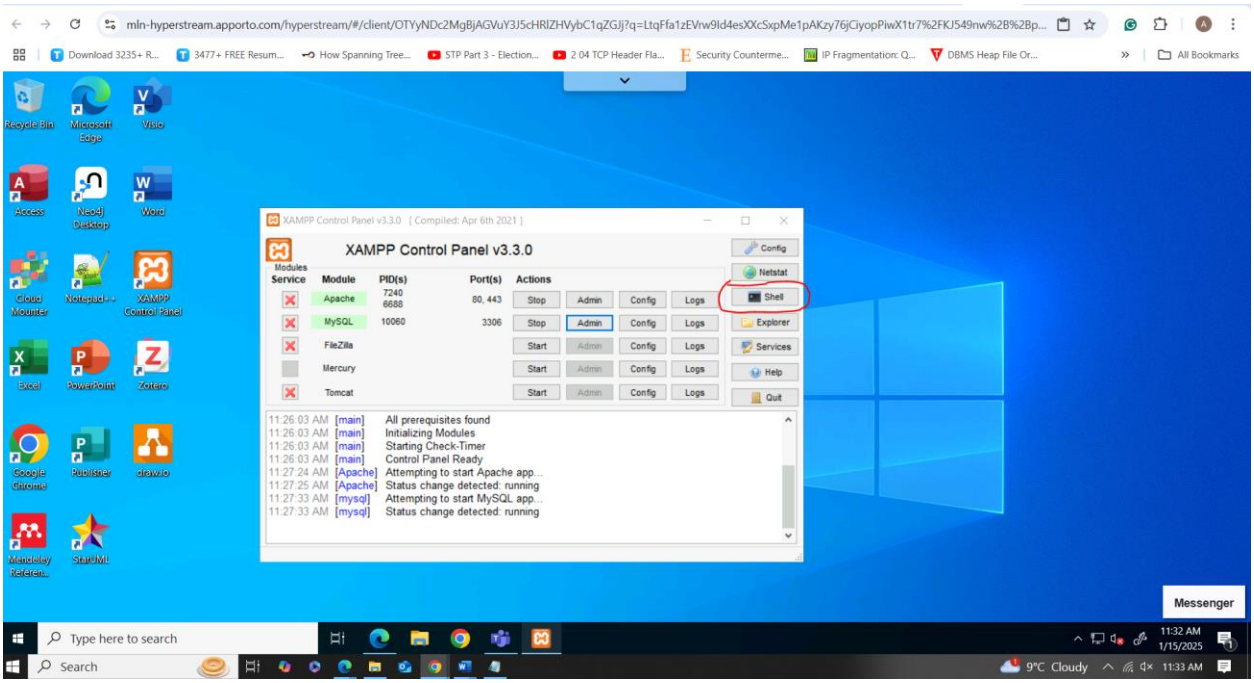


Figure 6: shell button for Command Line Interface

2 Create Database

First, we need to create database.

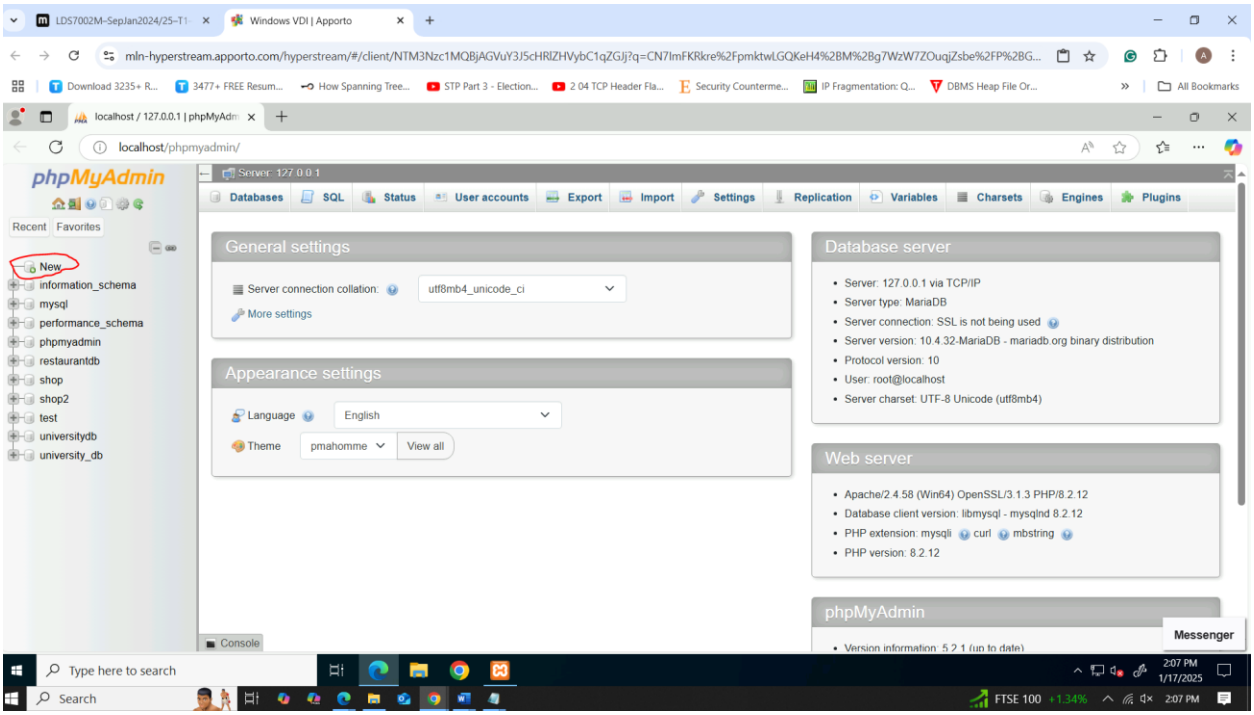


Figure 7 : new button for creating database

We named it Restaurant_Chain_DB.

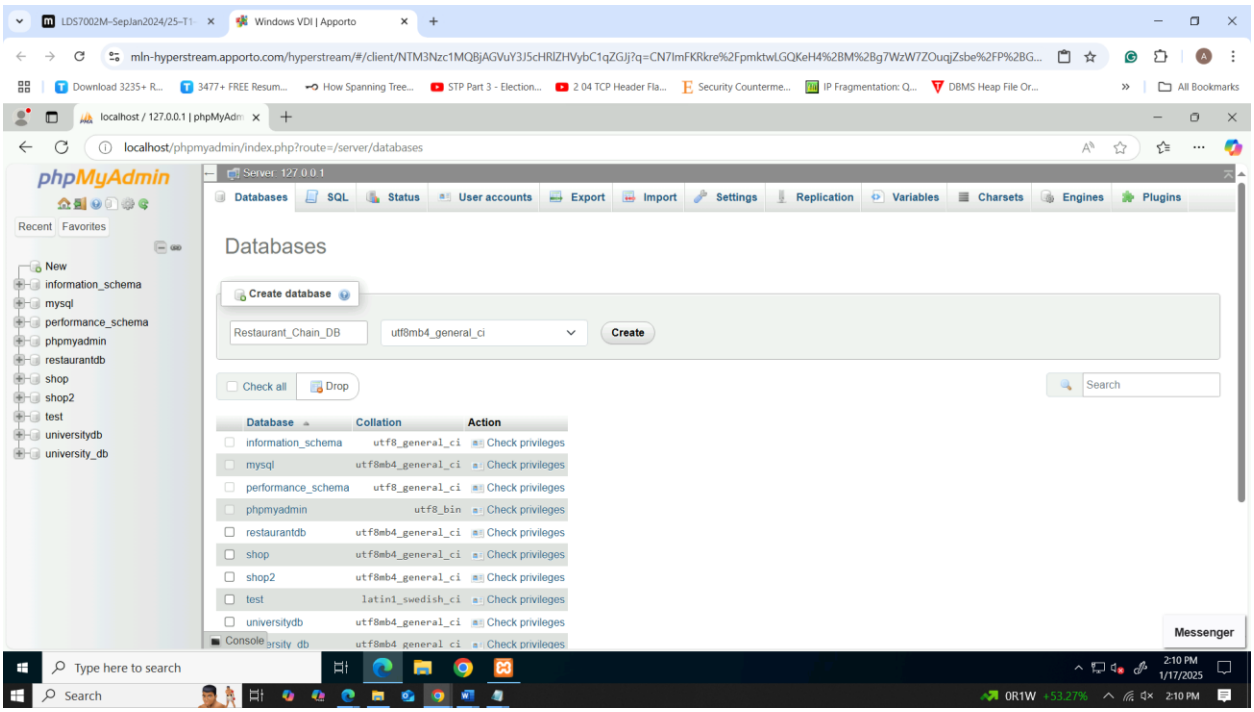


Figure 8 : restaurant_chain_db database

In left side we can see restaurant_chain_db database.

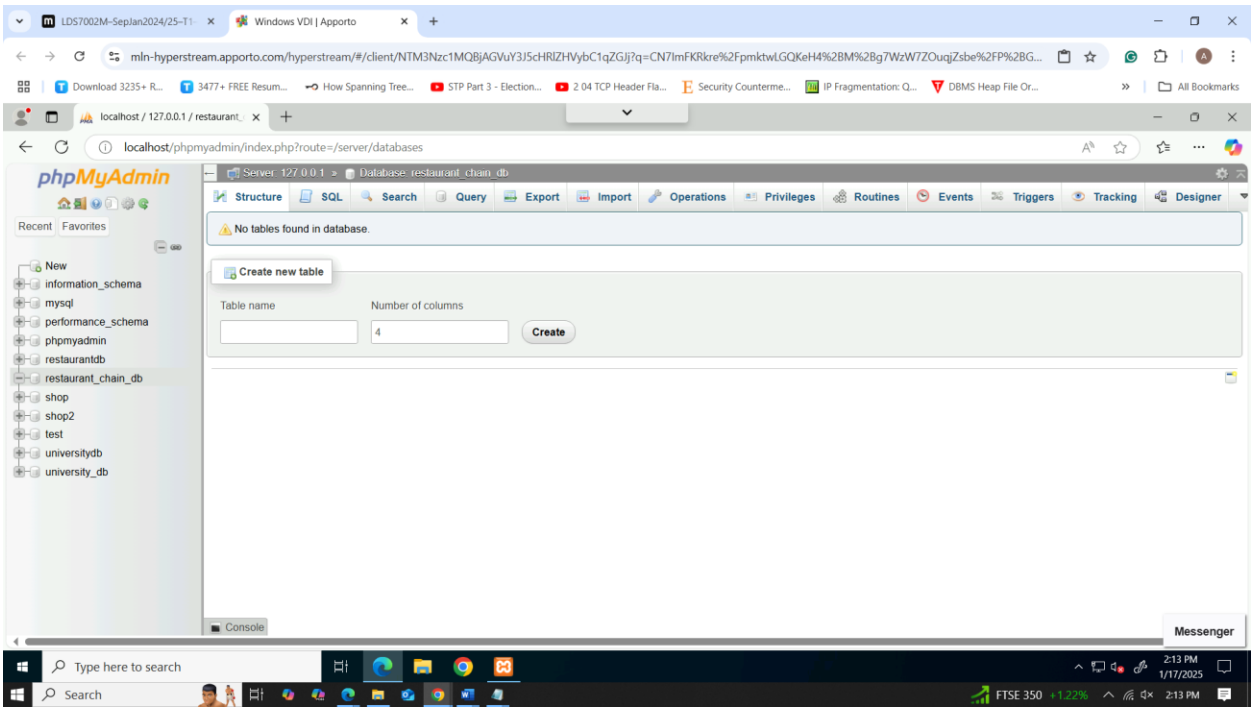


Figure 9: created the restaurant_chain_db database

3 Create Tables

- Menu_item

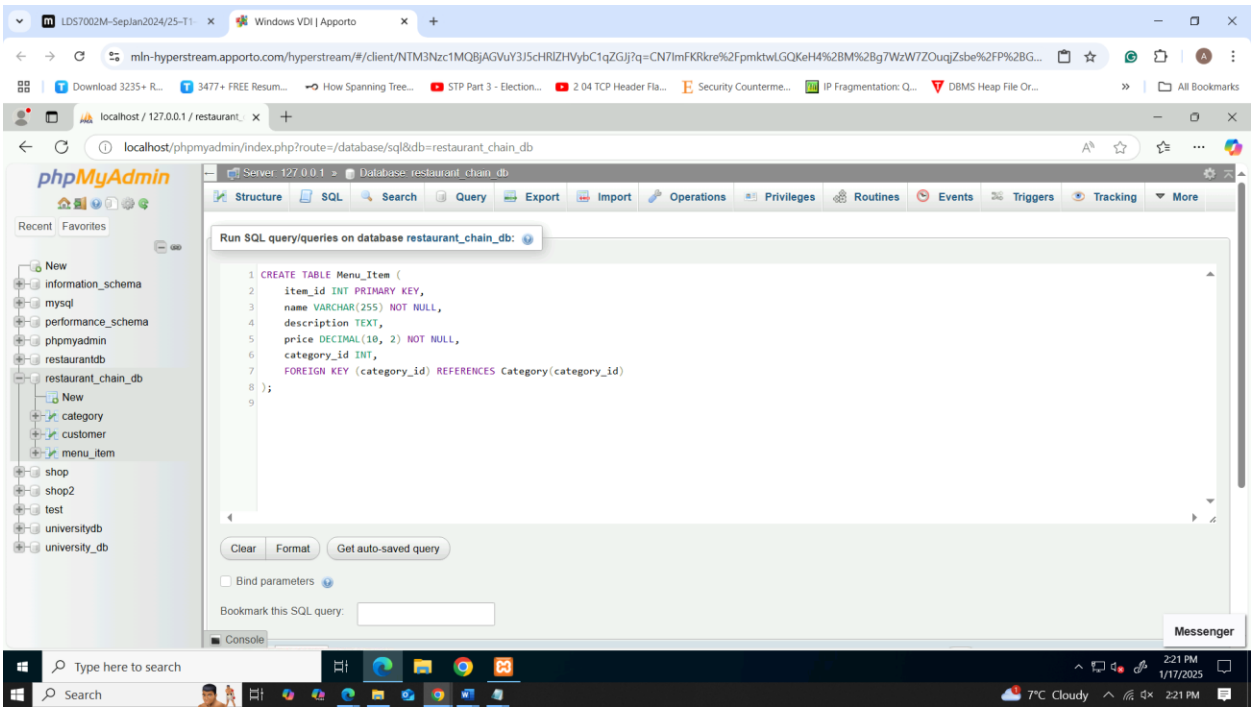


Figure 10: create table query of Menu_item

This SQL statement creates a Menu_Item table with columns: item_id (auto-incremental integer unique key), name (string that cannot be empty and has the length of 255 characters maximum), description (text intended for the display of detailed descriptions), price (non-null decimal of the 10S scale, 2D precision), and category_id (link to a category). The FOREIGN KEY (category_id) REFERENCES Category(category_id) guaranties that the category_id in the Menu_Item table matches the existing one in the Category table. Make certain that a table named Category already exists and it creates a column known as category_id upon which the key relationship is based.

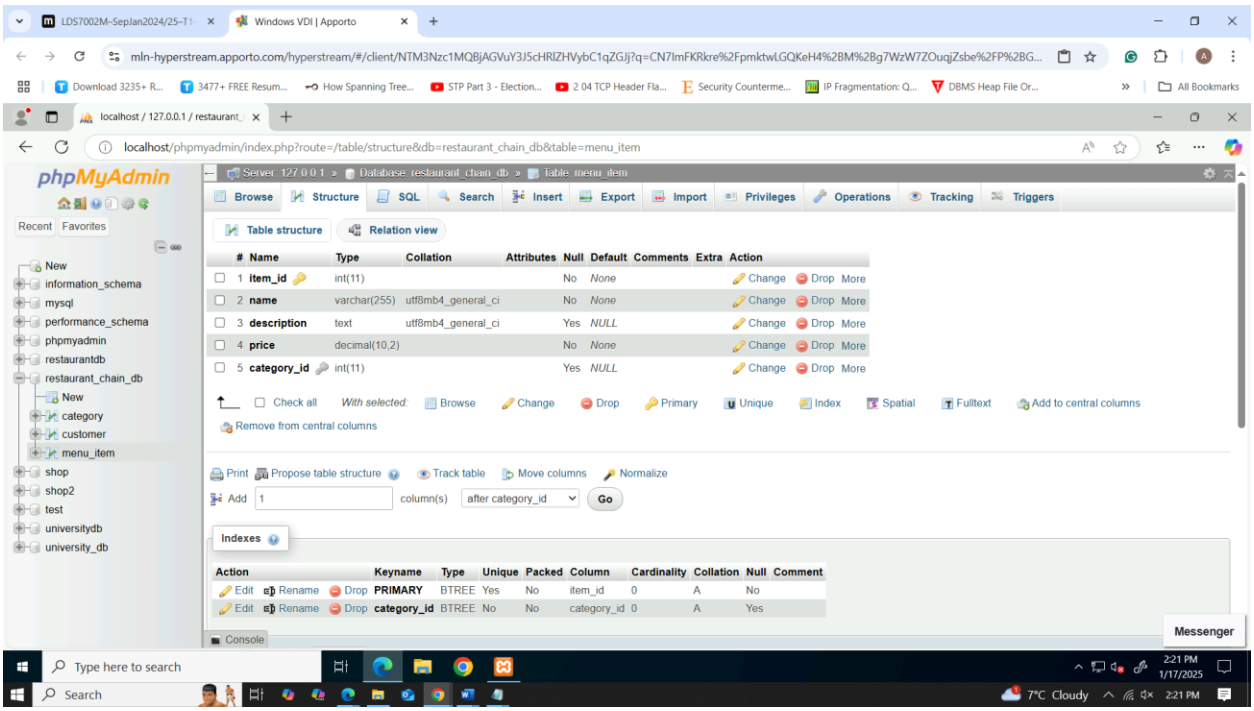


Figure 11: structure of Menu_item table

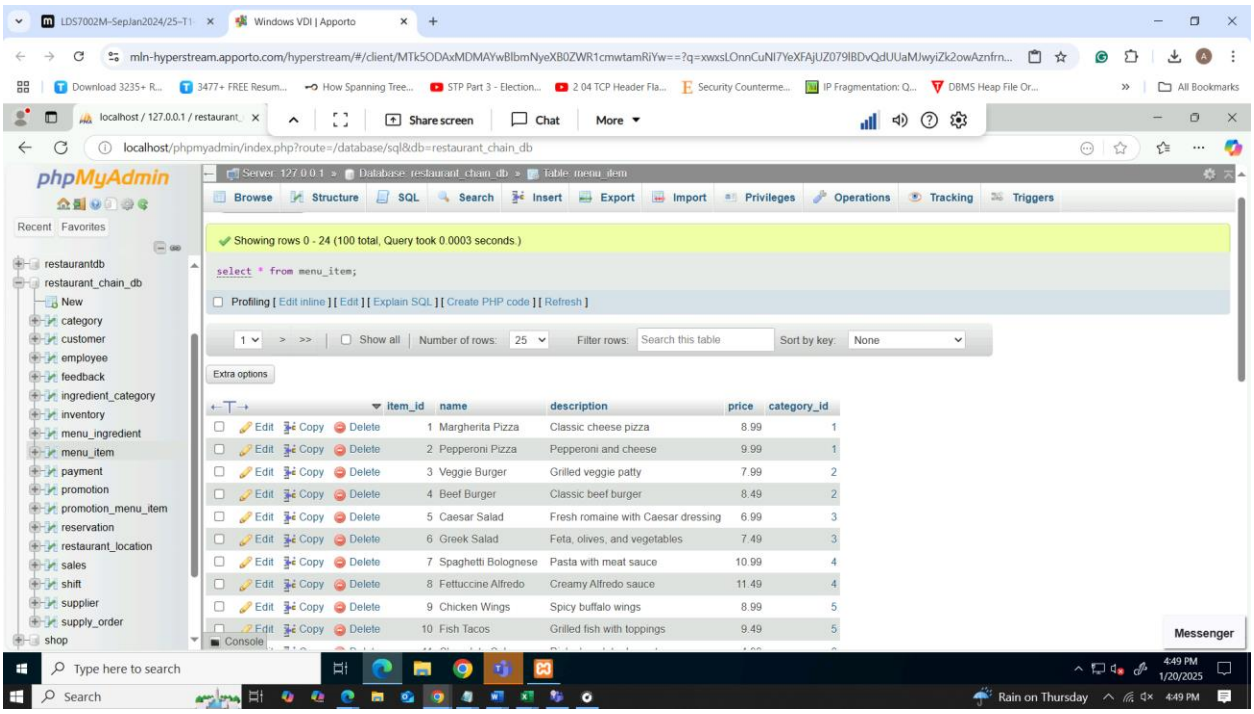


Figure 12 : select query for menu_item

- Category

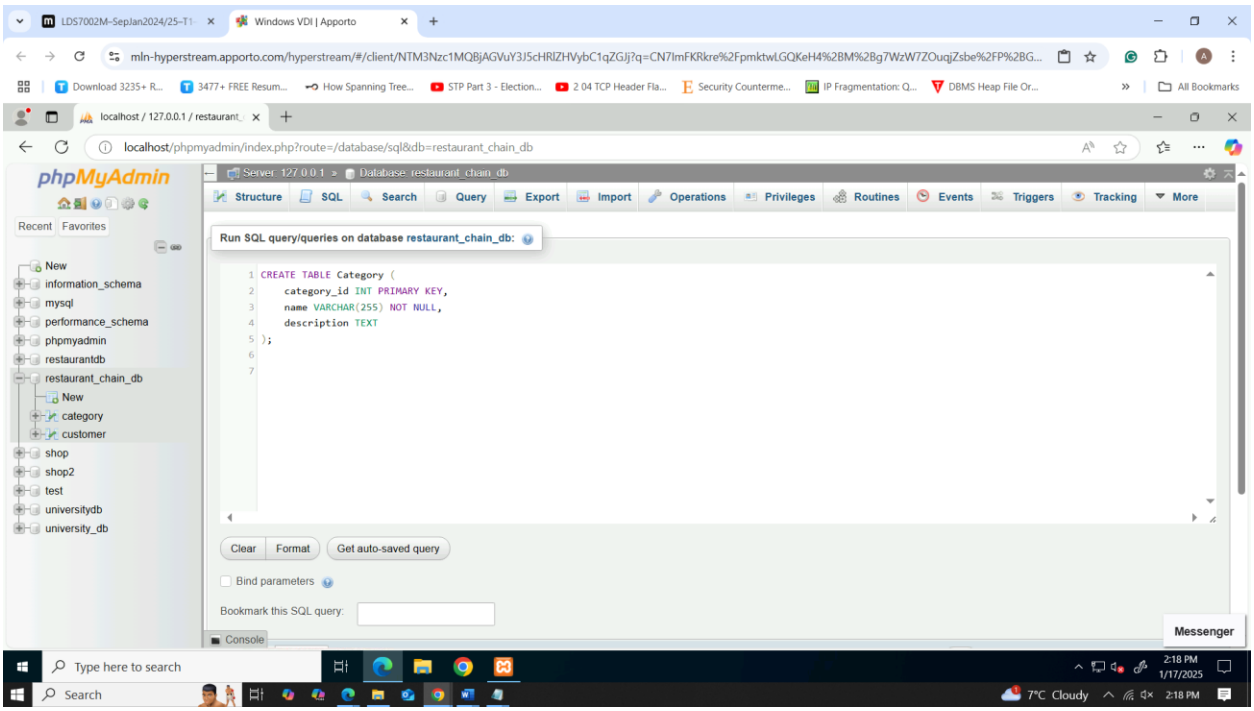


Figure 13: create table query of category

The Category table organizes menu items into distinct groups with three columns: category_id (an integer field that is Primary key and auto incremented, used to uniquely identify each category), name (string field for category name which can contain up to 255 characters) and description (used to contain additional information, actually a text field). This table makes the items appear consistent and it can easily be categorized on the menu. The Menu_Item table use the category_id in it which is a foreign key that helps in relating the items to the appropriate categories to avoid data inconsistency as well as to enhance easy sorting of the menu categories.

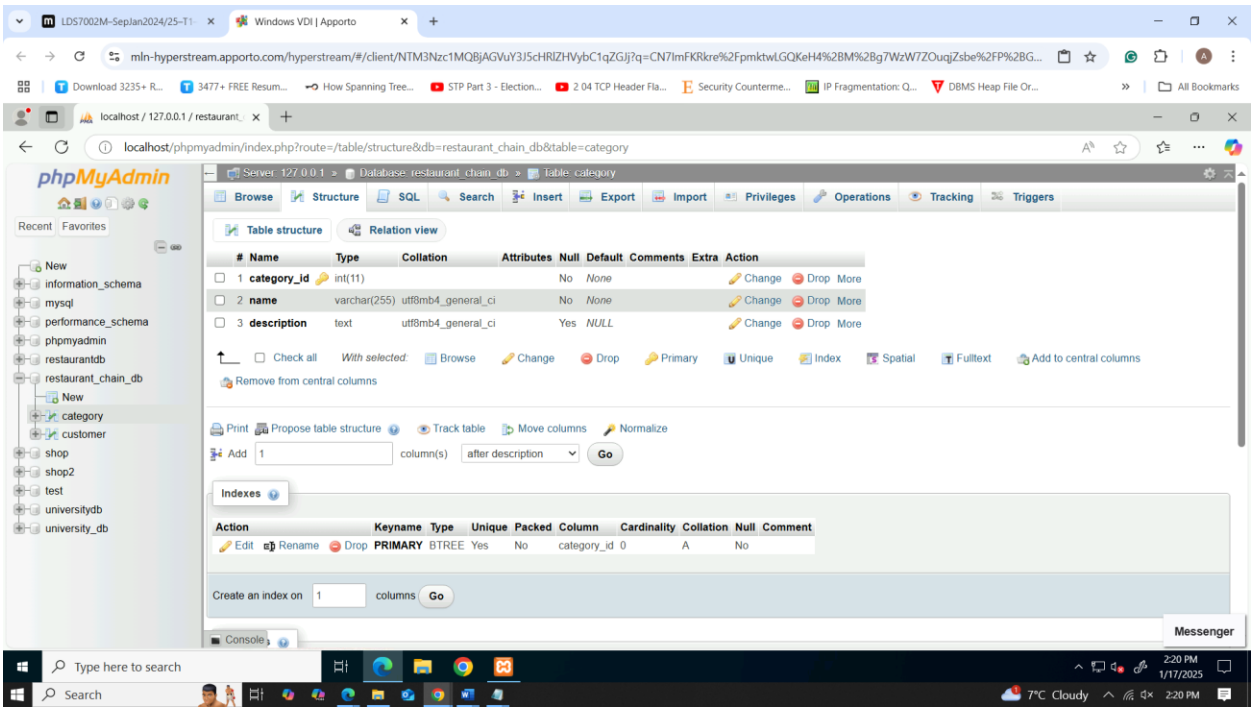


Figure 14: structure of category table

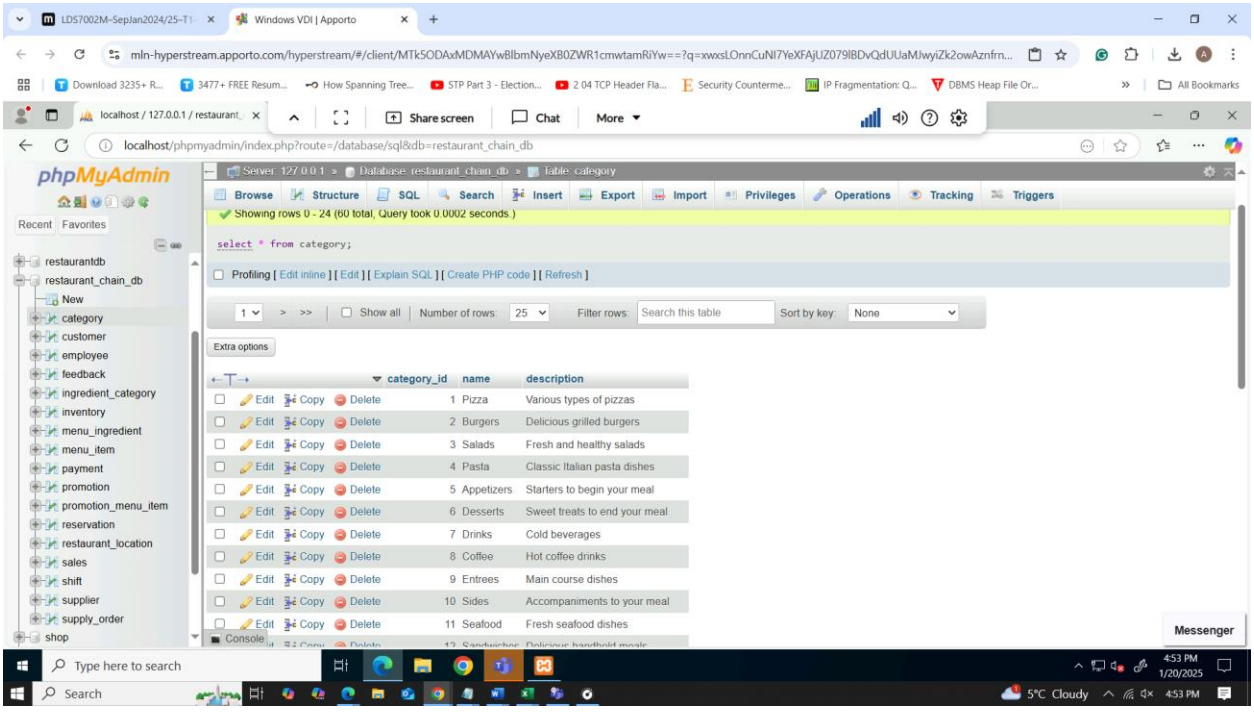


Figure 15: select query for category table

- Customer

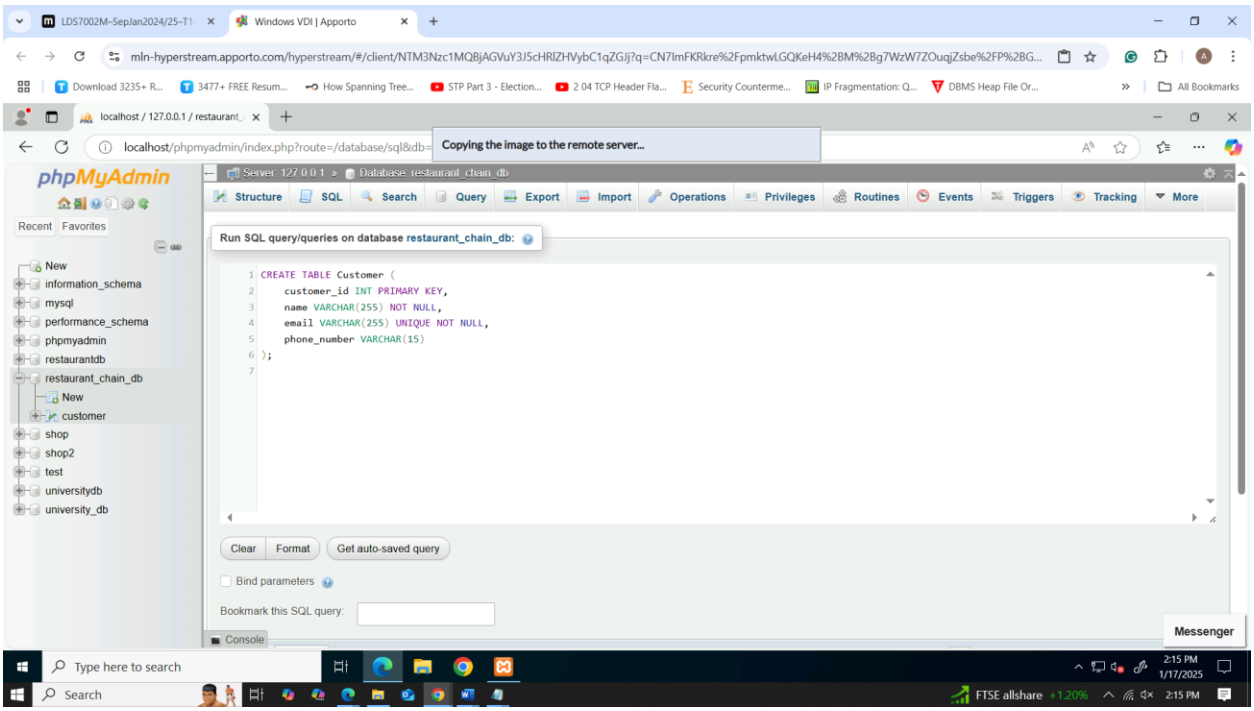


Figure 16: create table query of customer

The SQL statement generates the structure of the Customer's table consisting of five columns: customer_id, which is defined as the primary key and cannot accept null values, represents a unique integer; name, which is a mandatory string of up to 255 symbols; email, which is also a mandatory string of up to 255 symbols, cannot contain duplicate values; phone_number which is an optional string of up to 15 symbols. This structure also reduces the possibility of having duplicate complications and Governor putting restrictions on fields such as the email field.

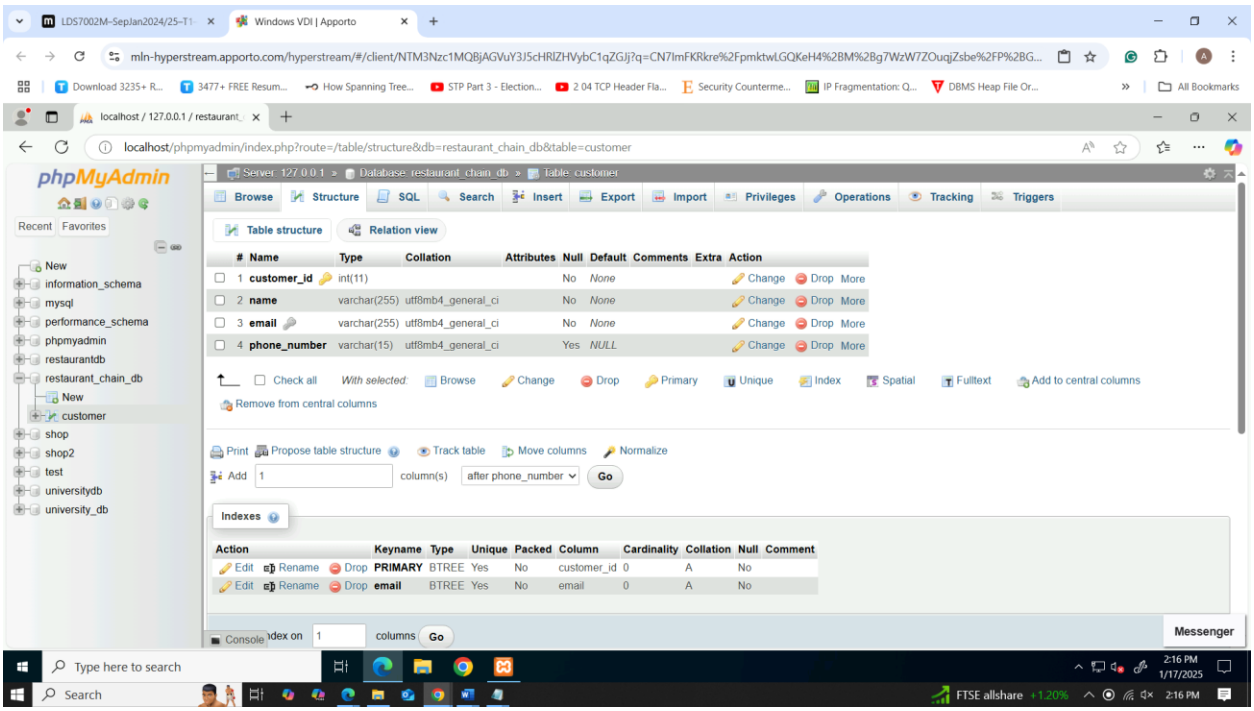


Figure 17: structure of customer table

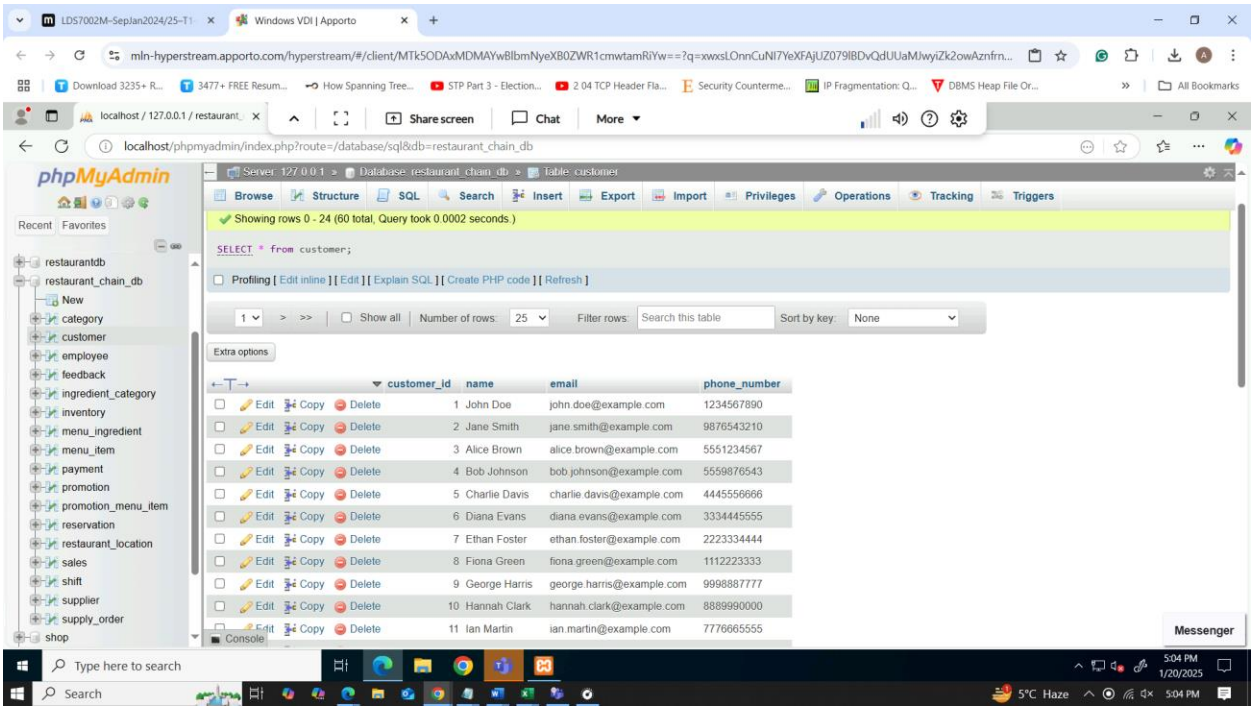


Figure 18: select query for customer table

- Reservation

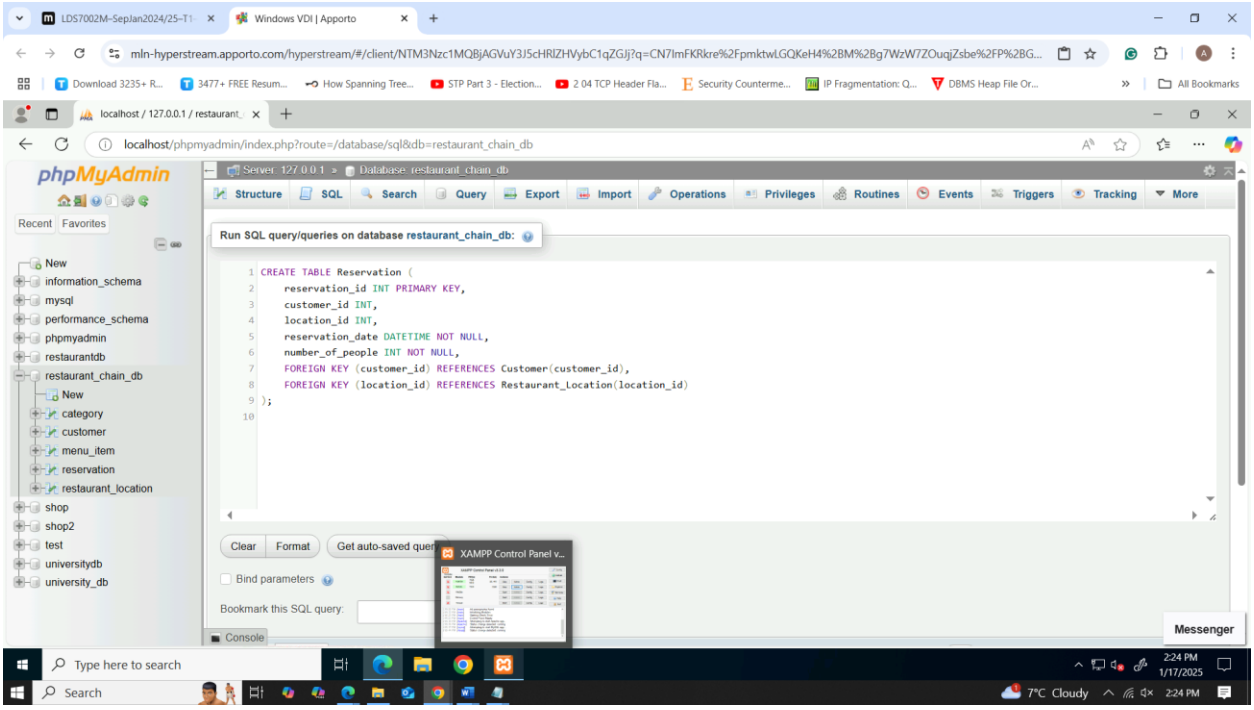


Figure 19: create table query of reservation

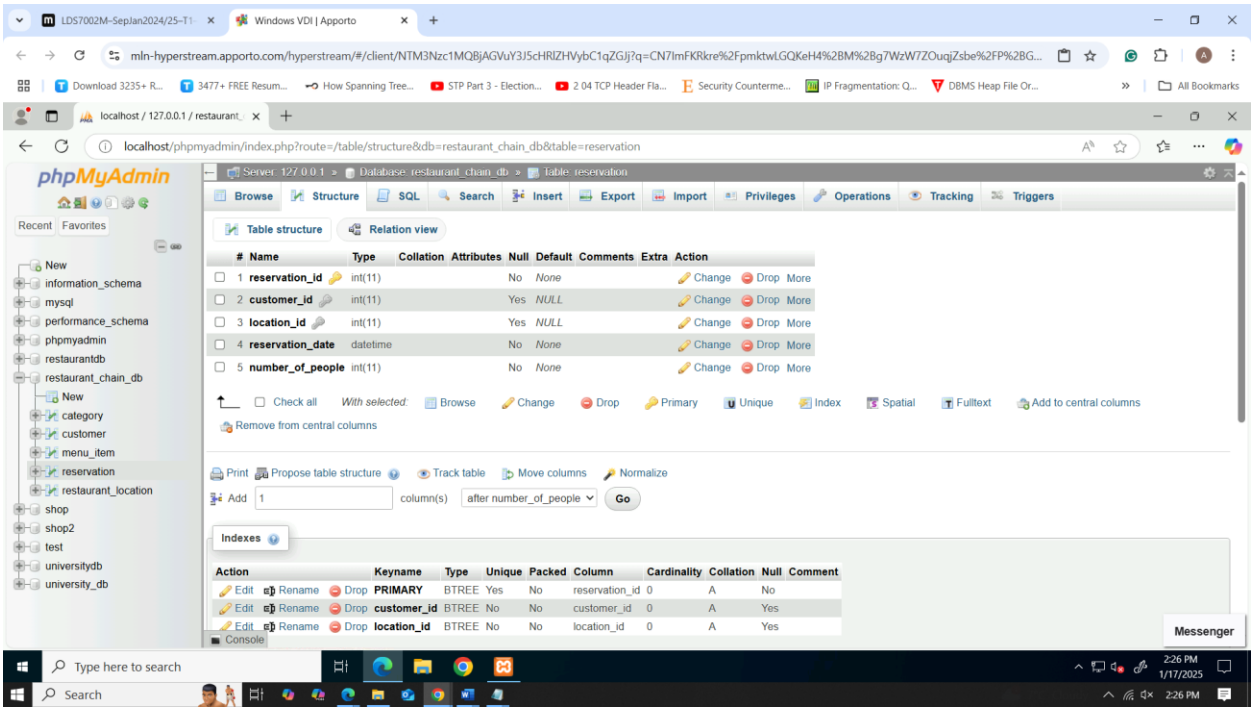


Figure 20: structure of reservation table

The Reservation table manages booking details with six columns: reservation_id(primary key-integer,unique identifier for each reservation), customer_id (used to link the two tables with foreign key pointing to the Customer table, customer_id used here is required), location_id (used to connect the two tablesby using foreign key pointing to the Restaurant_Location table, location_id used here is also required), reservation_date DATETIME, not flexible-needed; number_of_people-int, not flexible-needed. The relationships that this table defines are simple but effective: defining how customers are related to restaurants, restaurants are related to reservations and how all of them can be related to customers to maintain data integrity as well as track all restaurants bookings effectively.

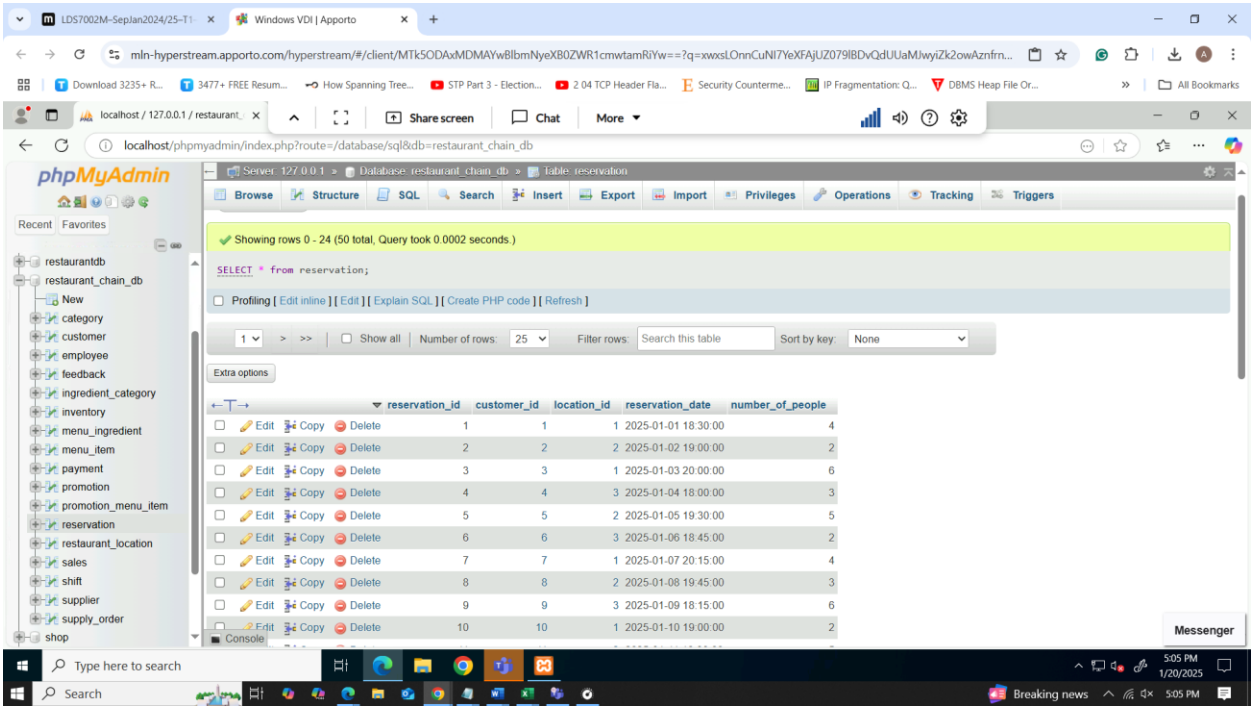


Figure 21: select query for reservation table

- Restaurant location

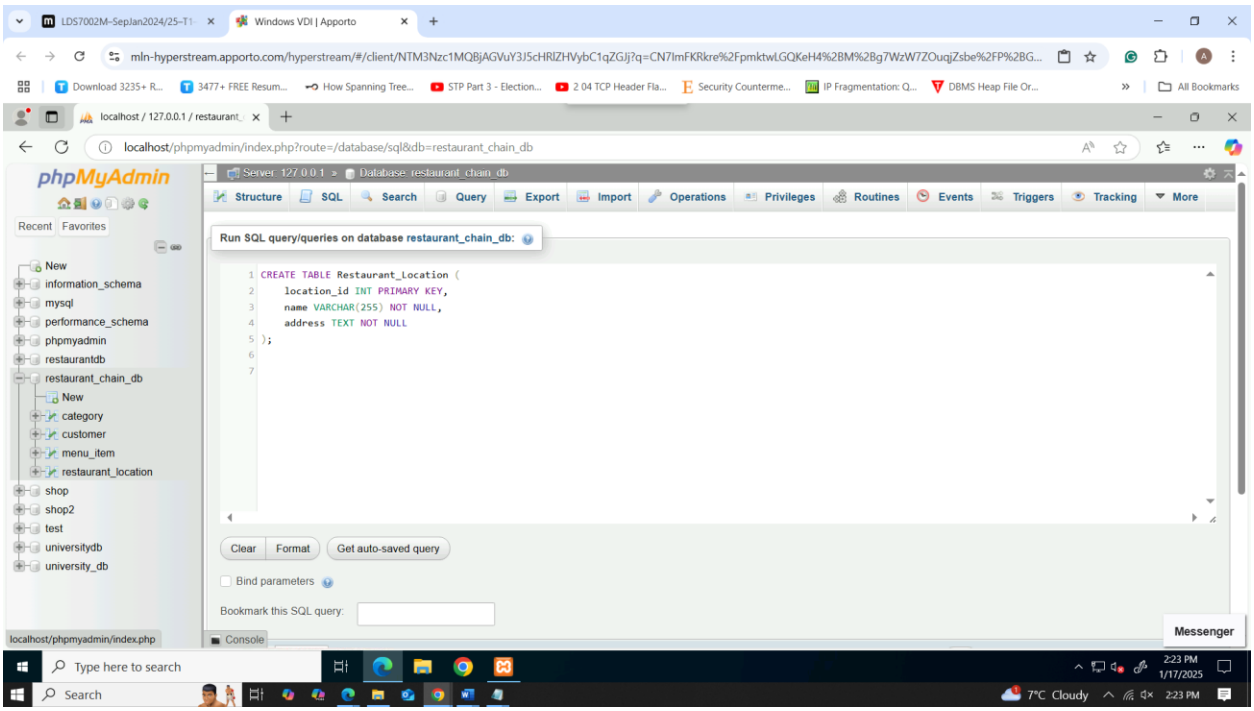


Figure 22: create table query of restaurant location

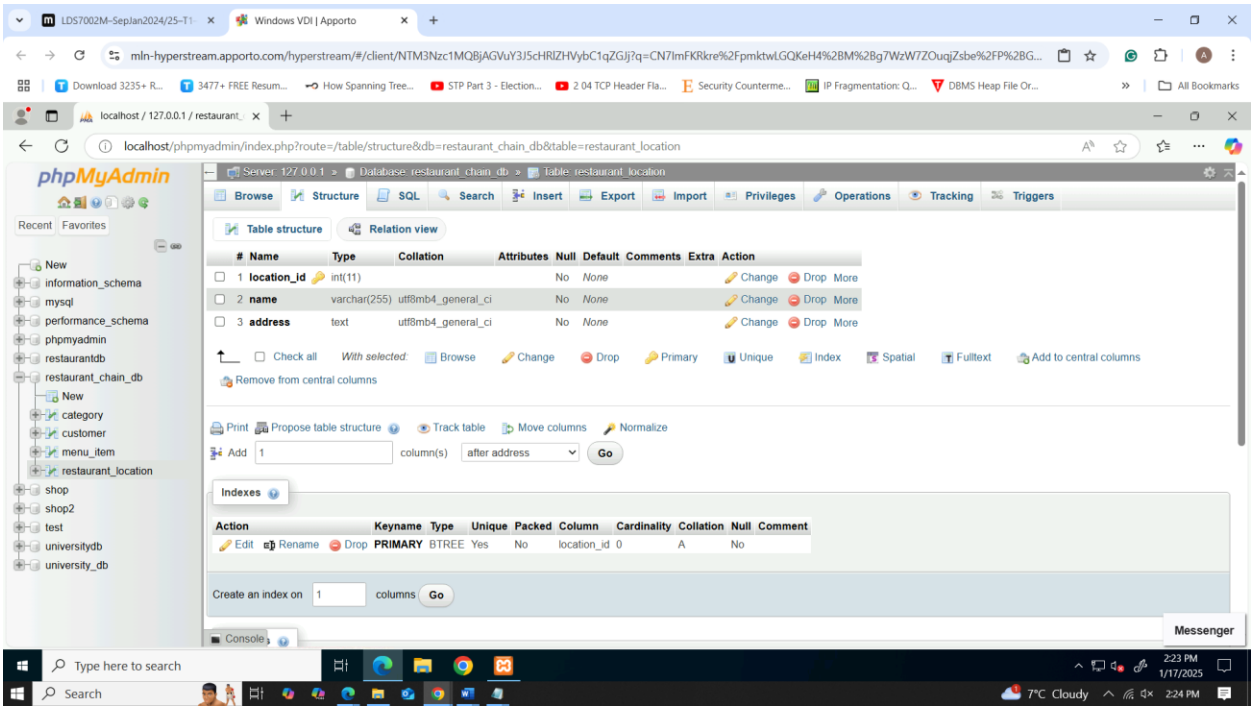


Figure 23: structure of restaurant location table

The Restaurant_Location table stores information about restaurant locations with three columns: location_id (an integer auto incremented unique field), name (a field that contains the name of the location, with a maximum length of 255 characters) and address (a required field for the full address of the location). It is also important to provide all the necessary information concerning the restaurant location and which acts as a key to make reference with other tables e.g. Reservation table where location_id is a foreign key that links all the reservations with particular restaurants. His design is coherent and makes organizing location-related data less problematic.

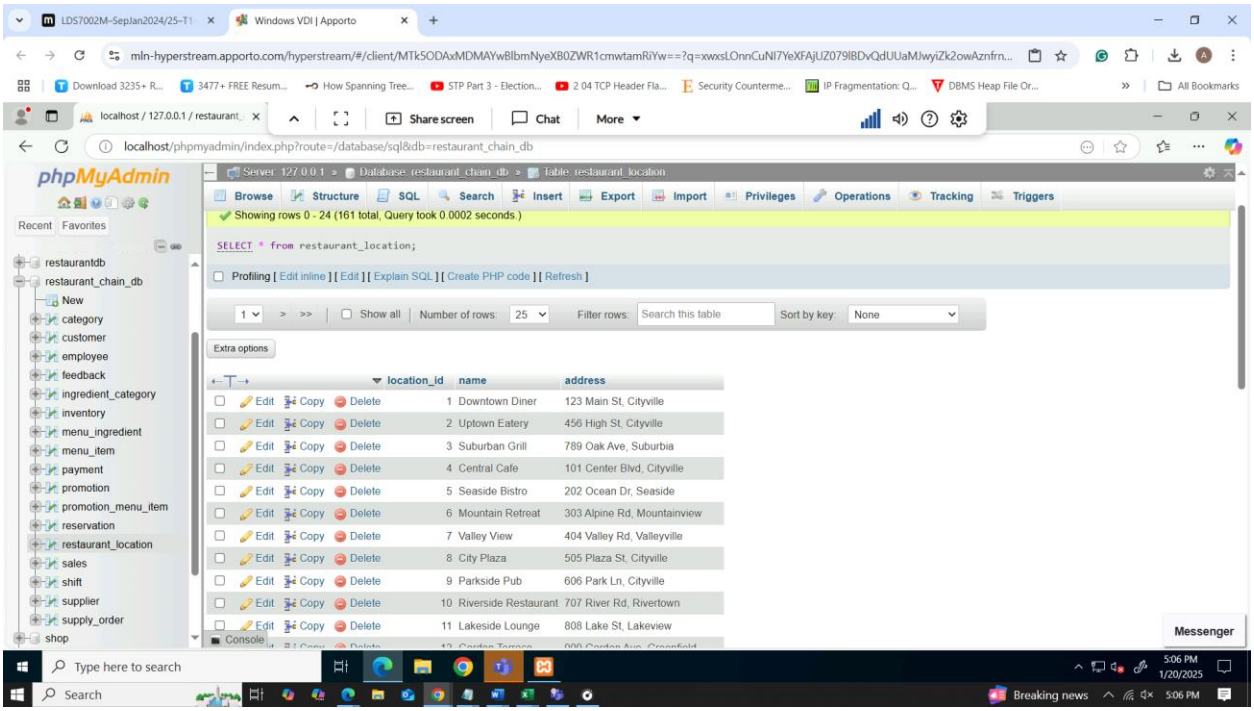


Figure 24: select query for restaurant location table

- Sales

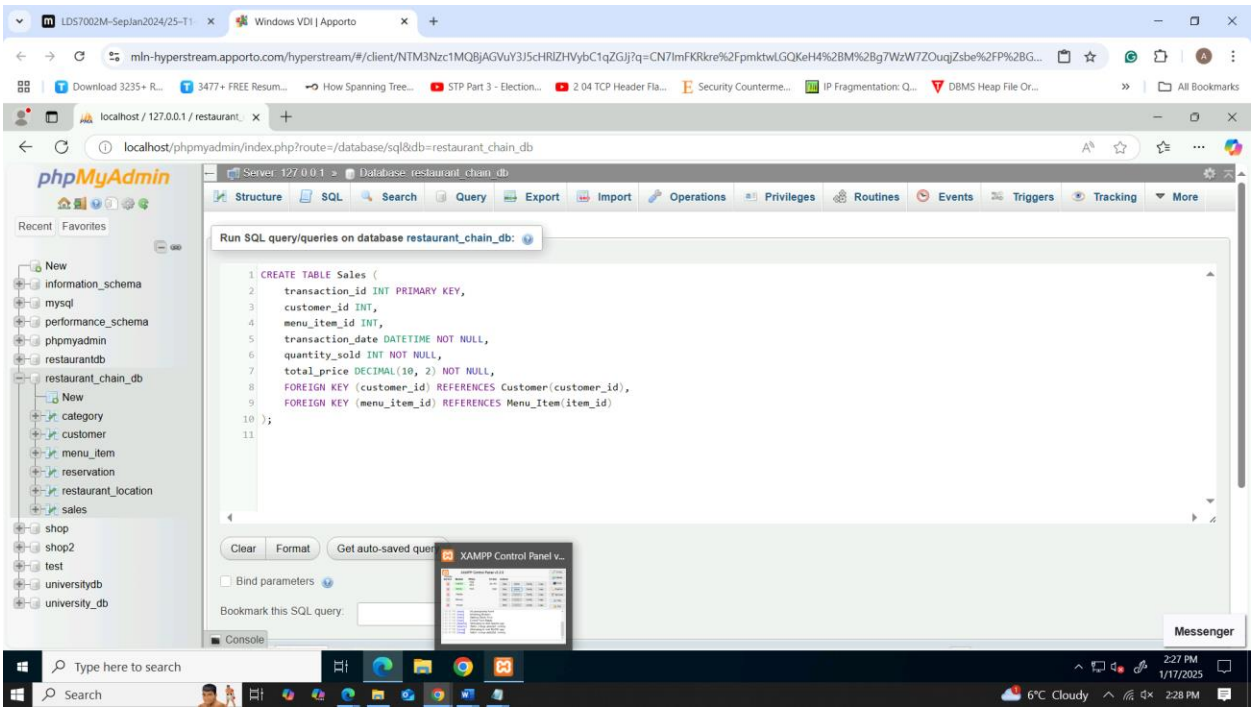


Figure 25: create table query of Sales

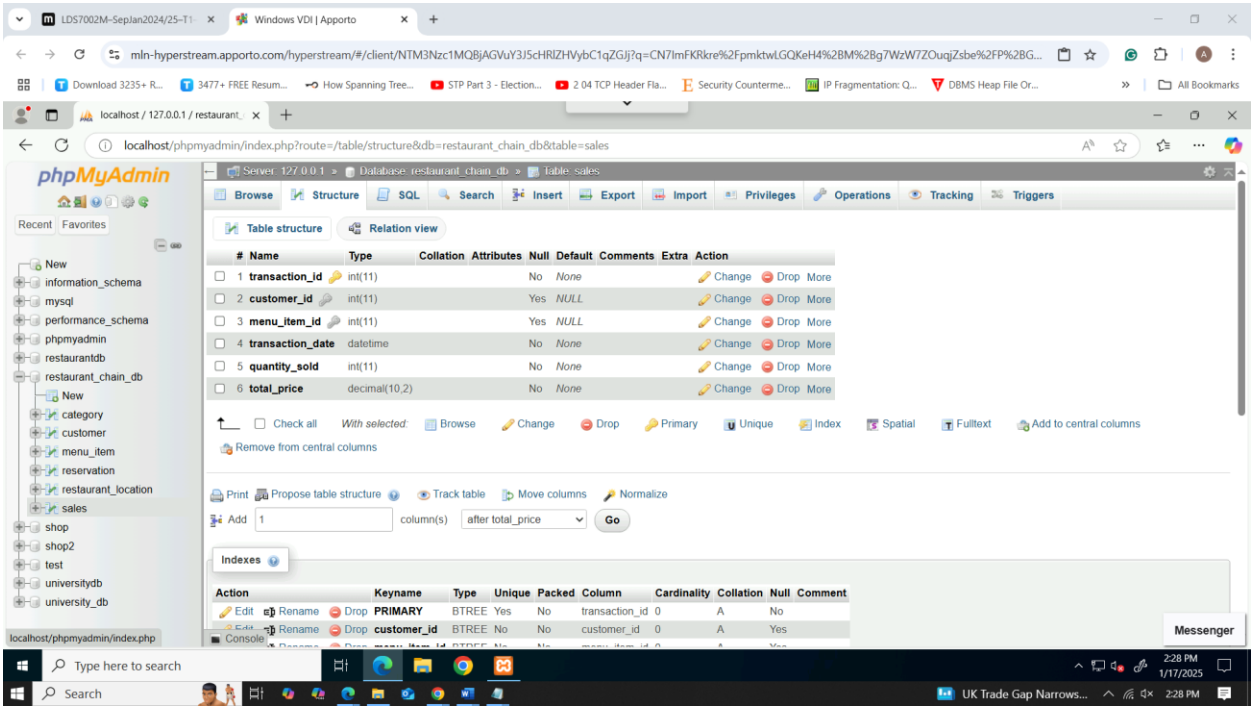


Figure 26: structure of sales table

The Sales table tracks sales transactions with six columns: transaction_id (an integer field serving as the primary key for the transactions table), customer_id (an integer used as a foreign key in reference to the

Customer.customer_id to connect the transaction to the original customer), menu_item_id (a foreign key referencing the Menu_Item.item_id used to determine the sold menu item), transaction_date (a required DATETIME field specifying the date and time of the transaction), quantity_sold (an integer indicating how many units of This table is also used to monitor and control sales information through the relation between customers and food options offered.

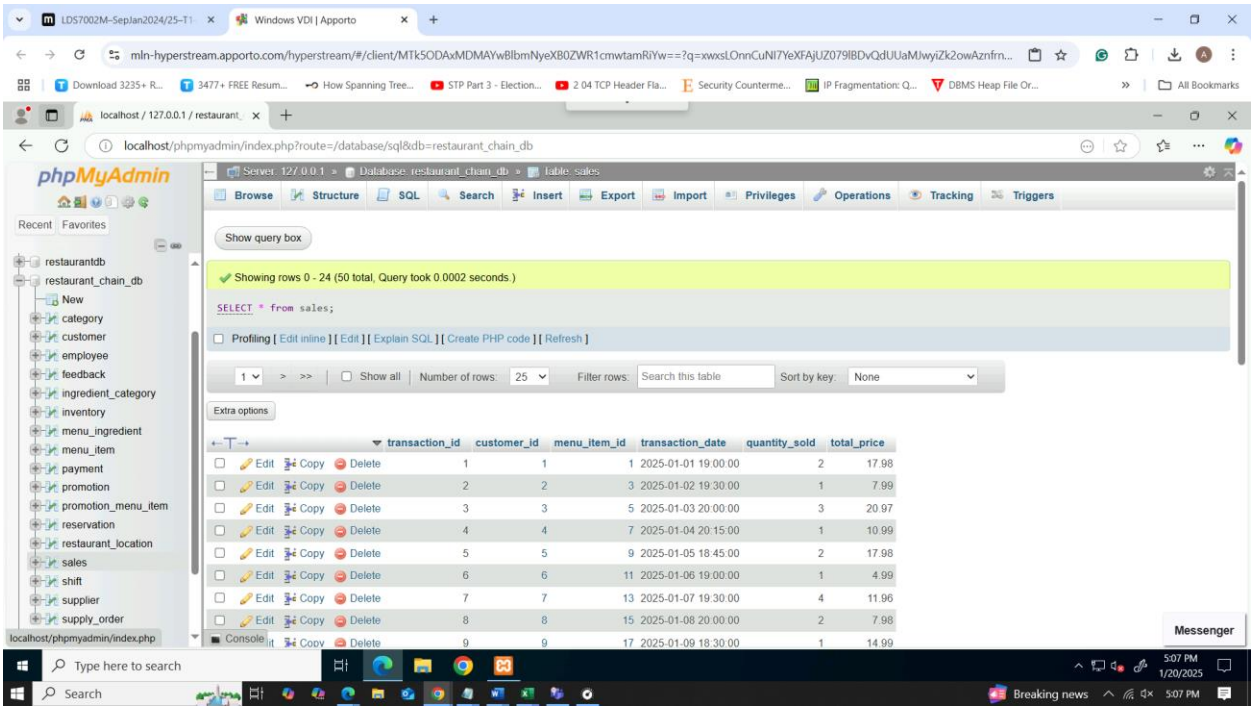


Figure 27: select query for sales table

- Inventory

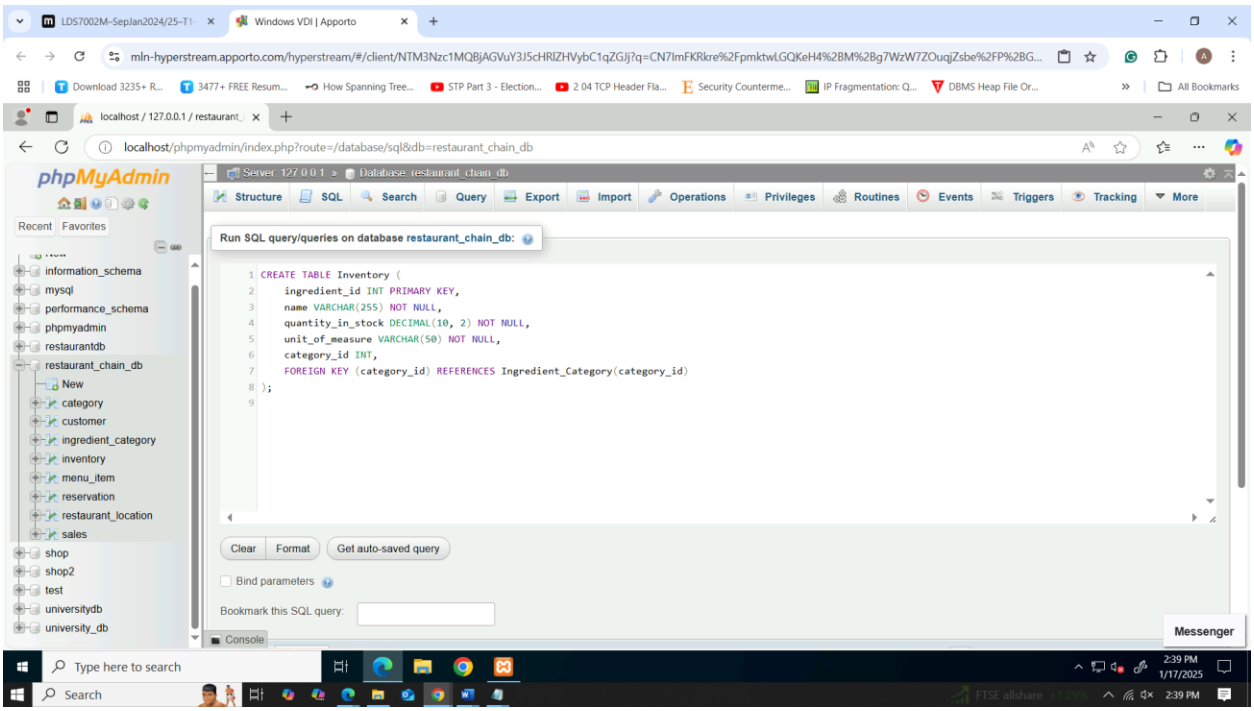


Figure 28: create table query of inventory

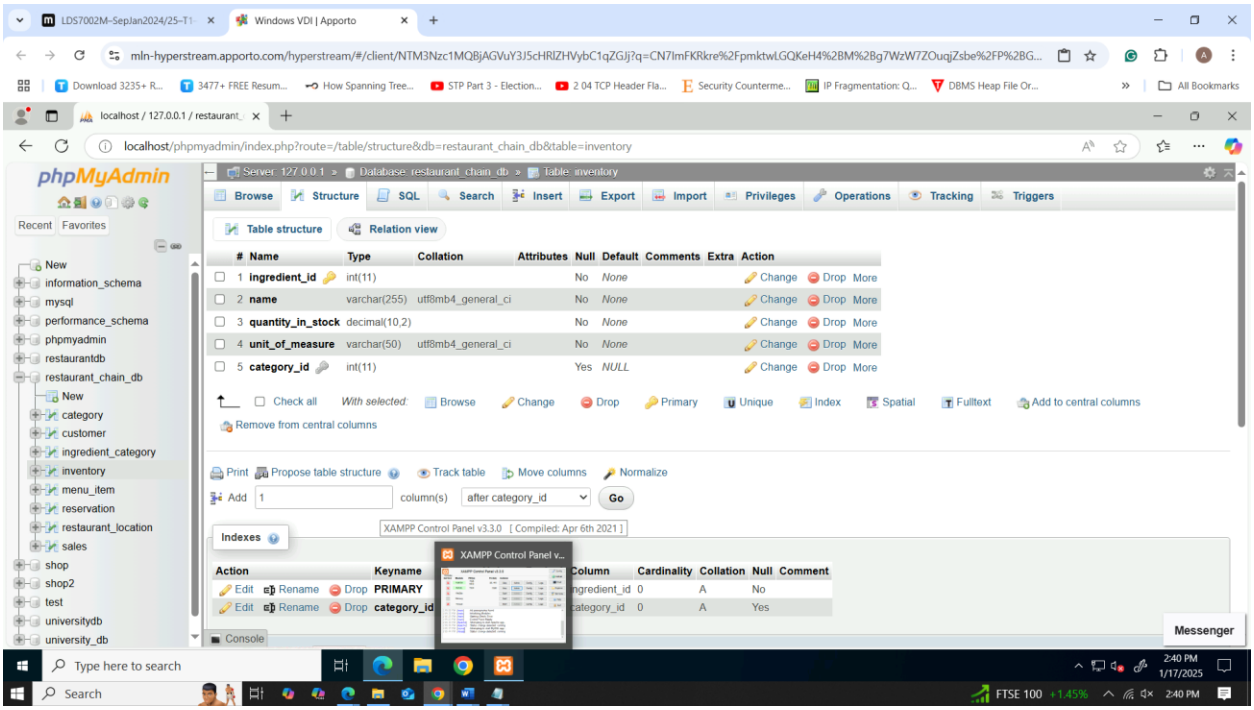


Figure 29: structure of inventory table

The Inventory table stores information about ingredients used in the restaurant, with five columns: ingredient_id (an integer.primary key to identify each ingredient uniquely), name (string required 255 character for the ingredient name), quantity_in_stock (required field, DECIMAL 10, 2 to store the quantity of the ingredient in stock), unit_of_measure (string required 50 character to have the unit of measure which can be kg/liter), and category_id (foreign key referencing the category of the ingredient using Ingredient_Category's category_id). It is useful to use this table for managing of ingredients and keep the check through the category reference table.

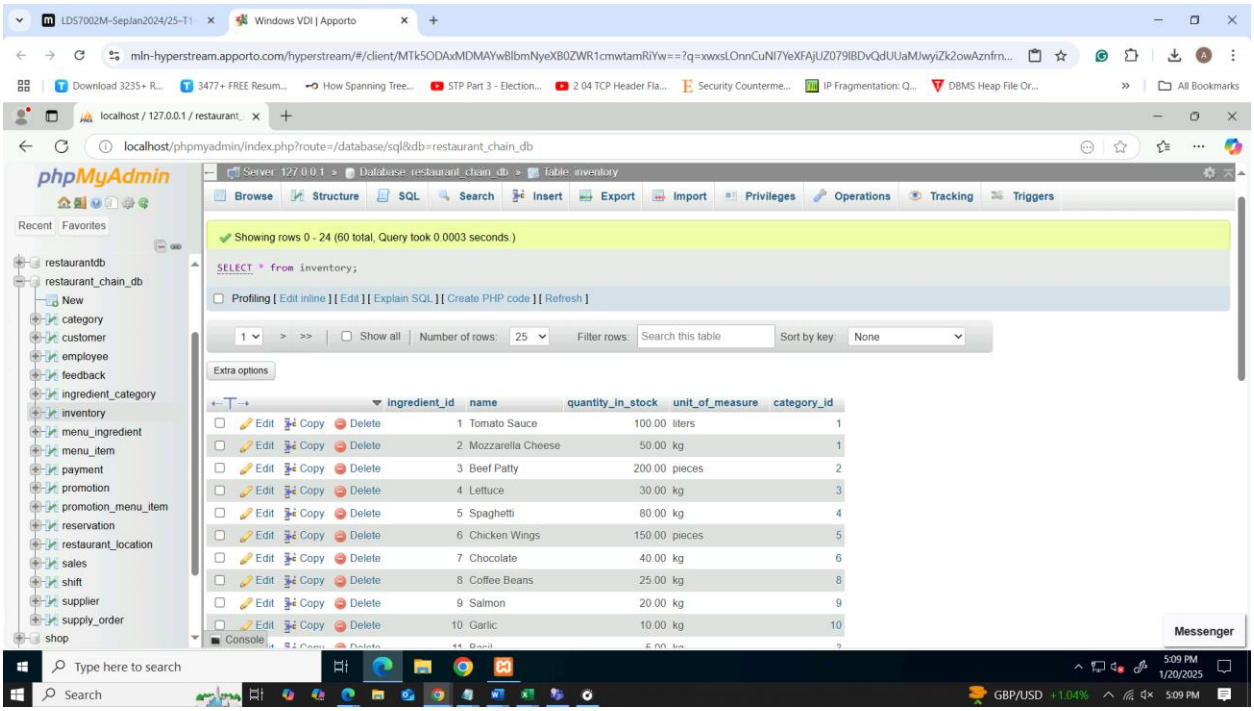


Figure 30: select query for inventory table

- Ingredient Category

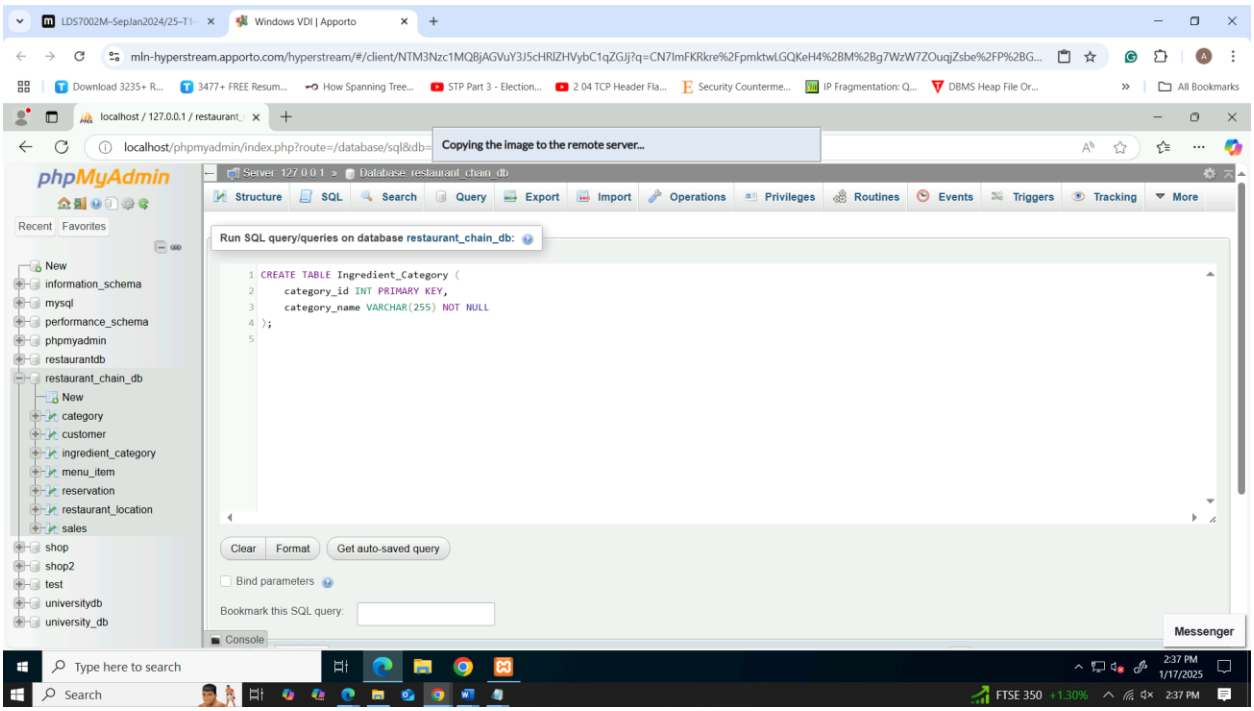


Figure 31: create table query of ingredient category

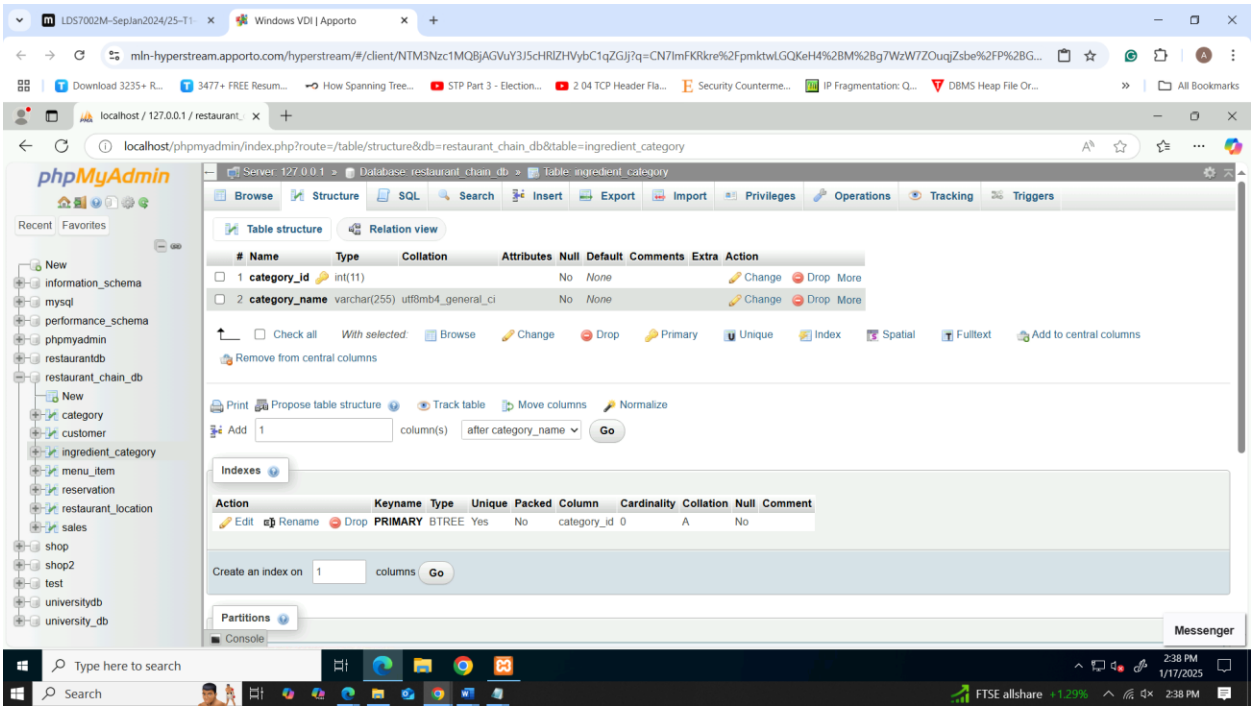


Figure 32: structure of ingredient category

The Ingredient_Category table organizes ingredients into specific categories, with two columns: The relevant fields are category_id (an integer with no less than 0, but no more than 2147483647, being the primary key of the table and uniquely identifying each category) and category_name (a mandatory string of no more than 255 characters containing the name of the category). This table enables.shtml the organisation of ingredients in a meaner that will facilitate their tracking and inventory management. This category is linked to the Inventory table by the category_id foreign key so that every ingredient can be linked with a valid category. This structure also enhances a better way of storage and identification of these ingredients in the restaurant's stock management system.

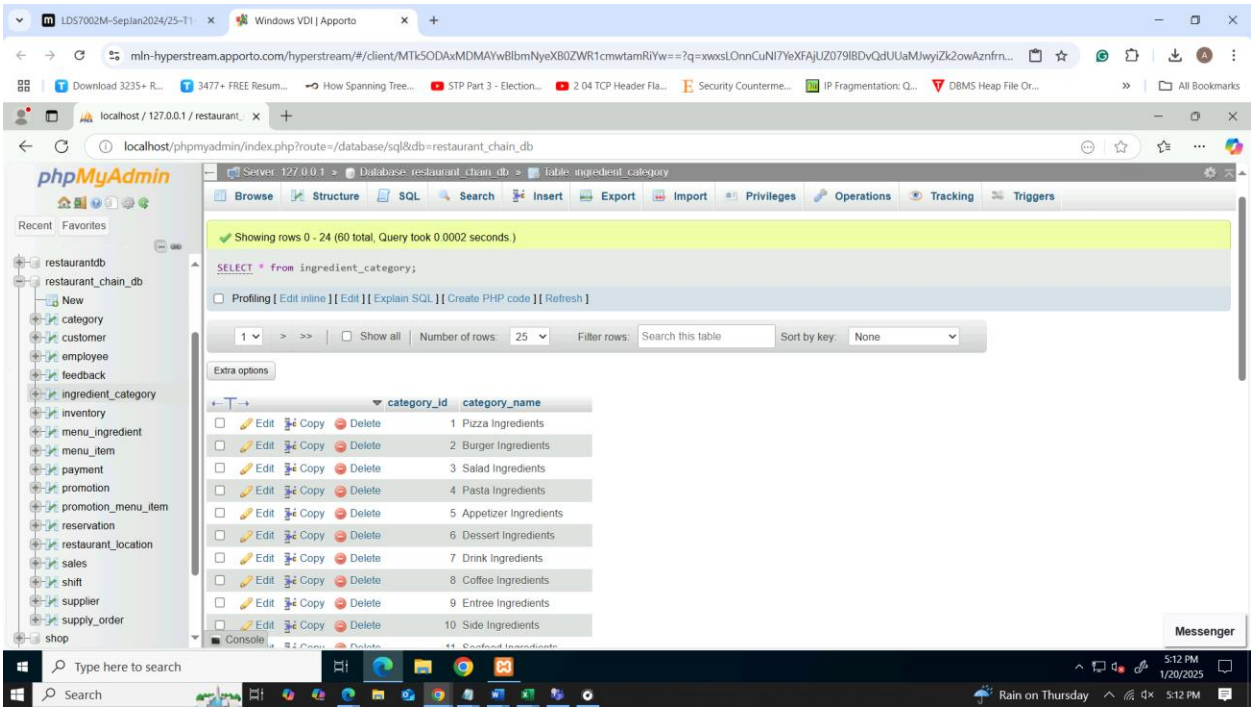


Figure 33: select query for ingredient category table

- Menu Ingredient

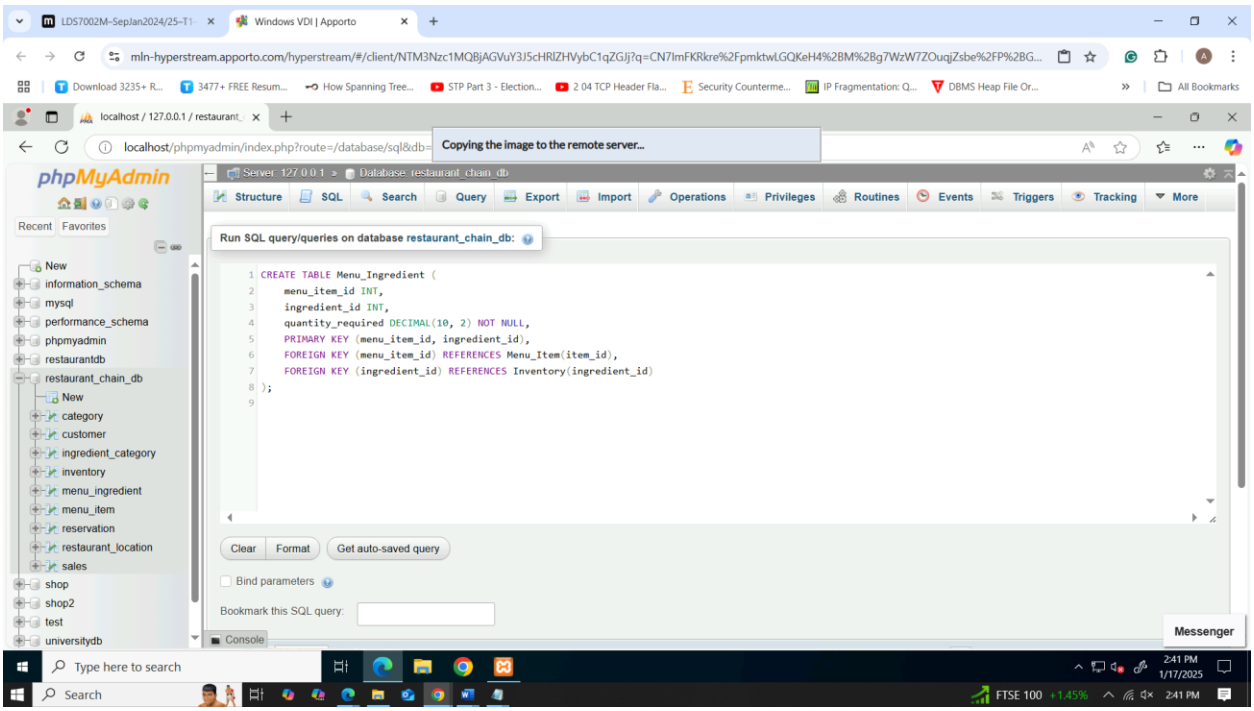


Figure 34: create table query of menu ingredient

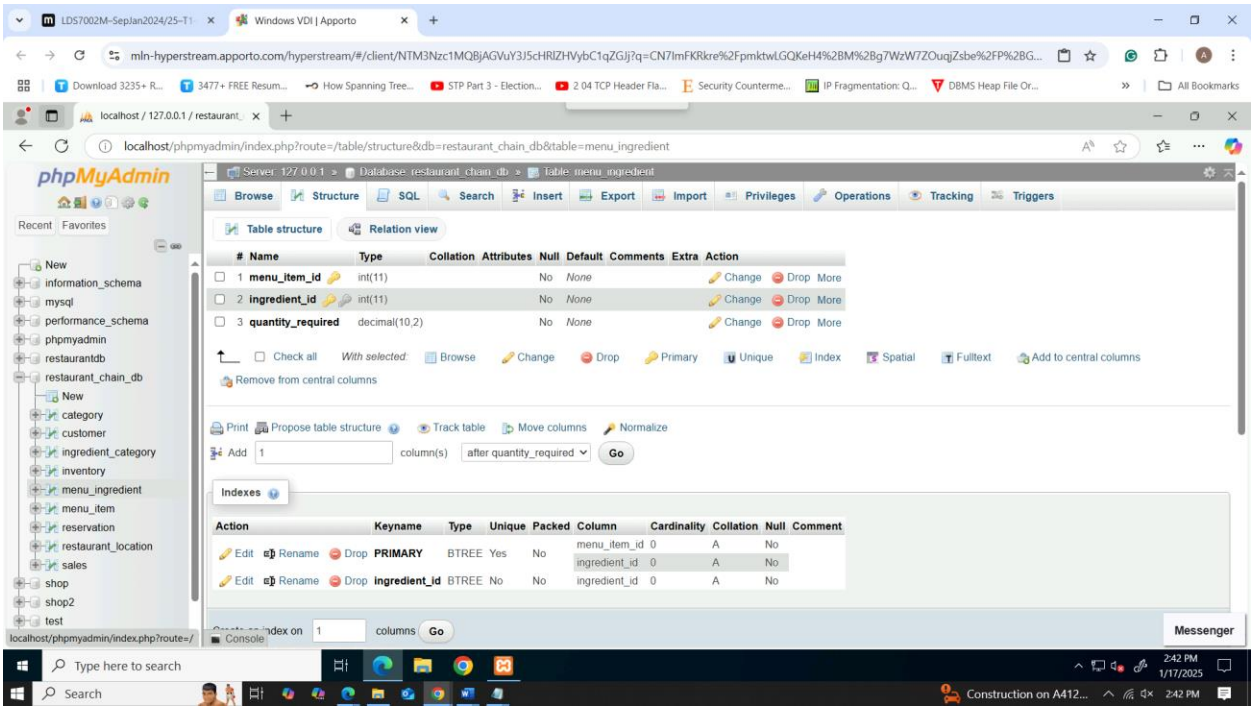


Figure 35: structure of menu ingredient

The Menu_Ingredient table links menu items to the ingredients required to prepare them, with three columns: Menu_Item ID (a reference link key to the Menu_Item table to identify which ingredient belongs to which menu item), Ingredient ID (a reference link key to the Inventory.item ID to state which ingredient is used), Quantity required (data type DECIMAL(10, 2) which is mandatory to define how much of the ingredient is required to prepare the menu item). The primary key is chosen as menu_item_id and ingredient_id as each of these make up a unique entity. This table supports the identification of the ingredients served with each dish and provides control and consistency between dishes and stocks.

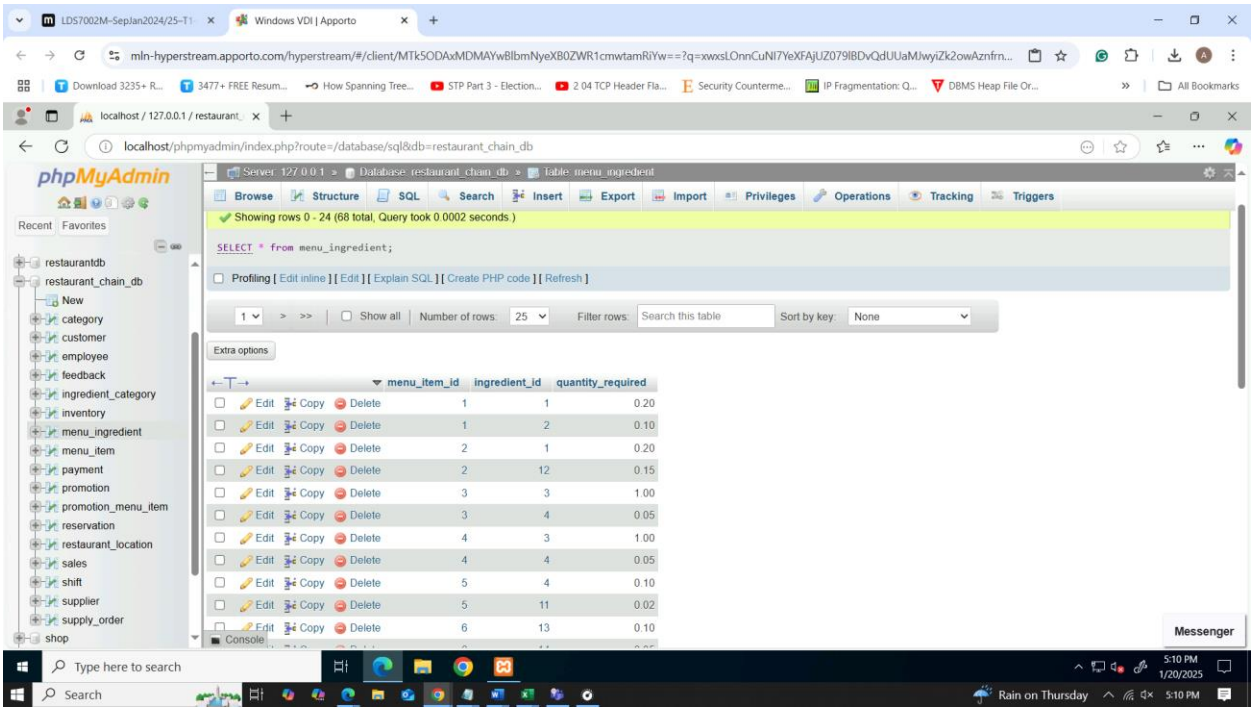


Figure 36: select query for menu ingredient table

- Employee

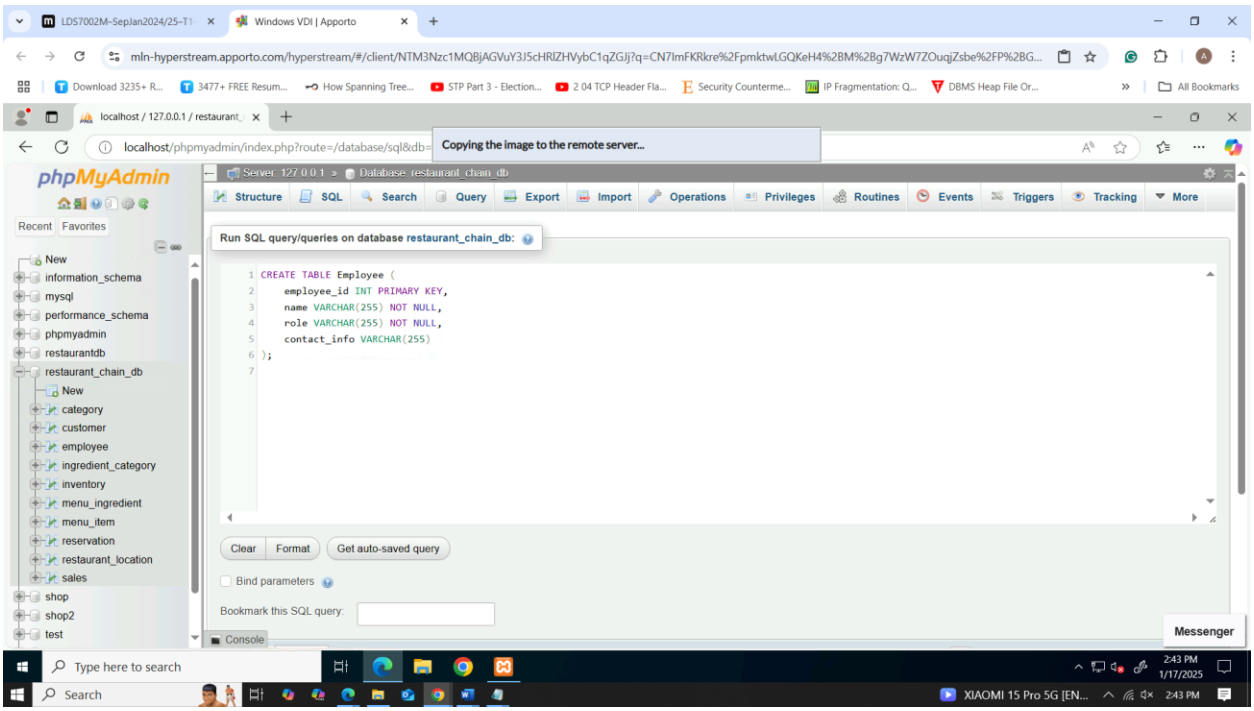


Figure 37: create table query of employee

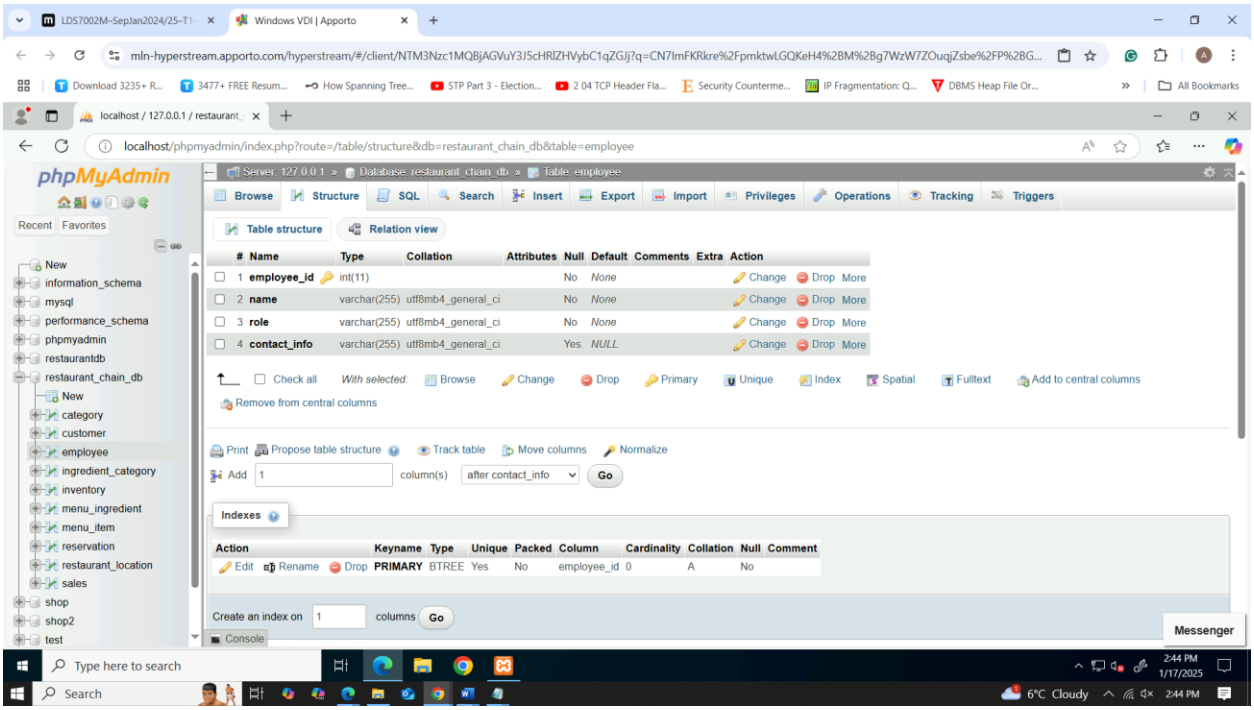


Figure 38: structure of menu ingredient table

The Employee table stores information about employees with four columns: employee_id, an integer unique key for each employee, name, which has a constraint to be mandatory string and must contain less than 256 characters to hold the name of the employee, role, which should also be a mandatory string with less than 256 characters to contain the position/department/role of the employee and contact_info, which is an option string containing less than 256 characters to include the phone number and/or email of the employee. This table assists in organization of the employees, tracking of those working within the company and the position they hold.

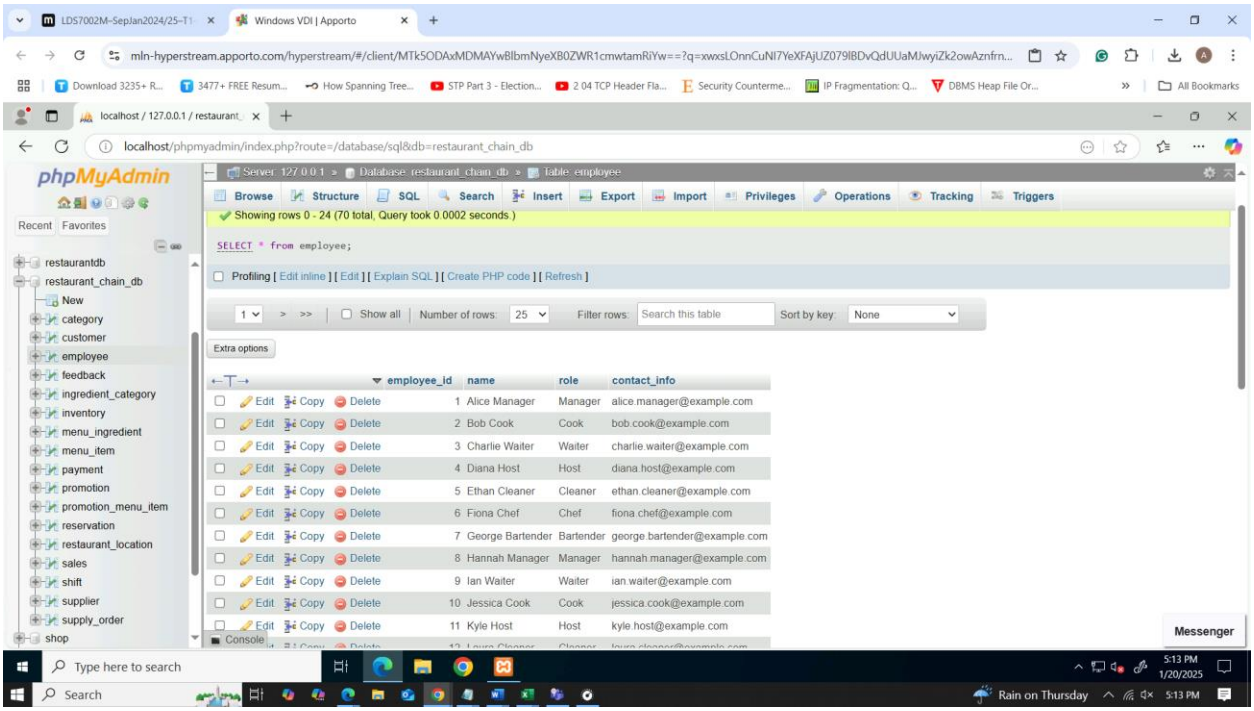


Figure 39: select query for employee table

- Shift

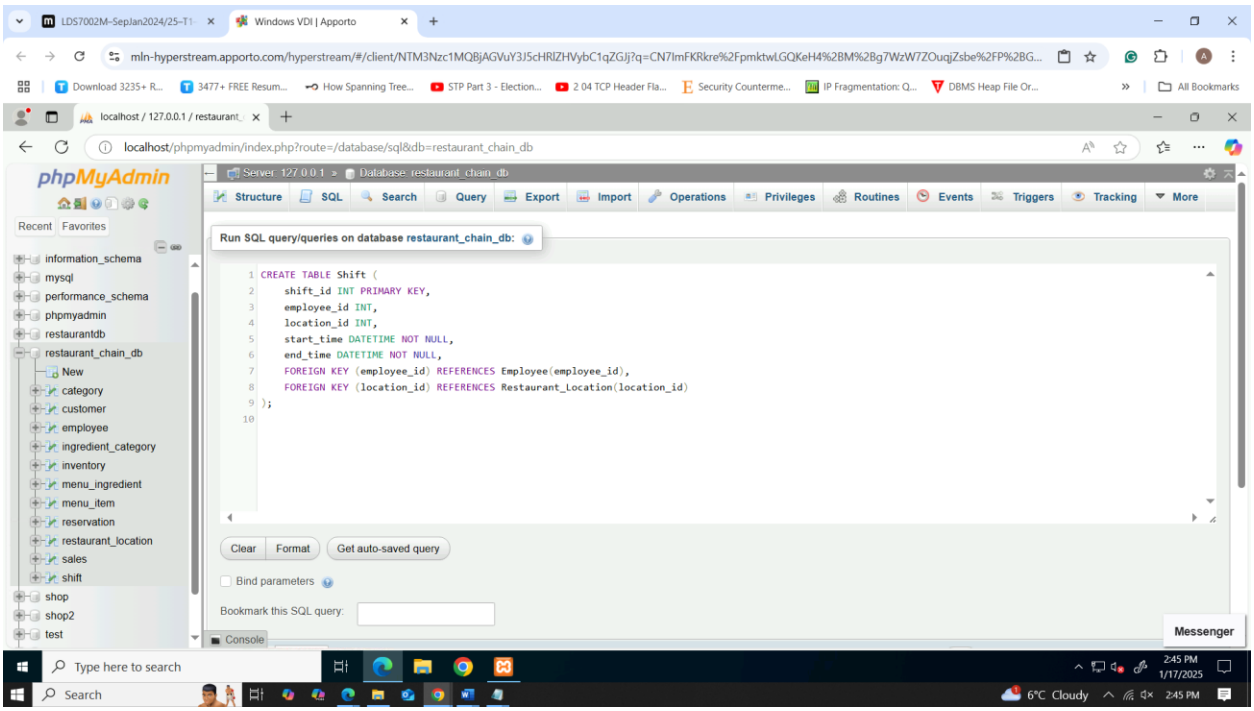


Figure 40: create table query of shift

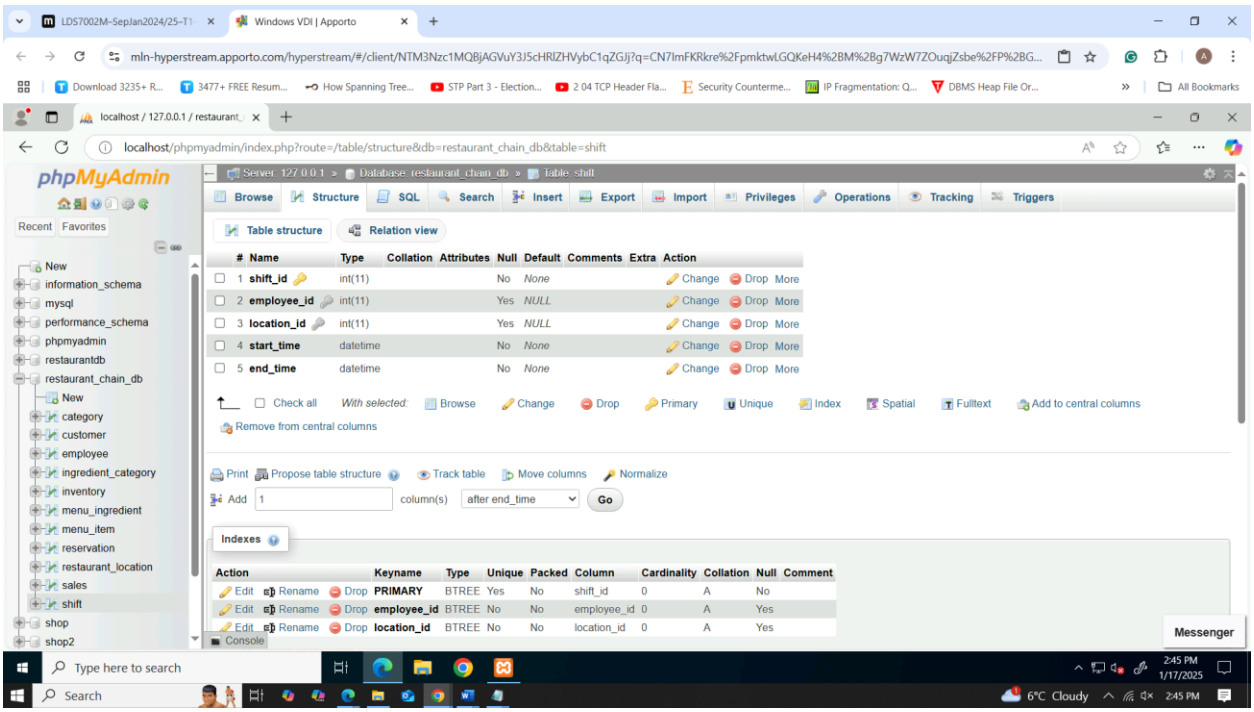


Figure 41: structure of shift table

The Shift table tracks employee work shifts with five columns: shift_id – integer unique identifier of the shift, employee_id – a foreign key referencing Employee Table to link the shift to the employee, location_id – a foreign key linking the shift to the restaurant location, start_time – a required DATETIME field indicating the time the shift started, end_time – a required DATETIME field indicating the time the shift ended. This table is useful in the employee scheduling process as it links the schedules to employees and restaurant locations for easy running.

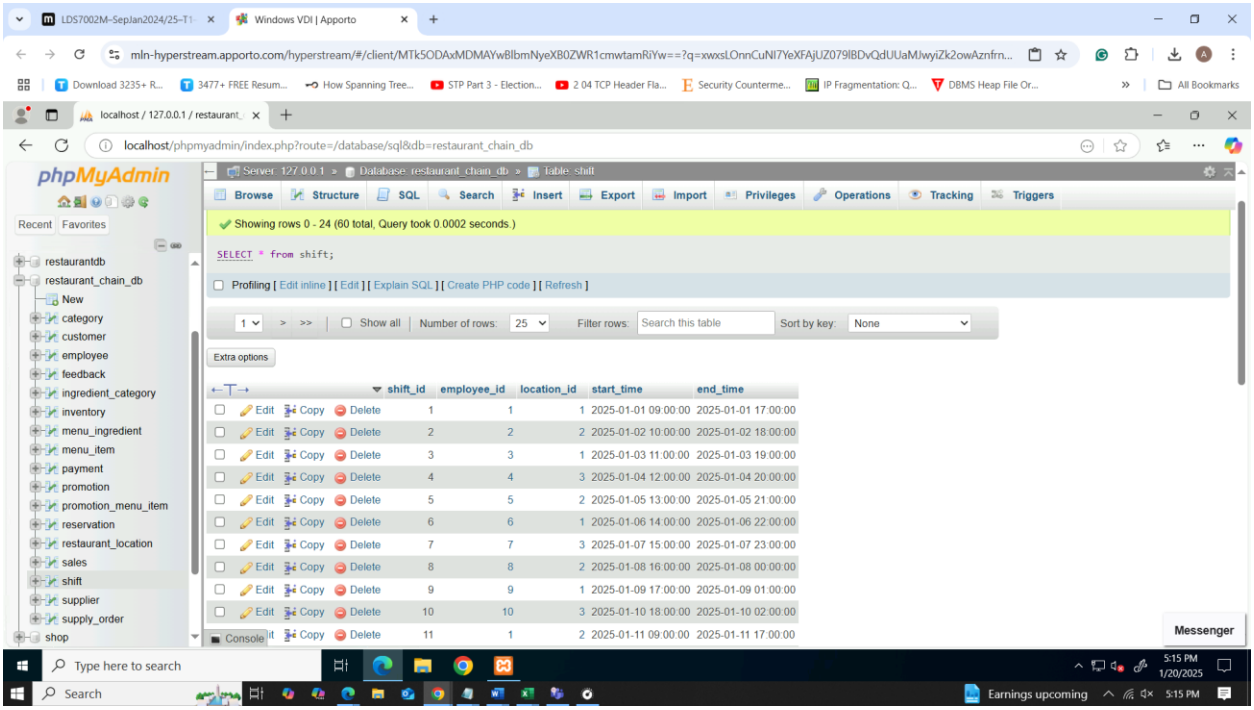


Figure 42: select query for shift table

- Supplier

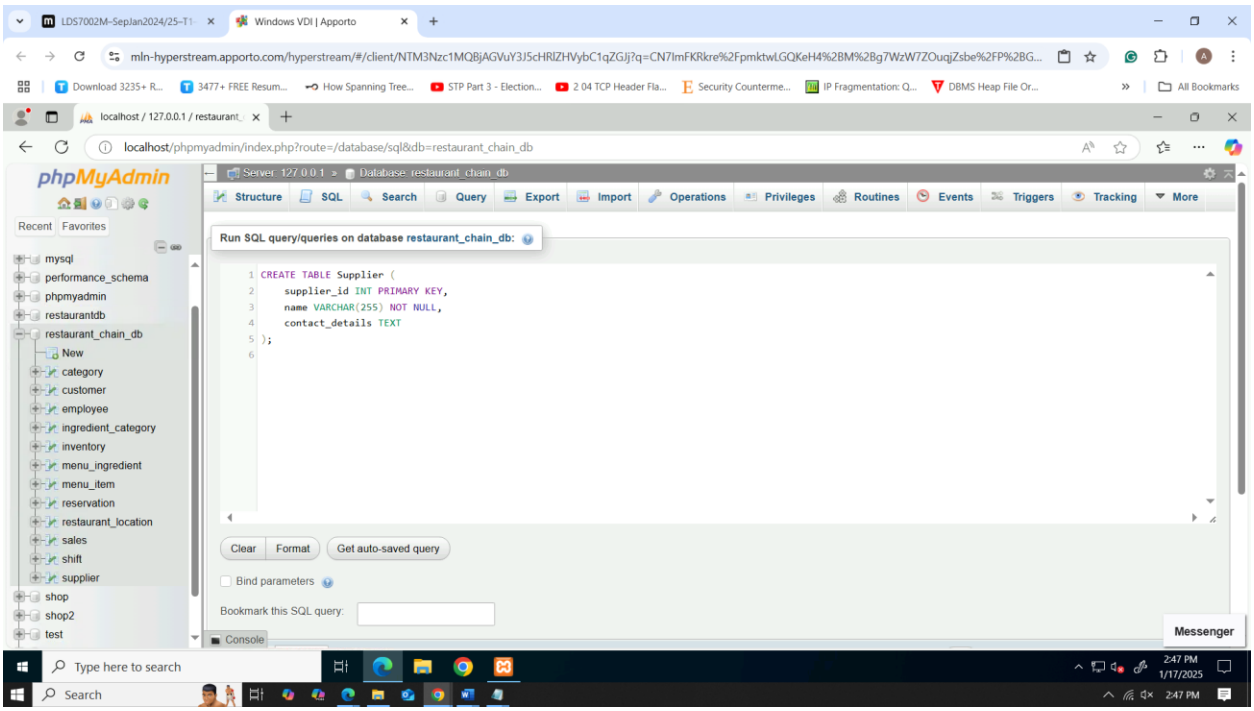


Figure 43: create table query of supplier

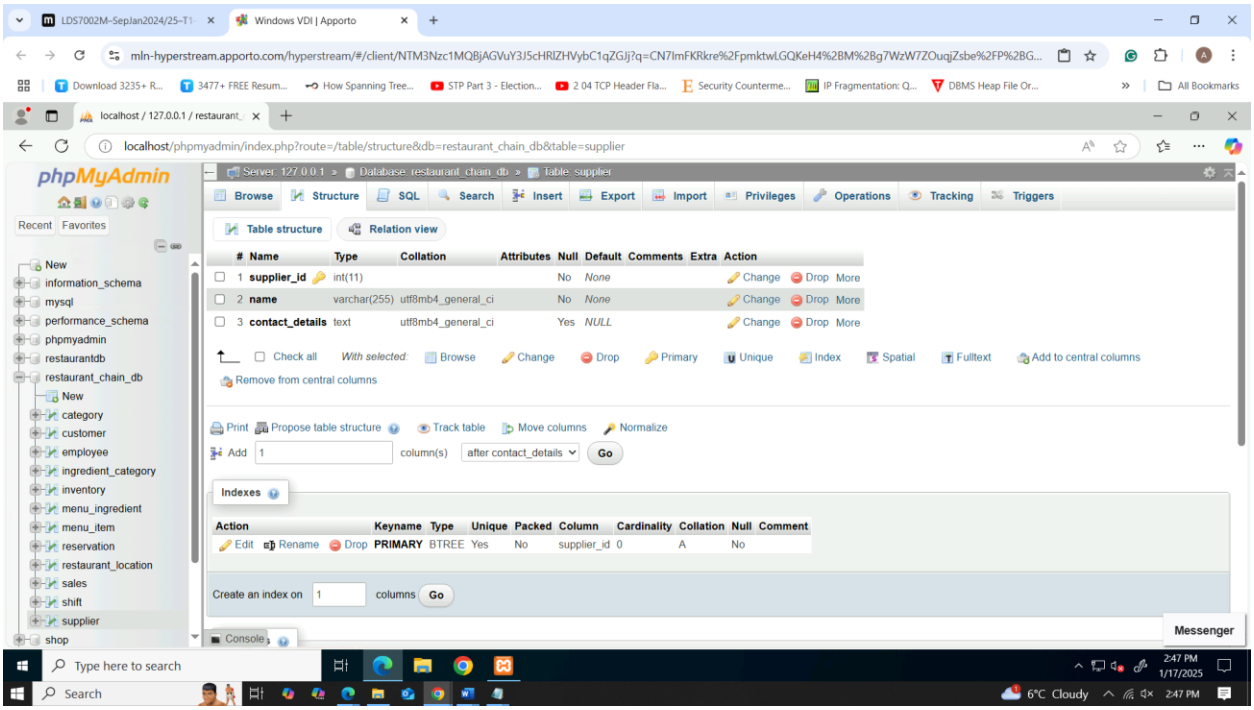


Figure 44: structure of supplier table

The Supplier table stores information about suppliers with three columns: name (an essential string field of 255 characters for the supplier name), and contact_details (a text type field to store all the supplier's contact information such as phone numbers, emails, and/or physical address, and supplier ID, which is an auto-generated integer to uniquely identify each supplier). This table assists in the relationship management of suppliers to guarantee that every supplier is recognized individually and his or her information is readily accessible for purchase and supply chain purposes.

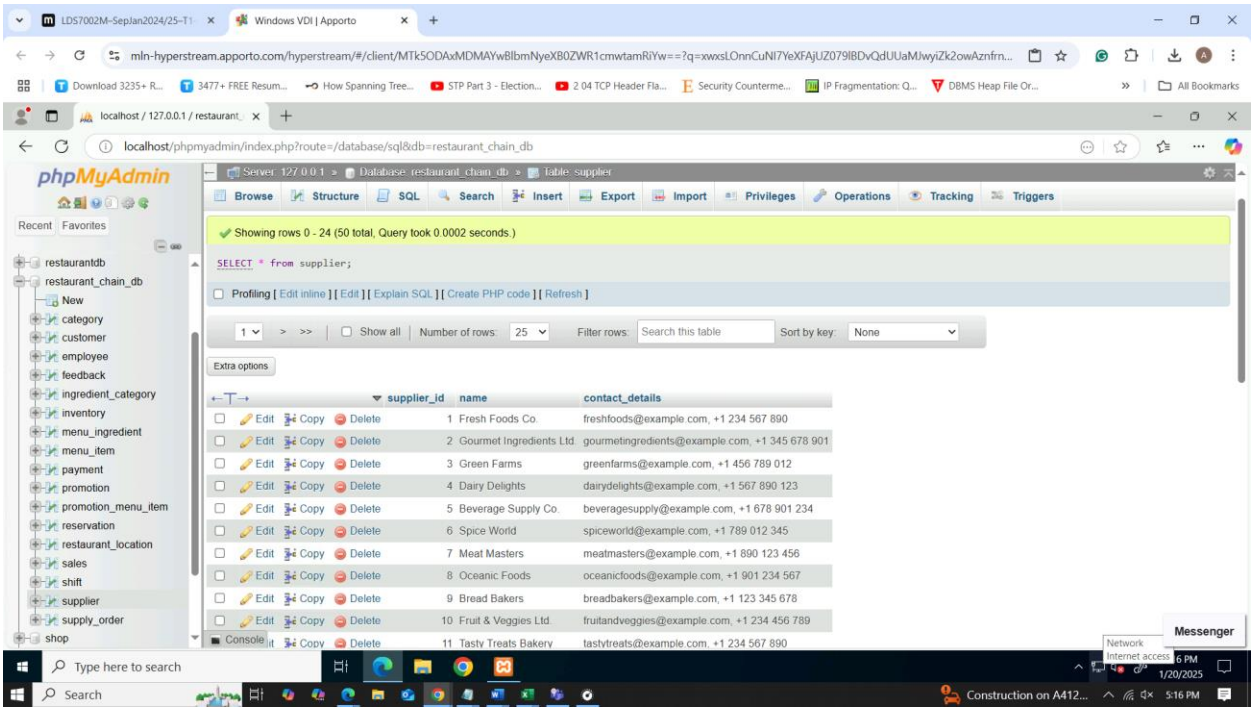


Figure 45: select query for supplier table

- Supply Order

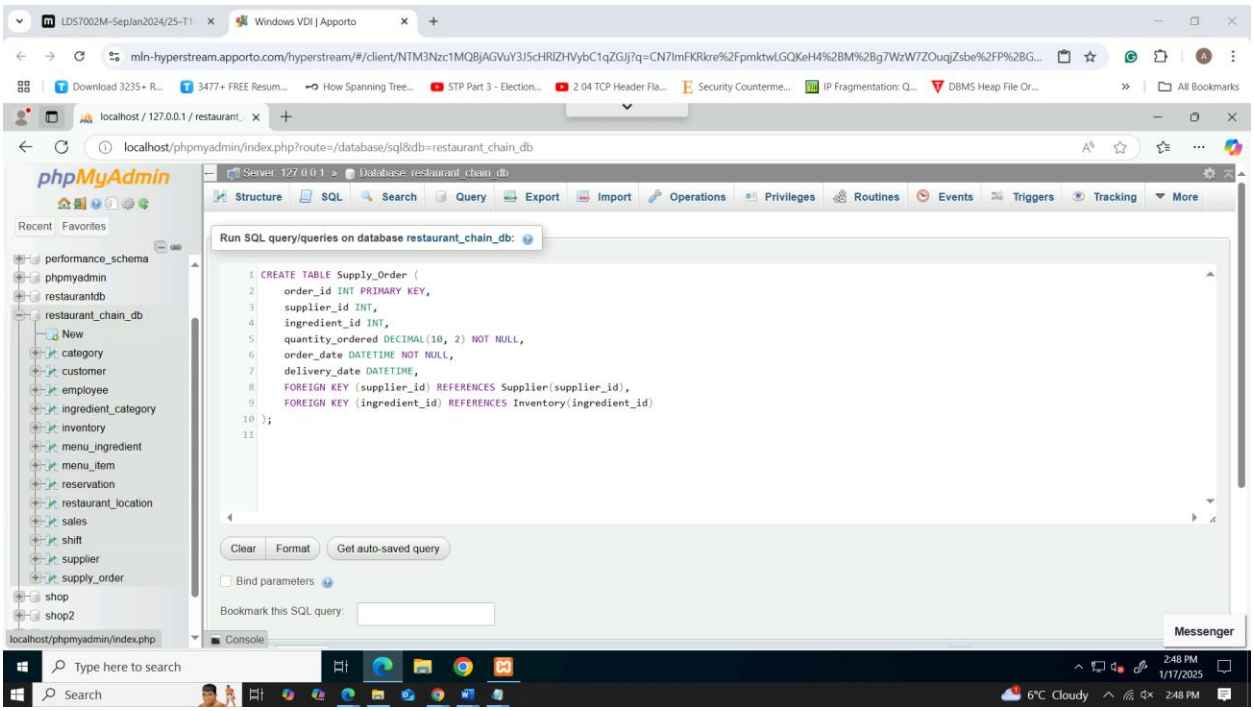


Figure 46: create table query of supply order

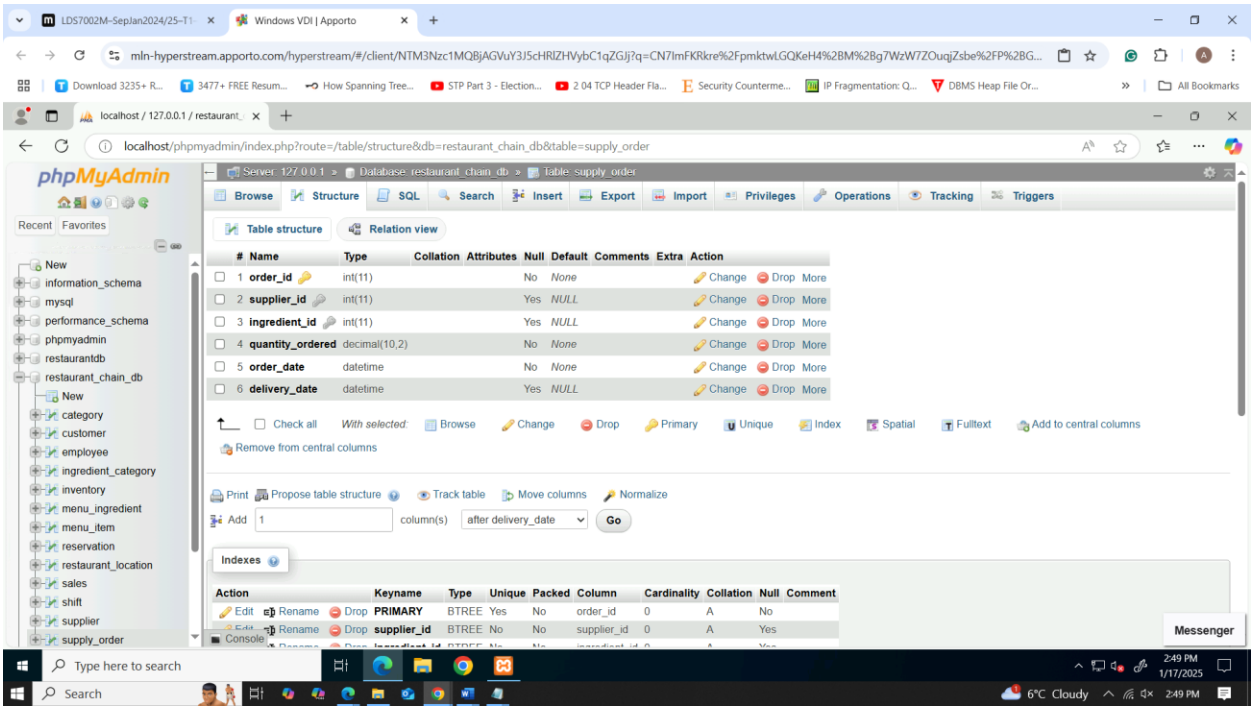


Figure 47:structure of supply order table

The Supply_Order table tracks supplier orders for ingredients with six columns: The fields in the order table are Order_ID a unique key identifying each order, Supplier_ID a key to the supplier in the supplier table, Ingredient_ID key to a specific ingredient in the inventory table and quantity ordered which is a decimal that has to be required to store the quantity of the ingredient that was ordered, Order_date which is a required datetime for the date the order was placed, and delivery date being an optional datetime that would store the expected or This design help in tracking the orders relating to supplies, relation between ingredients and suppliers and order details.

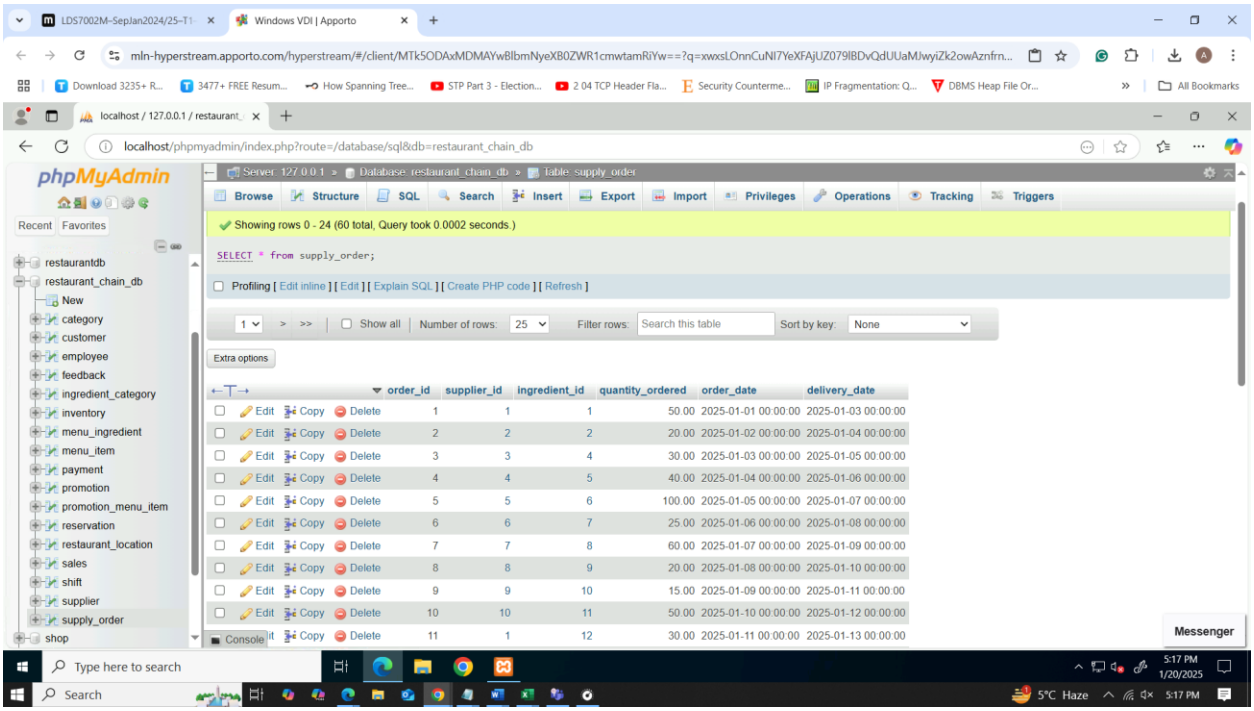


Figure 48: select query for supply order table

- Payment

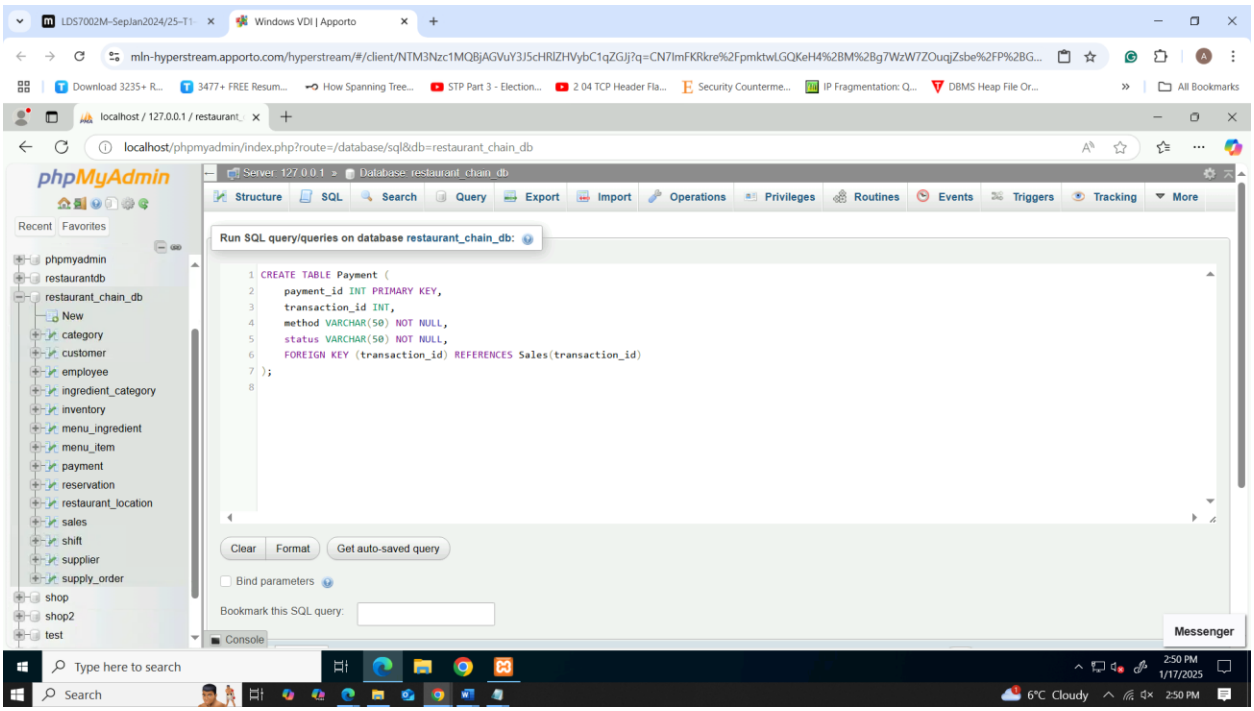


Figure 49: create table query of payment

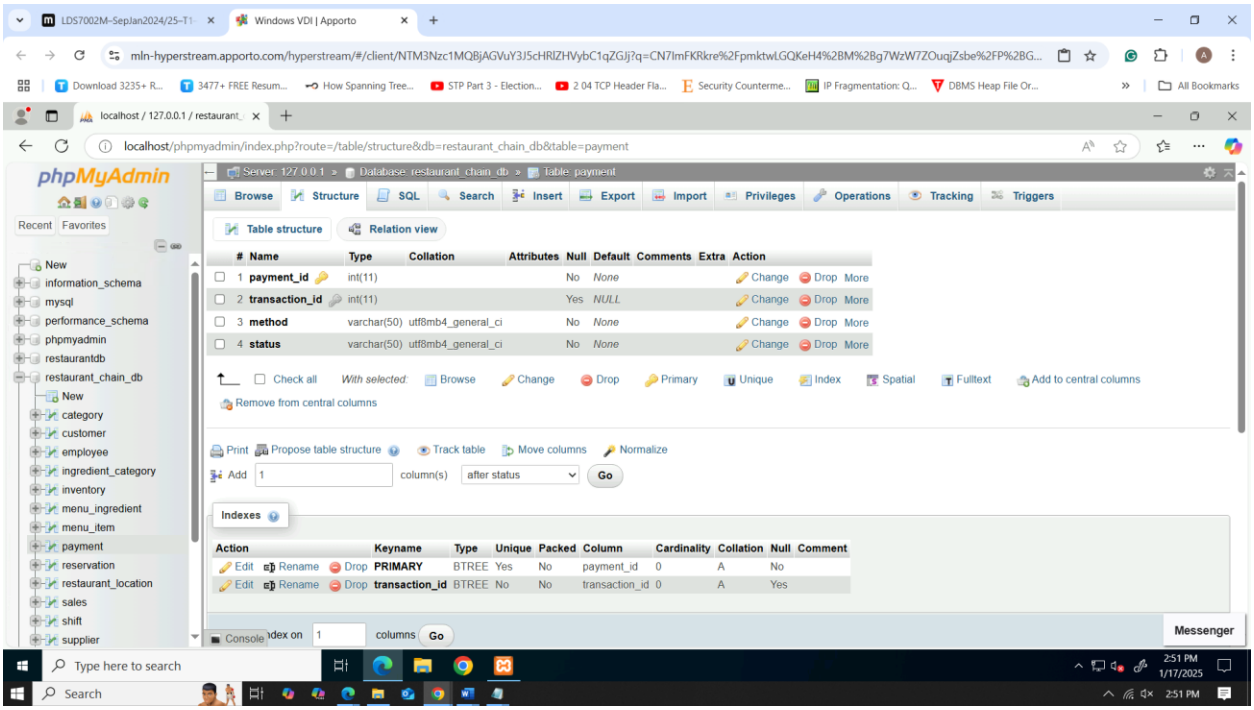


Figure 50: structure of payment table

The Payment table records payment details for sales transactions with four columns: payment_id respectively; transaction_id respectively; method respectively, which must be a string of no more than 50 characters describing the type of payment, for example “credit card,” “cash,” or “online payment;” and status respectively, which also must be a string of no more than 50 characters describing the status of the payment, for instance “completed,” “pending,” or “failed.” This table facilitates adequate payment, relating it to transactions while displaying core information for payment processing and analysis.

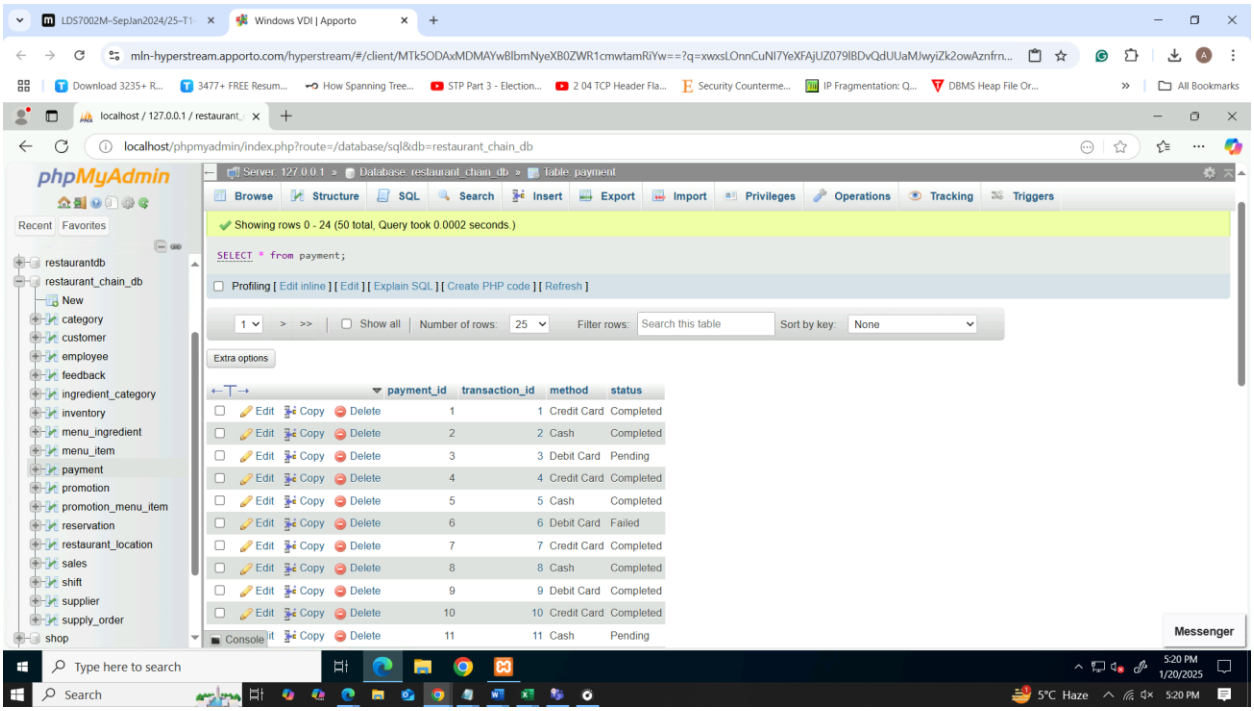


Figure 51: select query for payment table

- Feedback

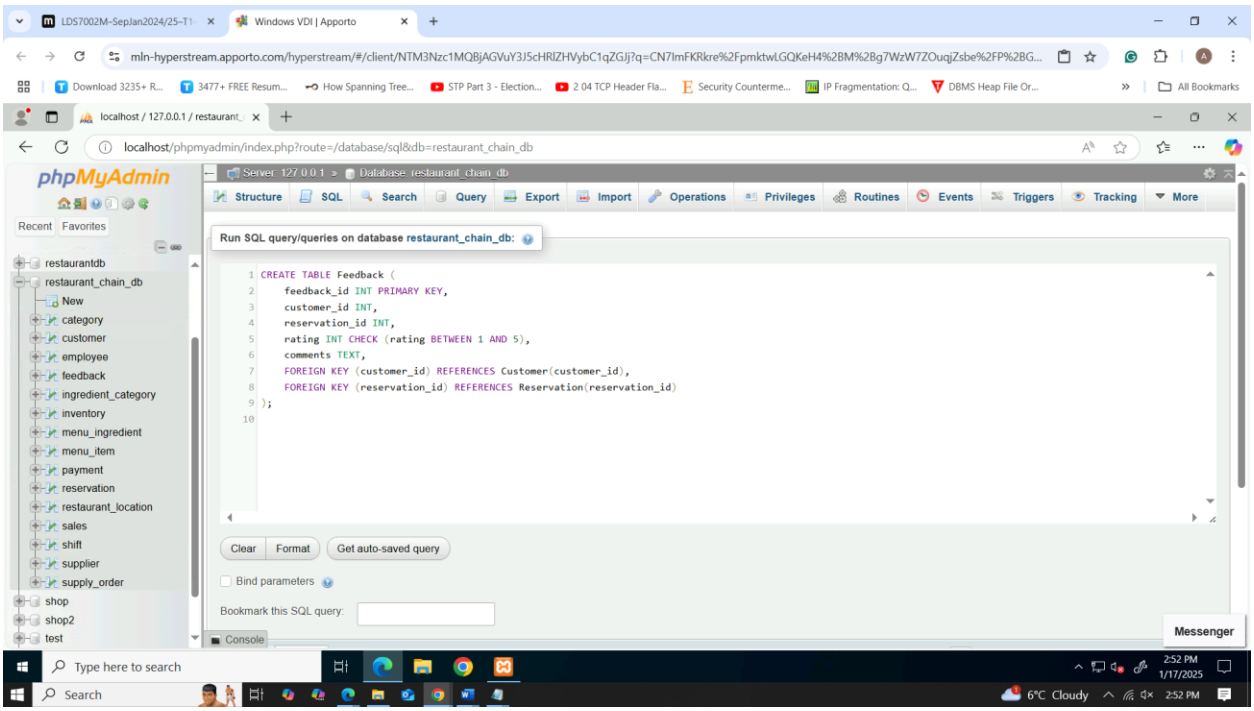


Figure 52: create table query of feedback

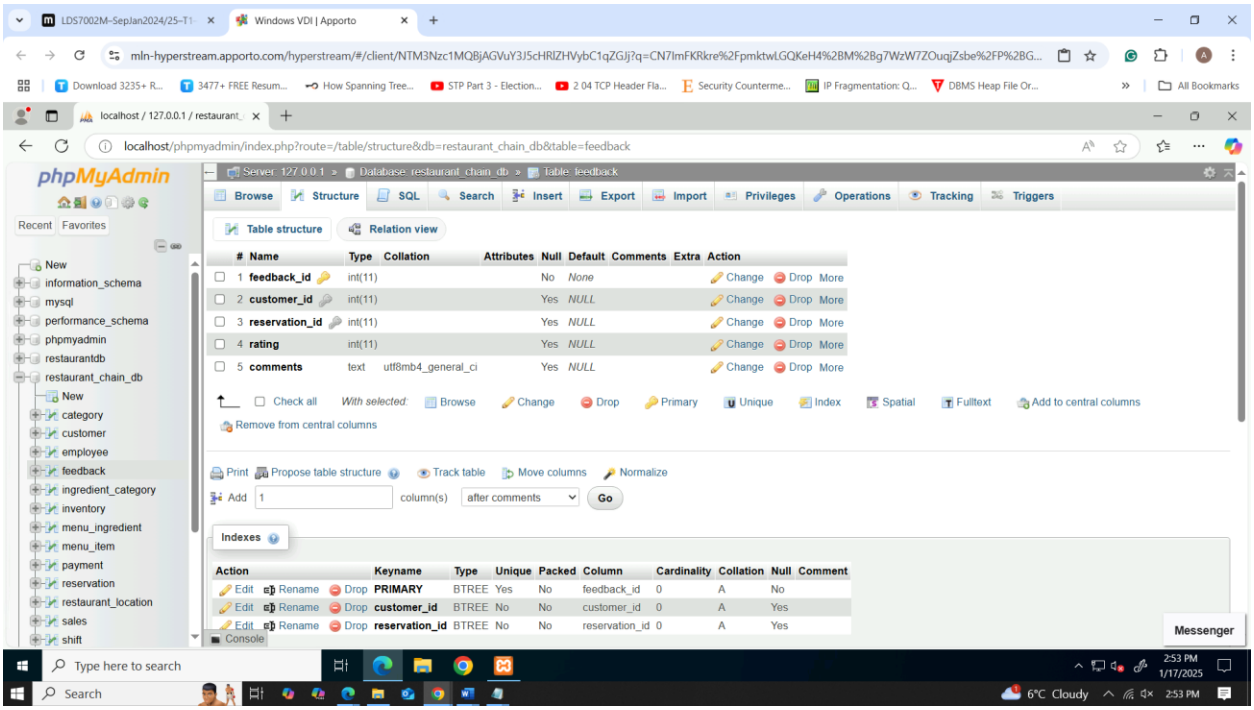


Figure 53: structure of feedback table

The Feedback table collects customer feedback on dining experiences with five columns: feedback_id (unique identifier for each feedback entry), customer_id (feedback connected with the client in table Customer), reservation_id (feedback connected with a specific reservation in table Reservation), rating (integer field with values from CHECK constraint between 1 and 5 – customer satisfaction), and comments (Text field for further notes or comments. With this table, collecting feedback regarding customers and reservations becomes easier to collect and analyze probable feedback for the evaluation of services and improvements of the dining section.

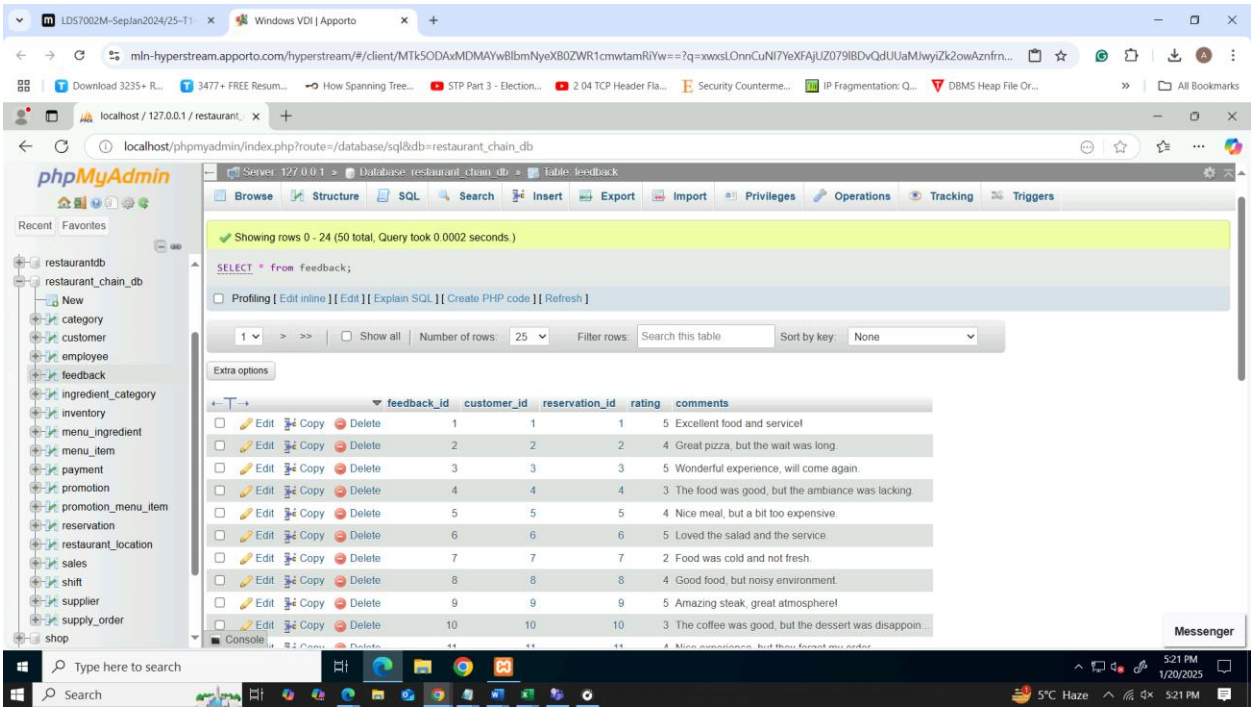


Figure 54: select query for feedback table

- Promotion

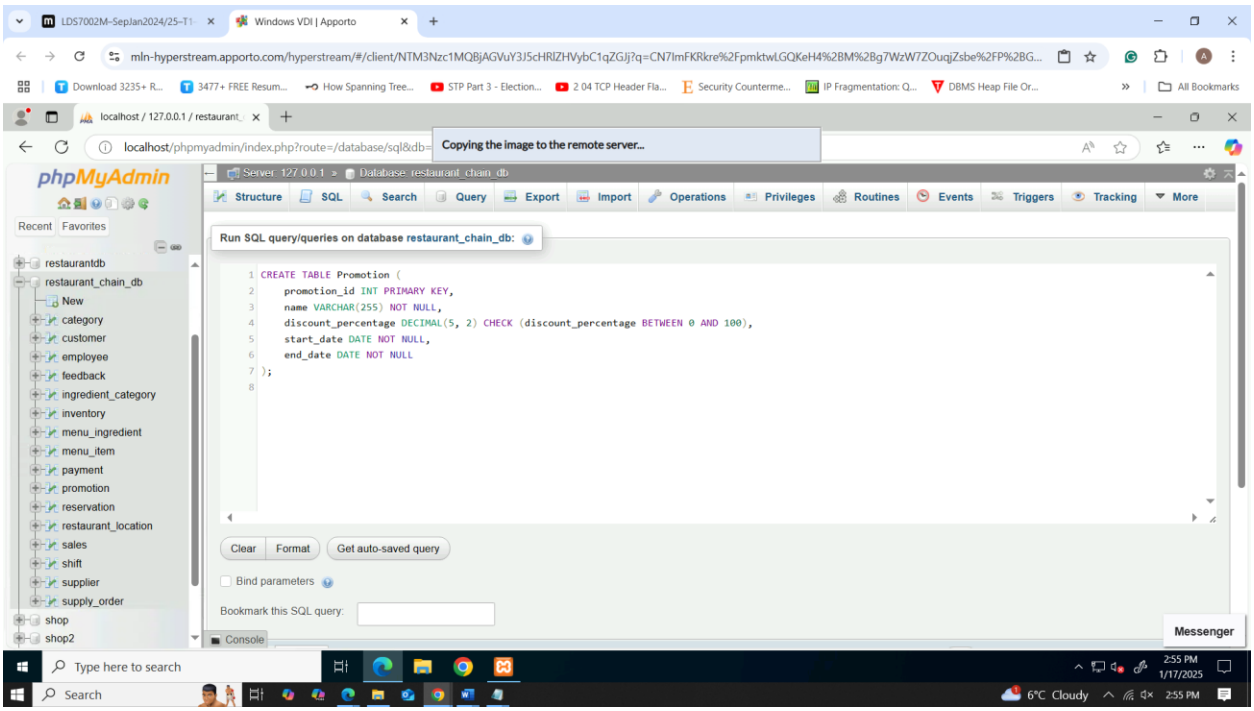


Figure 55: create table query of promotion

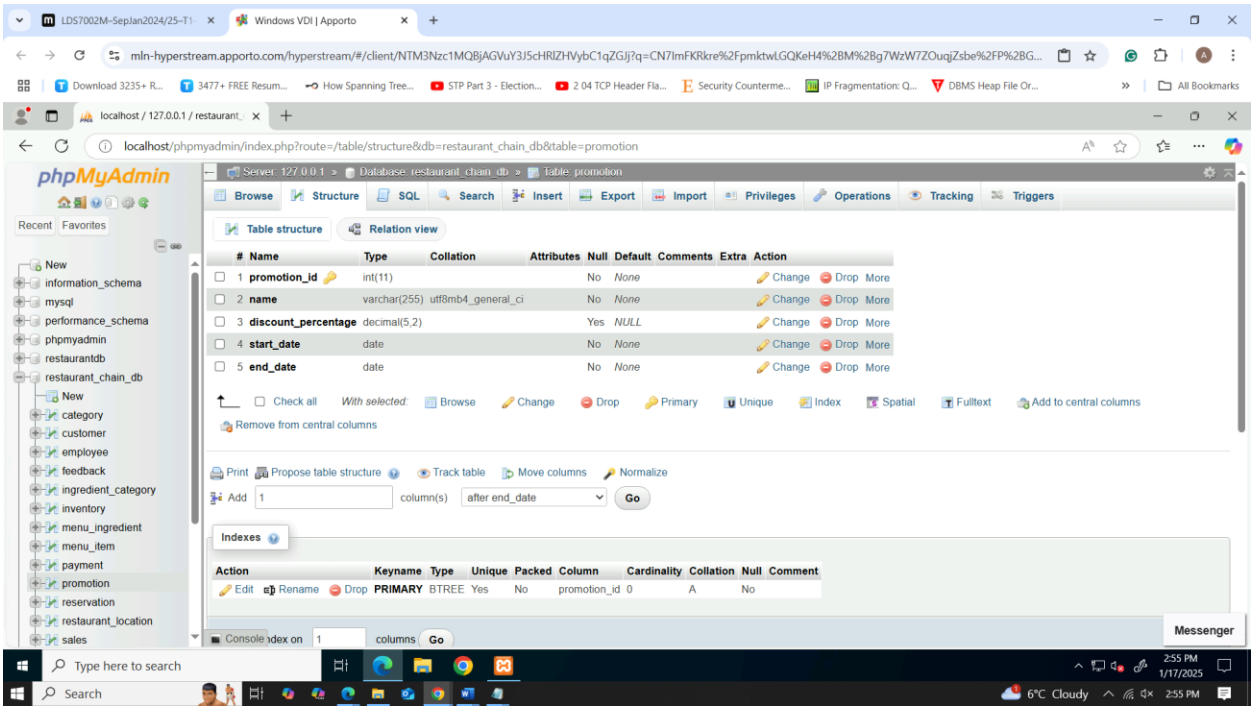


Figure 56: structure of promotion table

The Promotion table stores details about promotional offers with five columns: promotion_id (integer, primary key uniquely identifying each promotion), name (string of up to 255 characters, which required field used for storing the name or title of the promotion), discount_percentage (DECIMAL(5 2), which is the required field storing the percentage of discount for the products and it has a check constraint to ensure that the values entered are between 0 and 100), start_date (required date field that which denote the start This table helps in manning and monitoring promotions by verifying the proper periods of discounts, and in preserving crucial promotional data for usage.

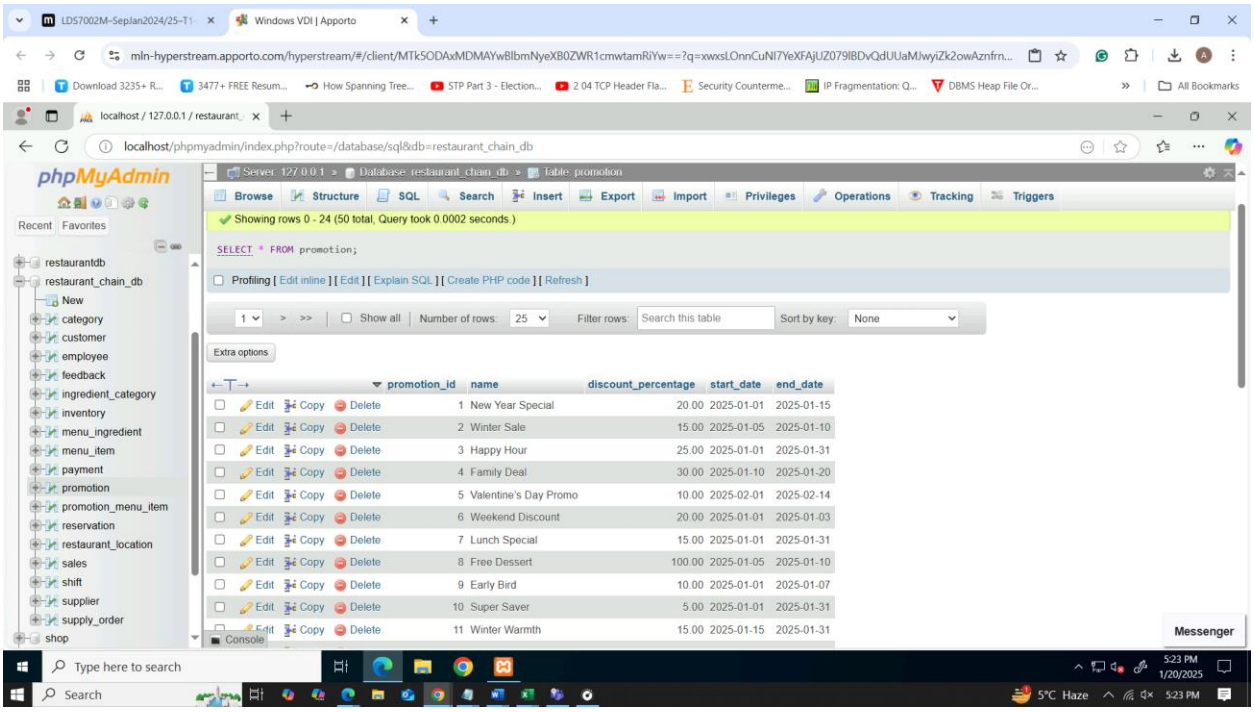


Figure 57: select query for promotion table

- Junction Table

A junction table is used to manage many-to-many (M:(N) relational databases structures because direct relationships go against the concept of normalization. The latter saves repetitional instances of the same relationship, thus making the database normalized. Also, it offers than-kadox for attributes related to the relationship only, such as periodic discounts in promotions. A junction table also has another advantage of being able to easily include or exclude relationships. For instance when considering the entities of promotions and menu items, one can create a junction table known as Promotion_Menu_Item which relates the two tables without a problem of atomicity or having the two entities repeat the same data through the existence of a foreign key or two.

Menu Item and Promotion junction table

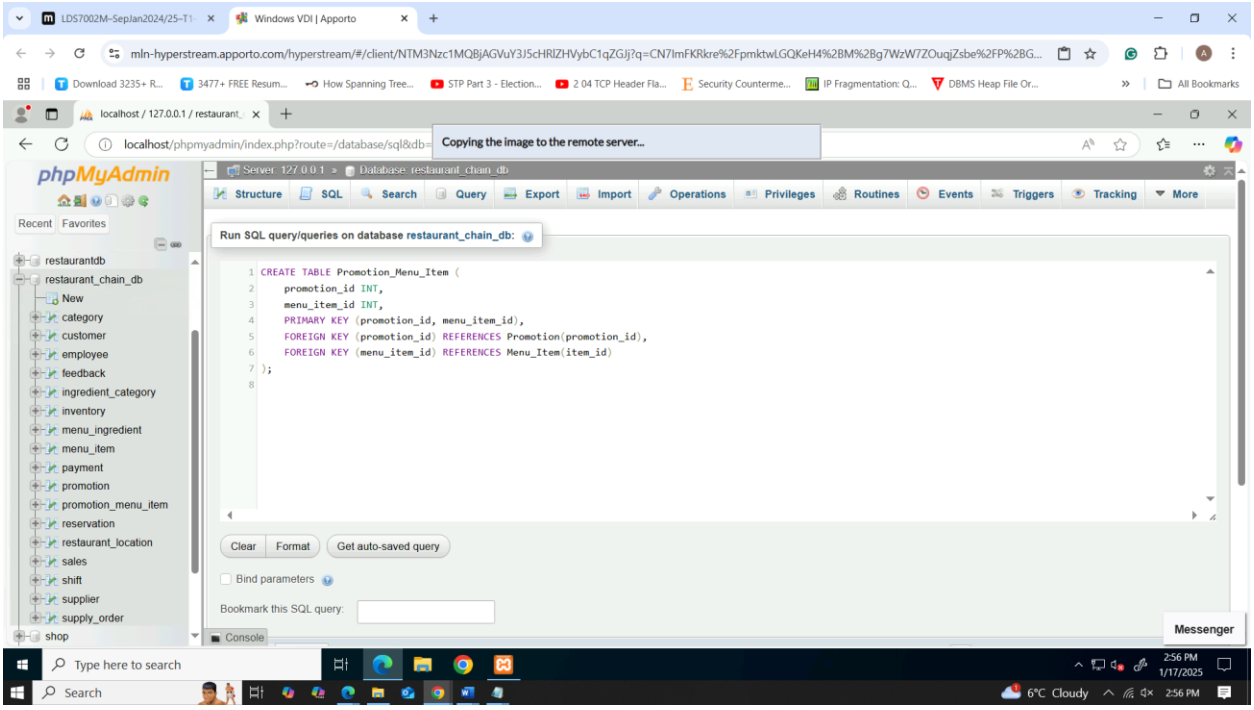


Figure 58: create table query of promotion menu item

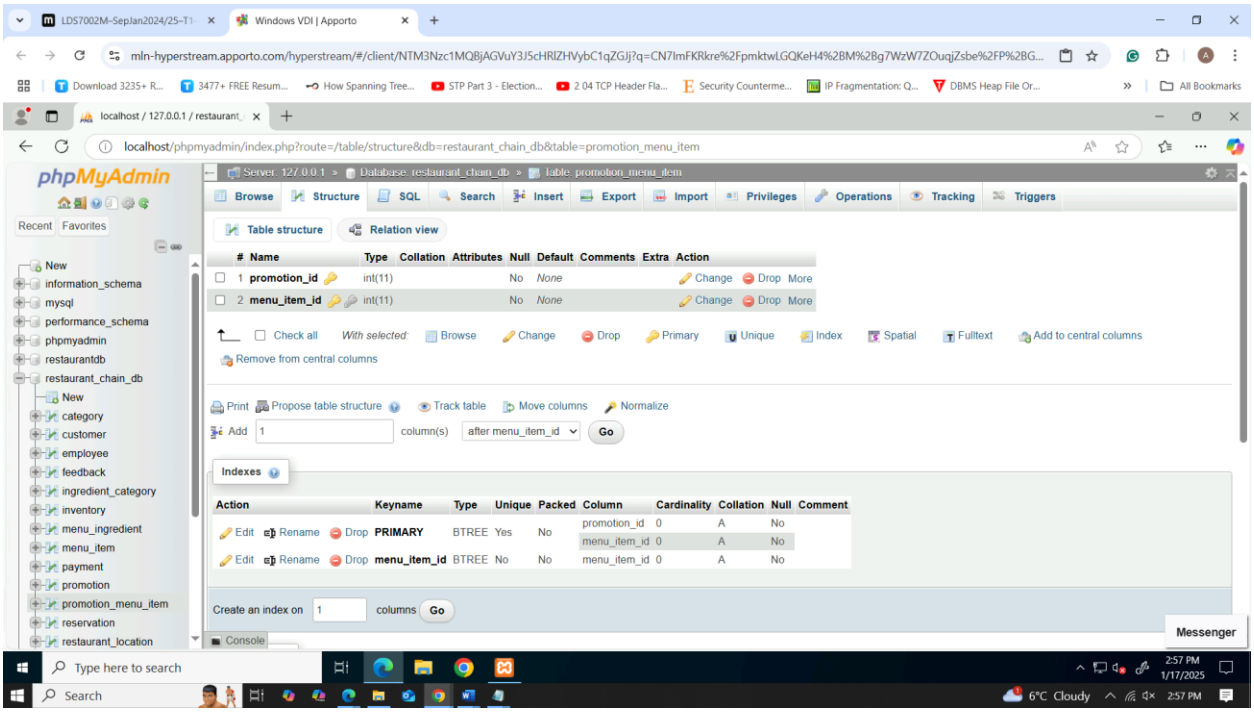


Figure 59: structure of promotion menu item table

Getting back to the description of the given shortage, let me note that Promotion_Menu_Item is a junction table that connects promotions with particular menu items. It has two foreign key columns: promotion_id in relation to the promotion_id in the Promotion table, and menu_item_id in relation with the item_id in the Menu_Item table. These foreign keys define relations between promotions and the products that belong to them, so, a promotion can be applied to definite plates or products.

Promotion ID and menu item ID are used as a composite key and therefore it is the primary key of the table. This helps in the uniqueness of the menu items so that in the promotion only one item is selected to be promoted. This structure makes it easy to map several items in the menu to a single promotion, help the restaurant group offer and keep accurate records of promotions and their relative menu items where necessary.

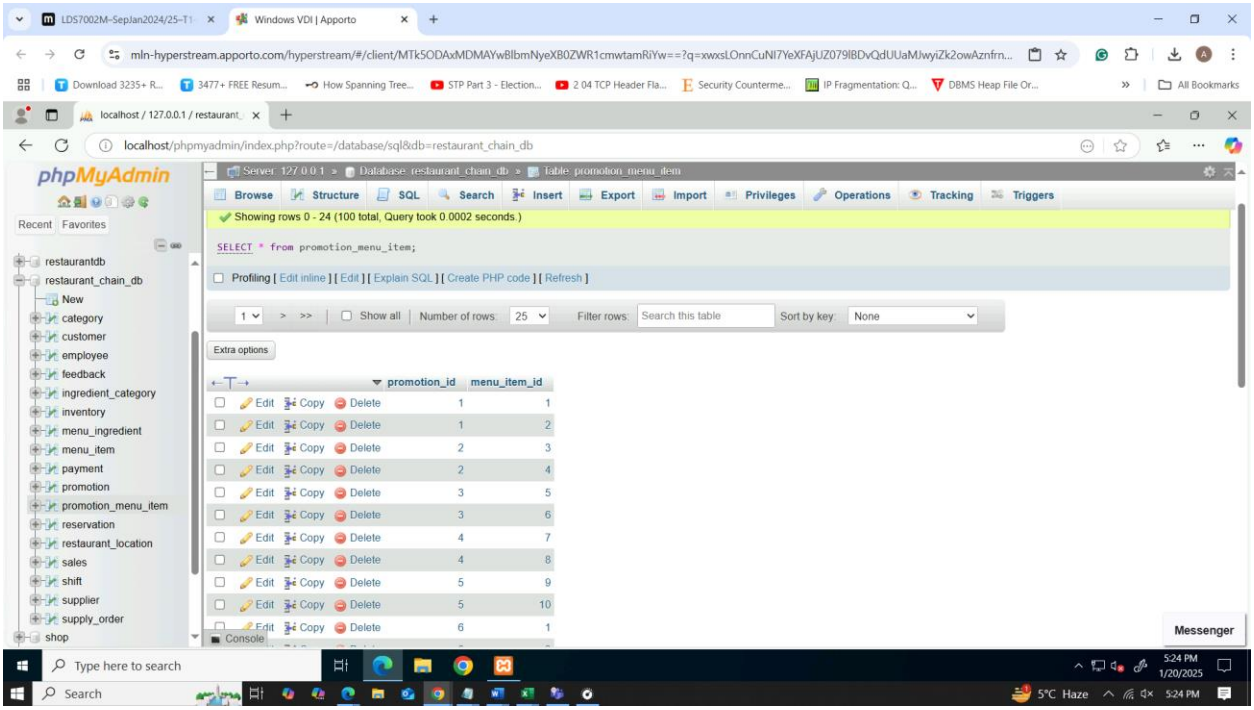


Figure 60: select query for junction table

4 Insert values to the database tables

- Customer

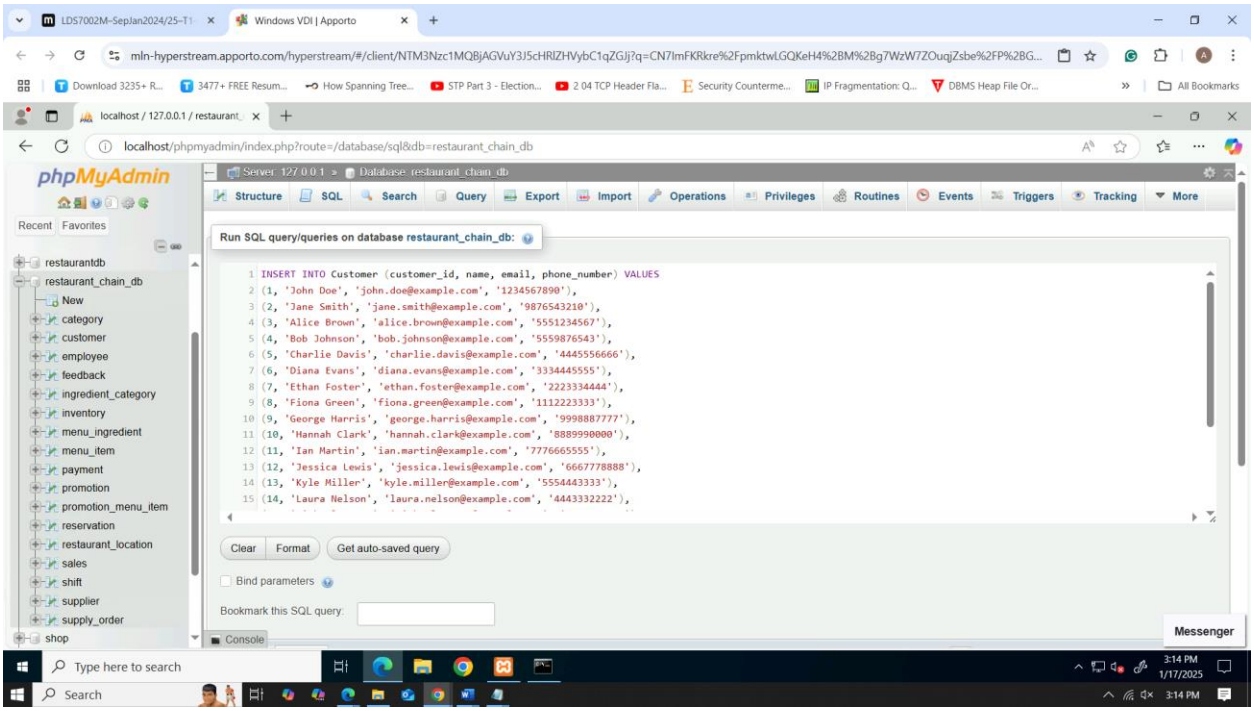


Figure 61: insert value to the customer table

- Category

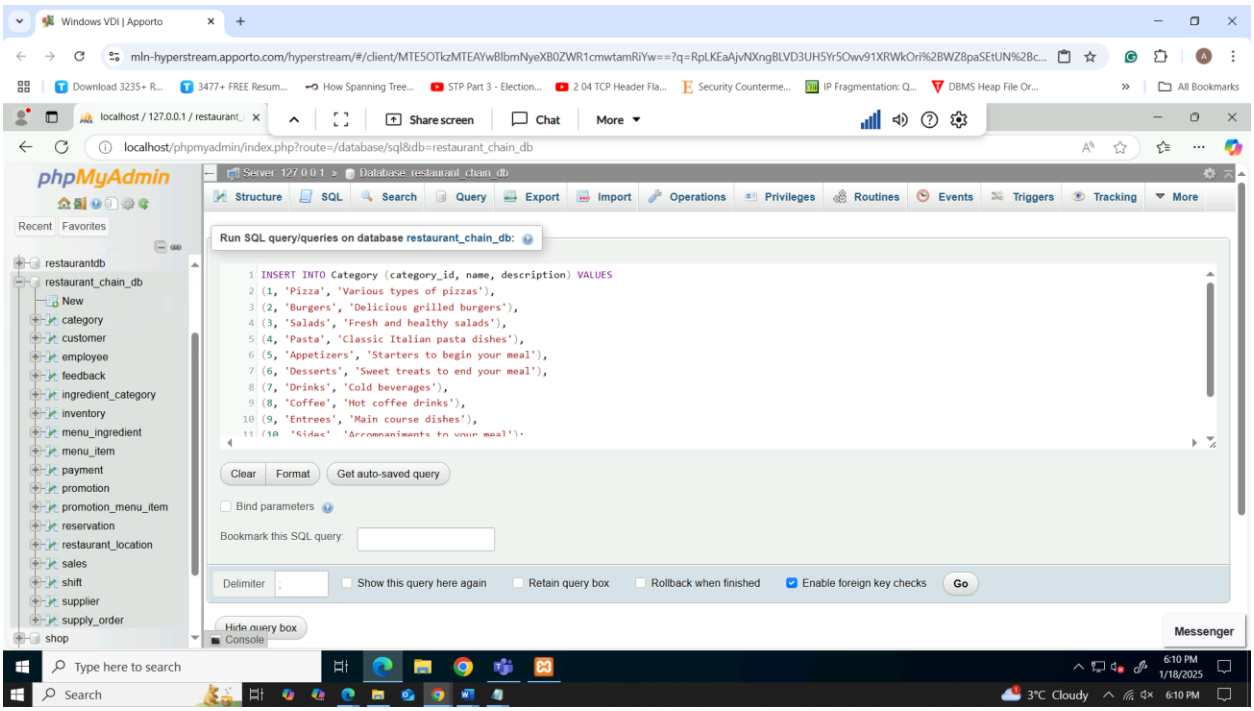


Figure 62: insert values to the category table

- Menu items

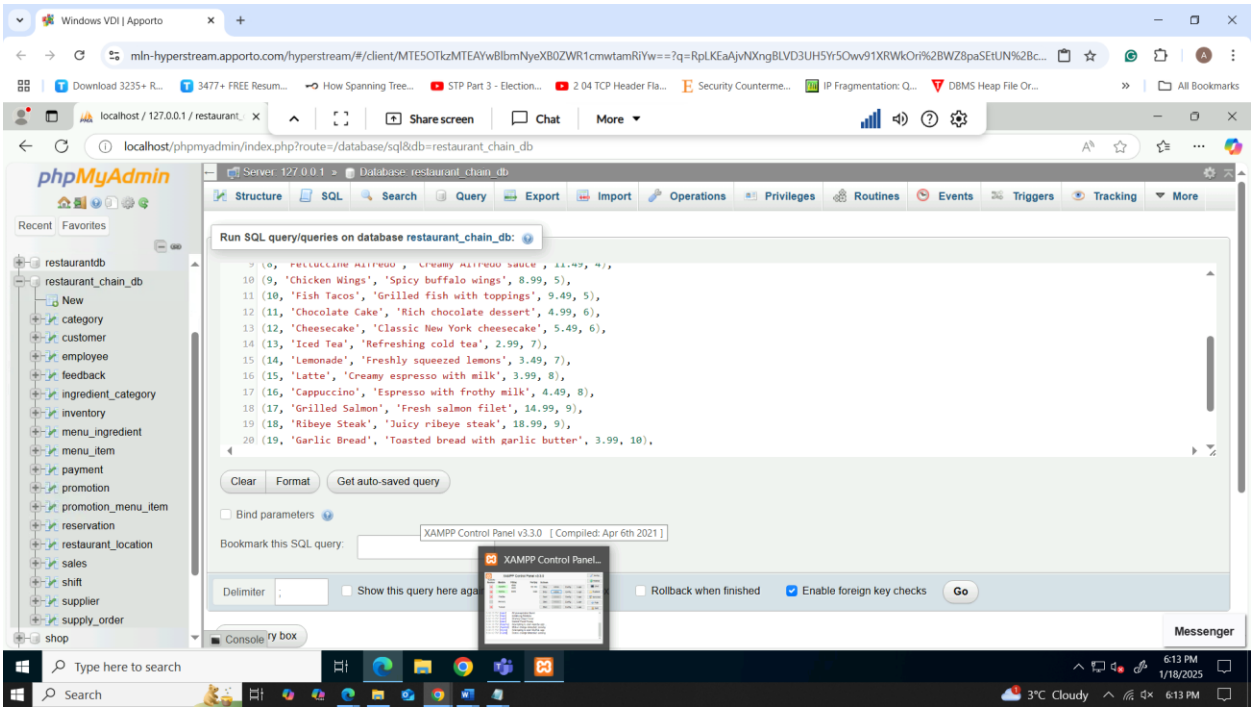


Figure 63: insert values to the menu item

- Restaurant location

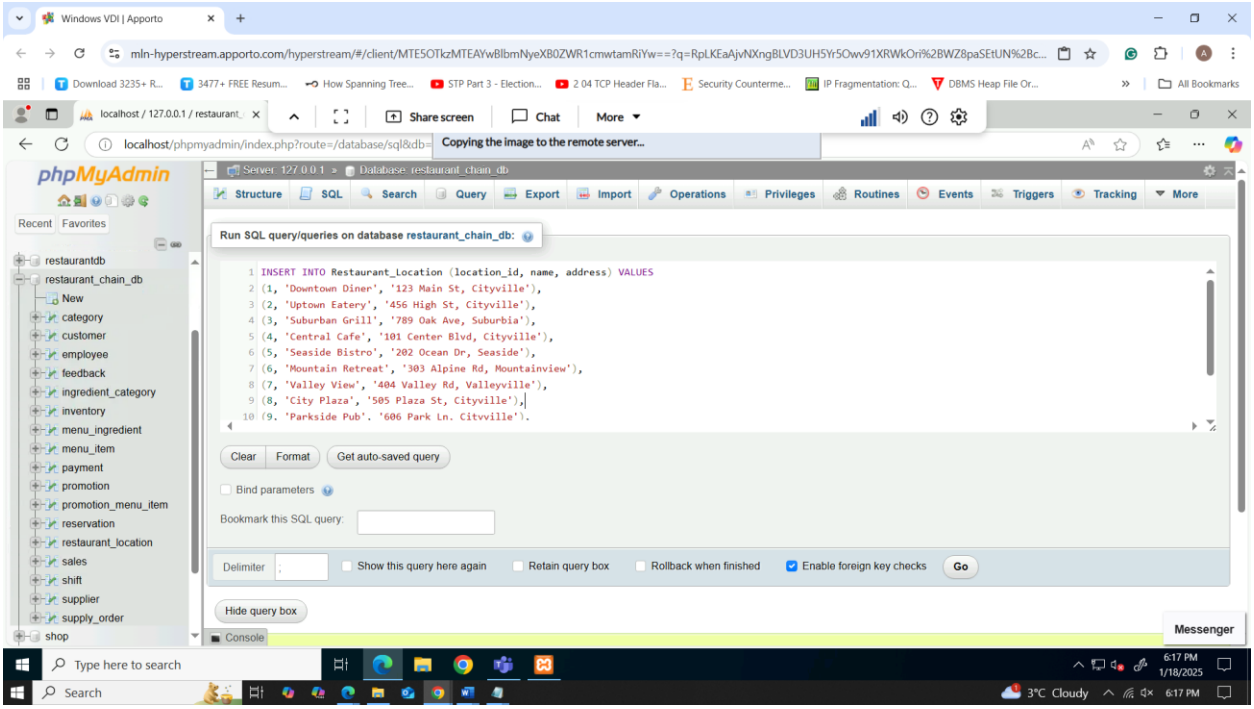


Figure 64: insert values to the restaurant location

- Reservation

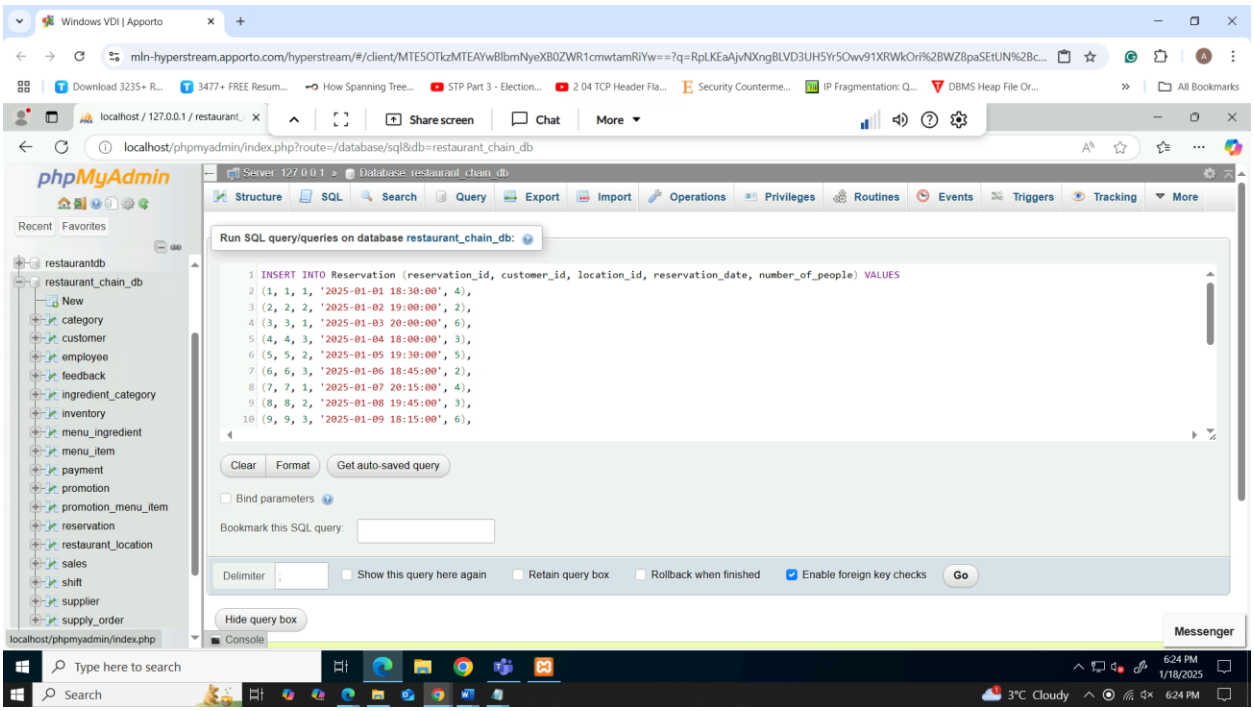


Figure 65: insert values to the reservation

- Sales

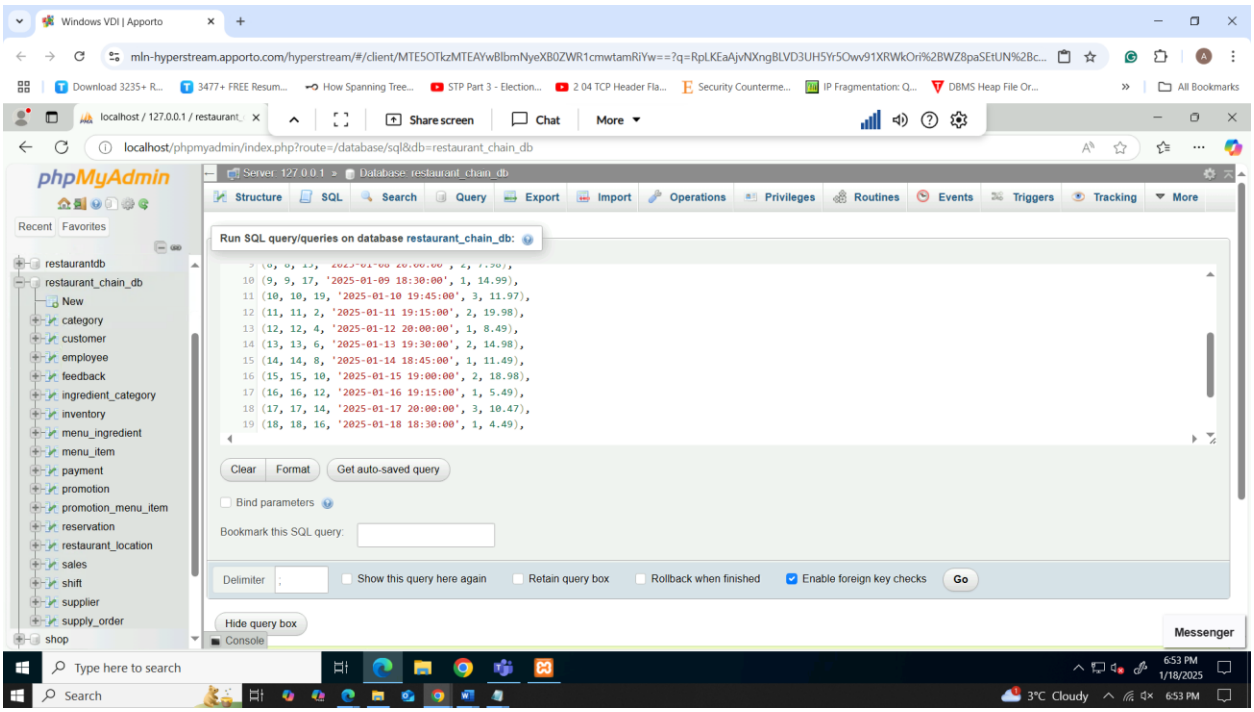


Figure 66: insert values to the sales table

- Ingredient category

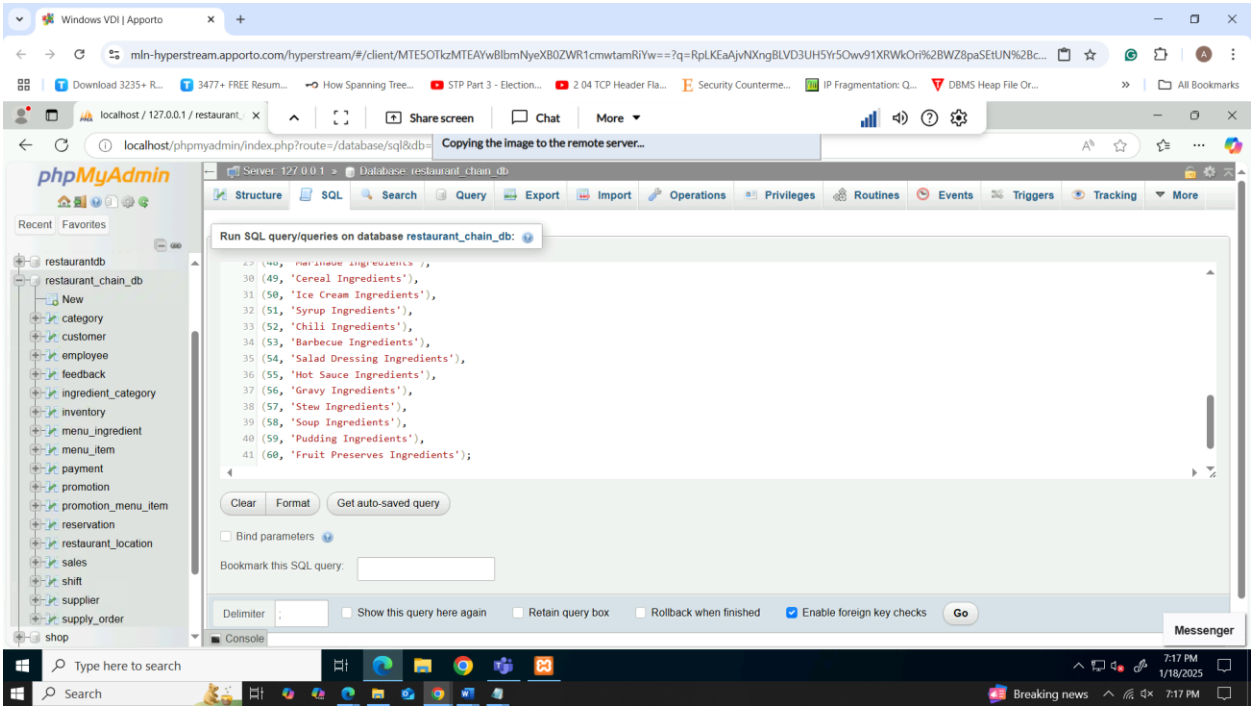


Figure 67: insert values to the ingredient category

- Inventory

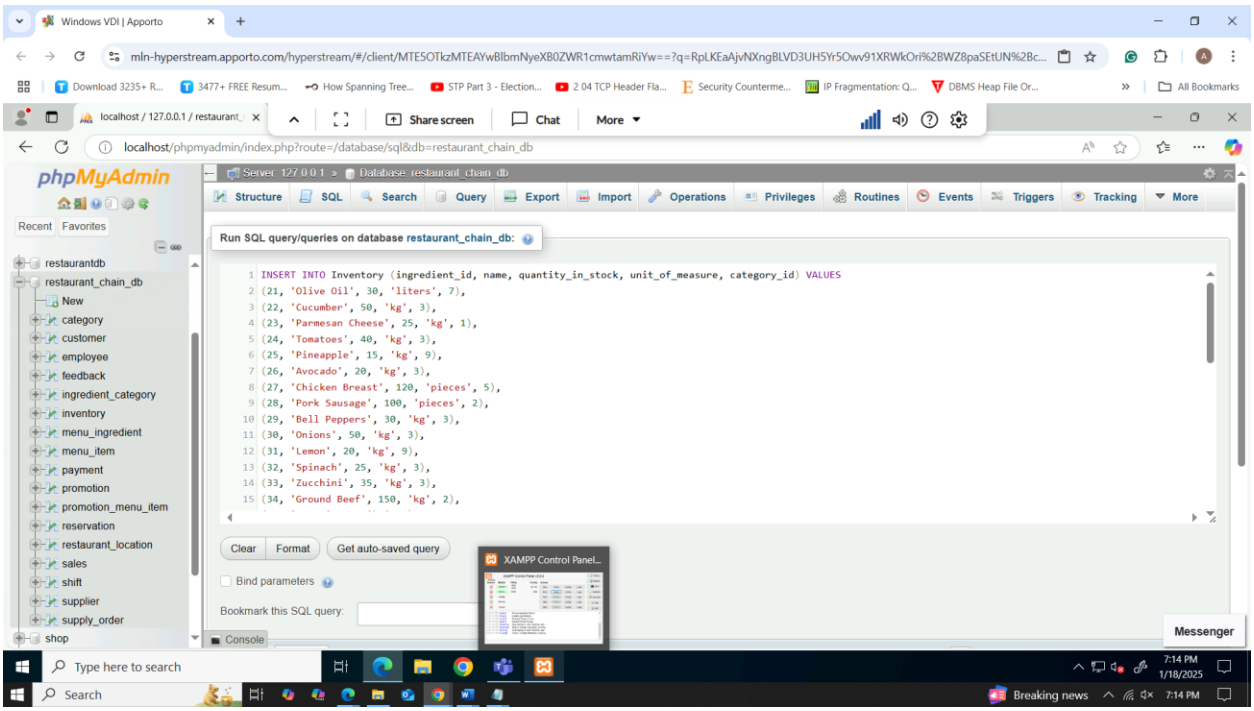


Figure 68: insert values to the inventory table

- Menu_ingredient

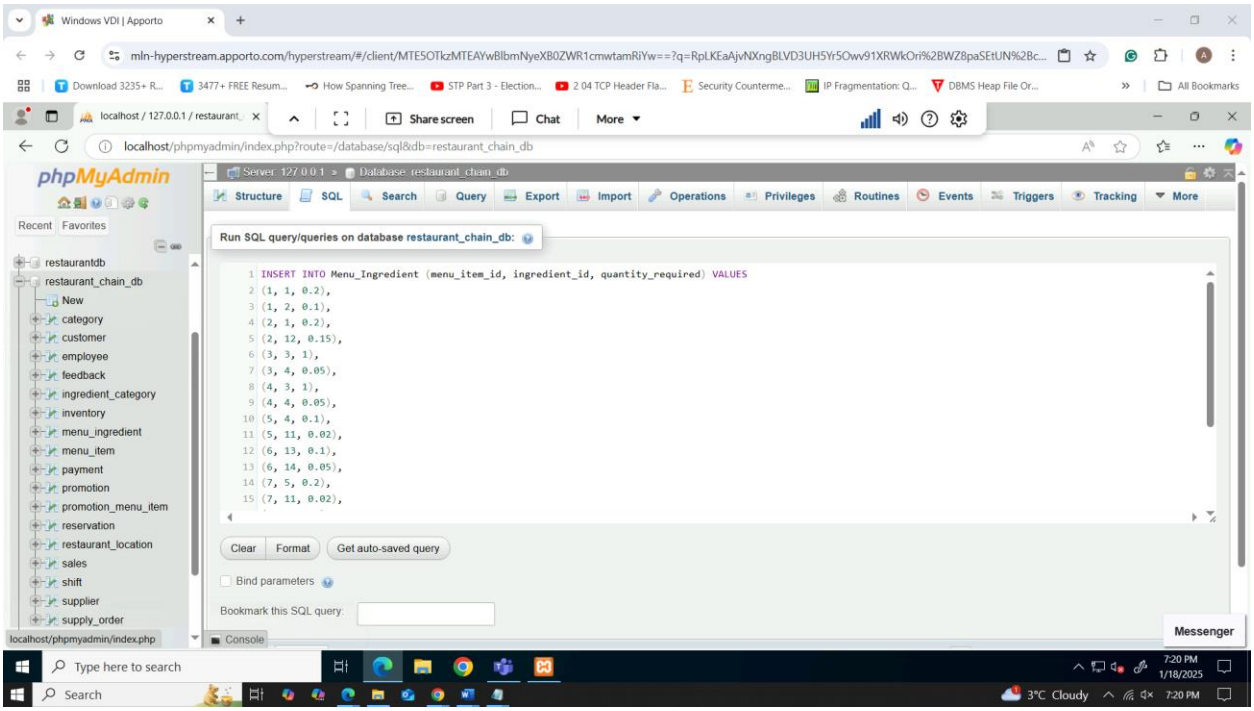


Figure 69: insert values to the menu ingredient table

- Employee

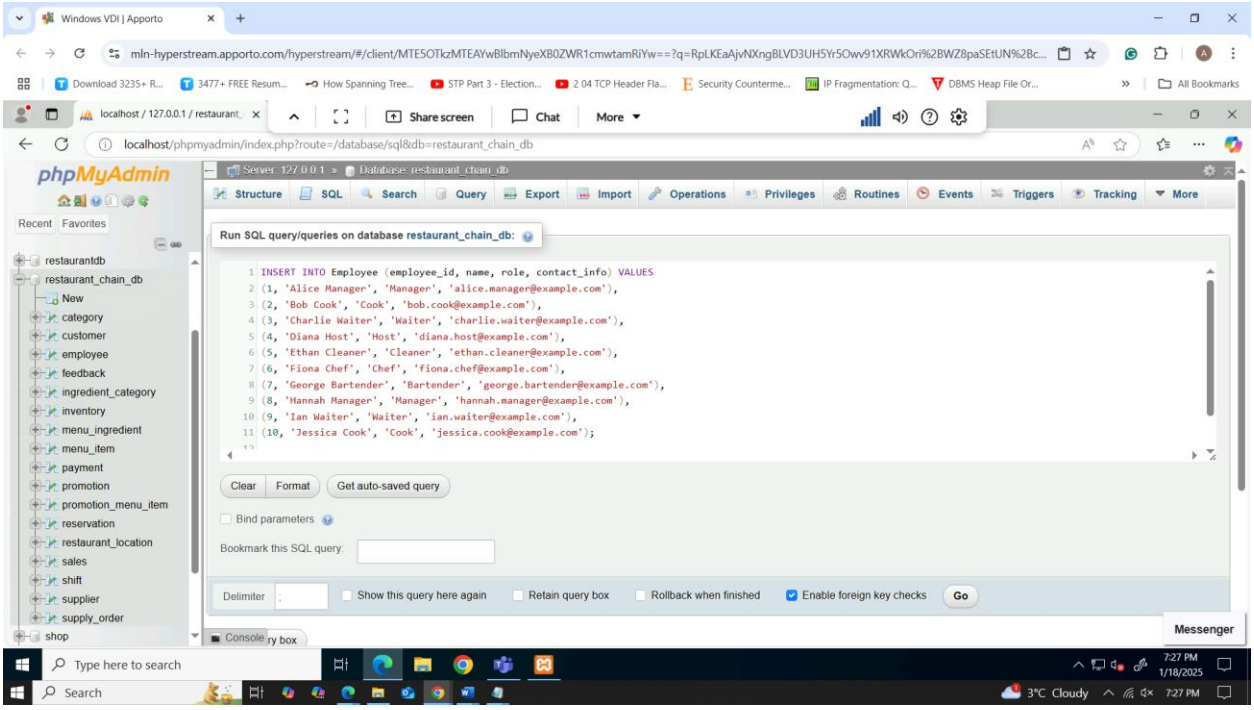


Figure 70: insert values to the employee table

- Shift

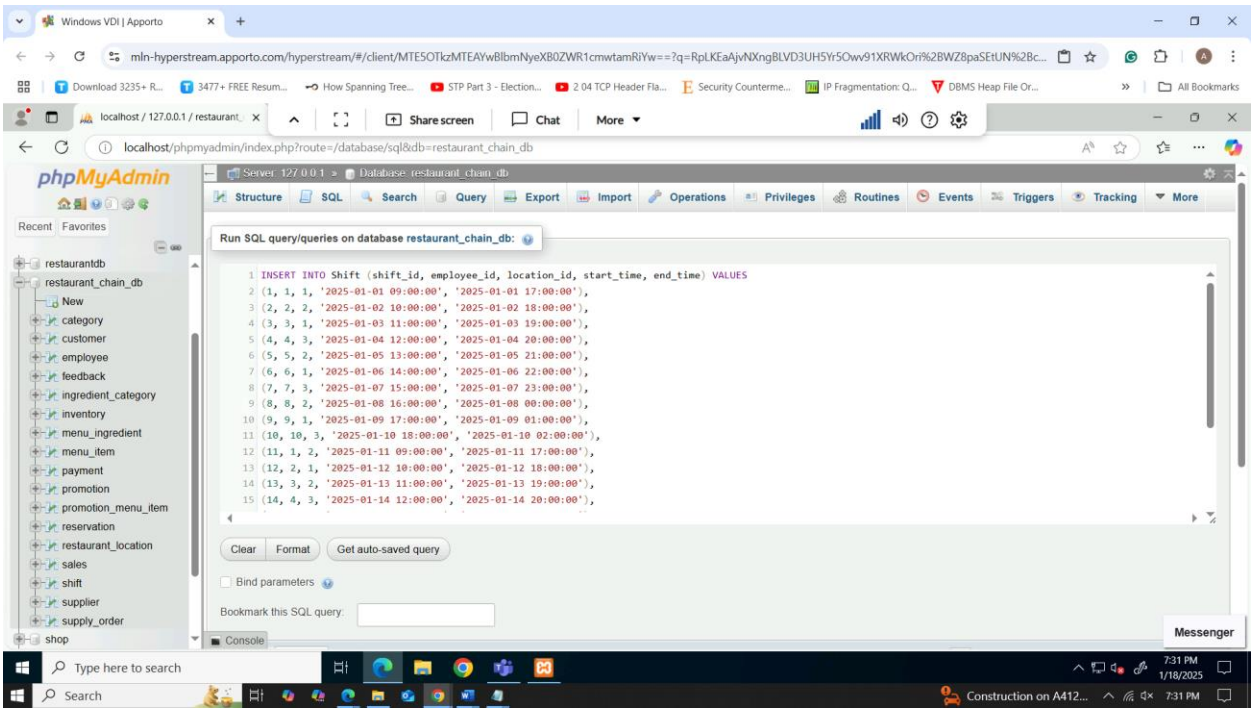


Figure 71: insert values to the shift table

- Supplier

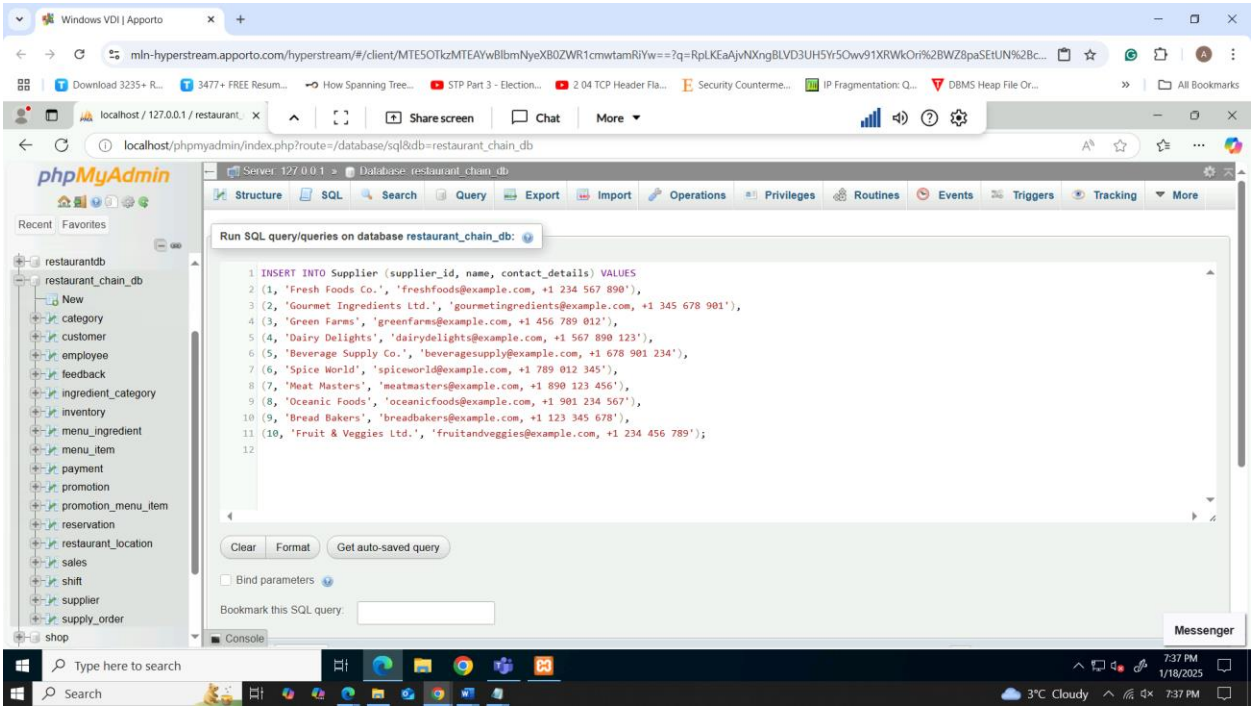


Figure 72: insert values to the supplier table

- Supply_order

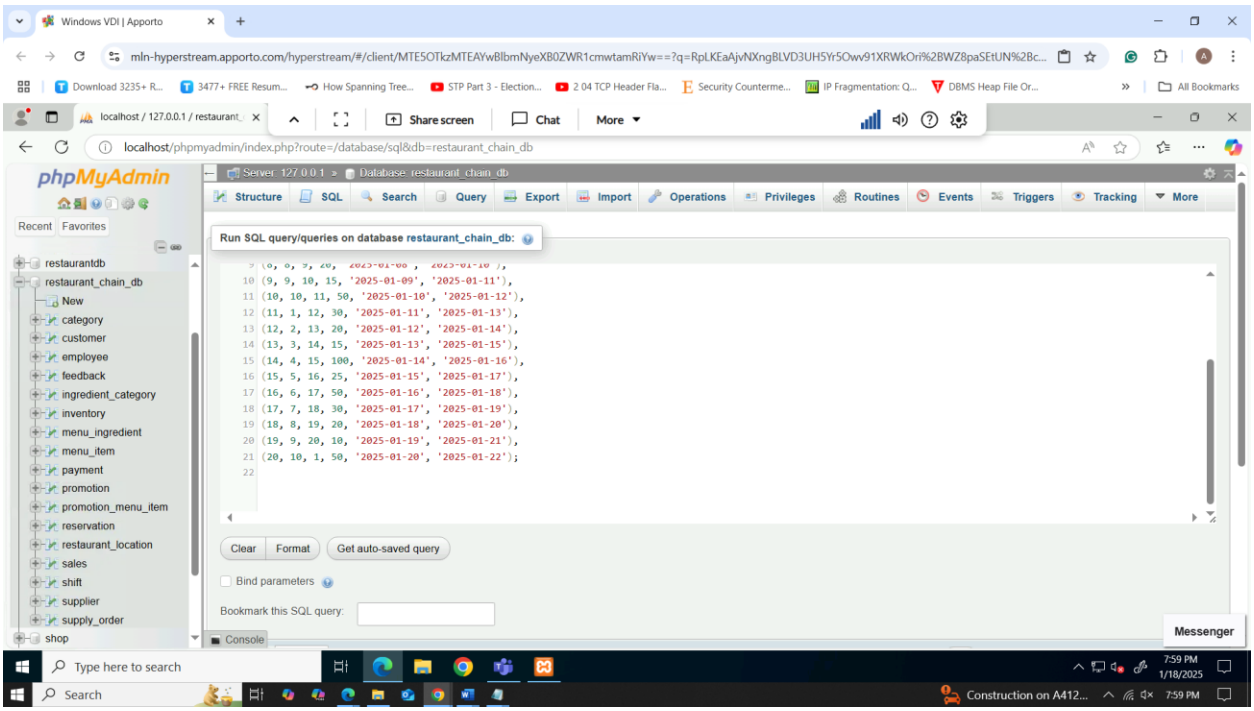


Figure 73: insert values to the supply order table

- Payment

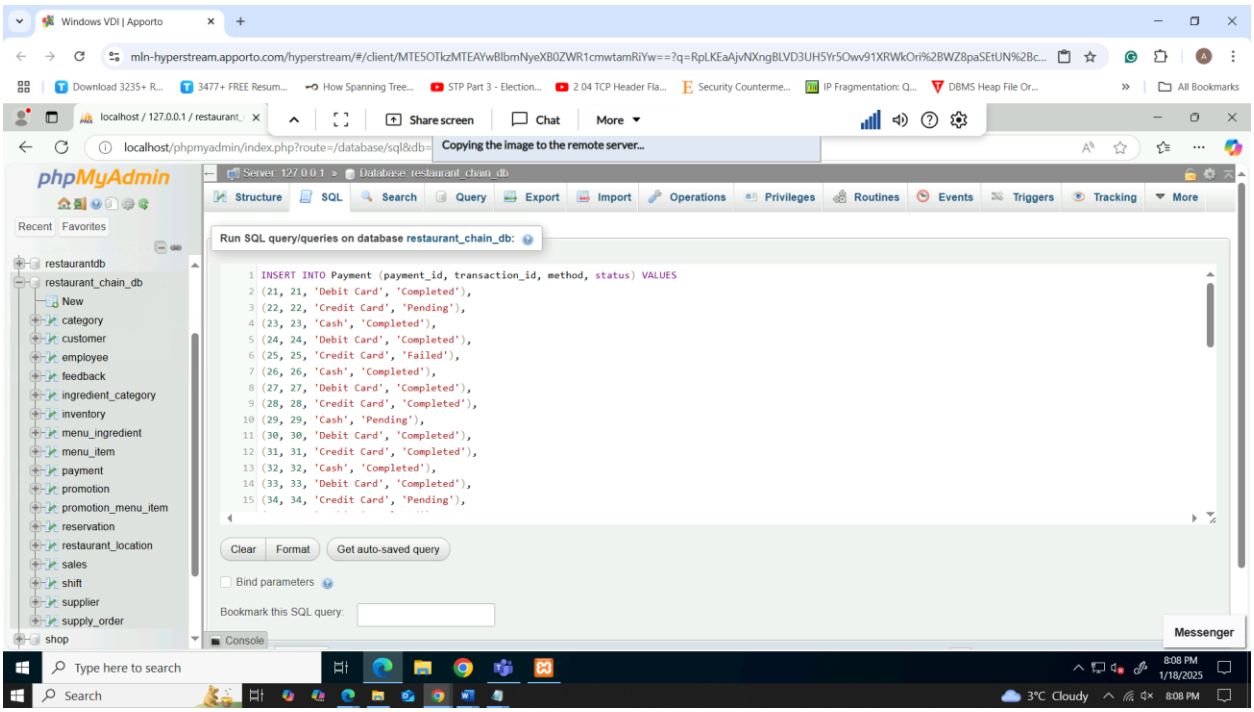


Figure 74: insert values to the payment table

- Feedback

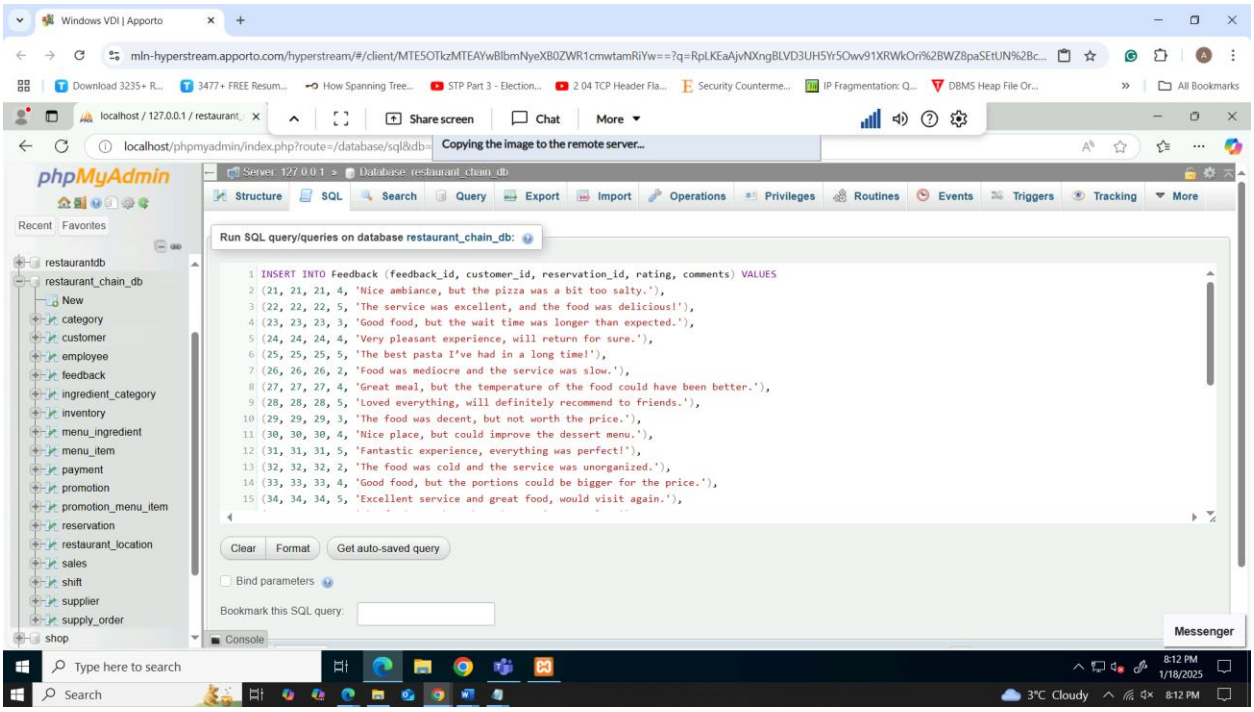


Figure 75: insert values to the feedback table

- Promotion

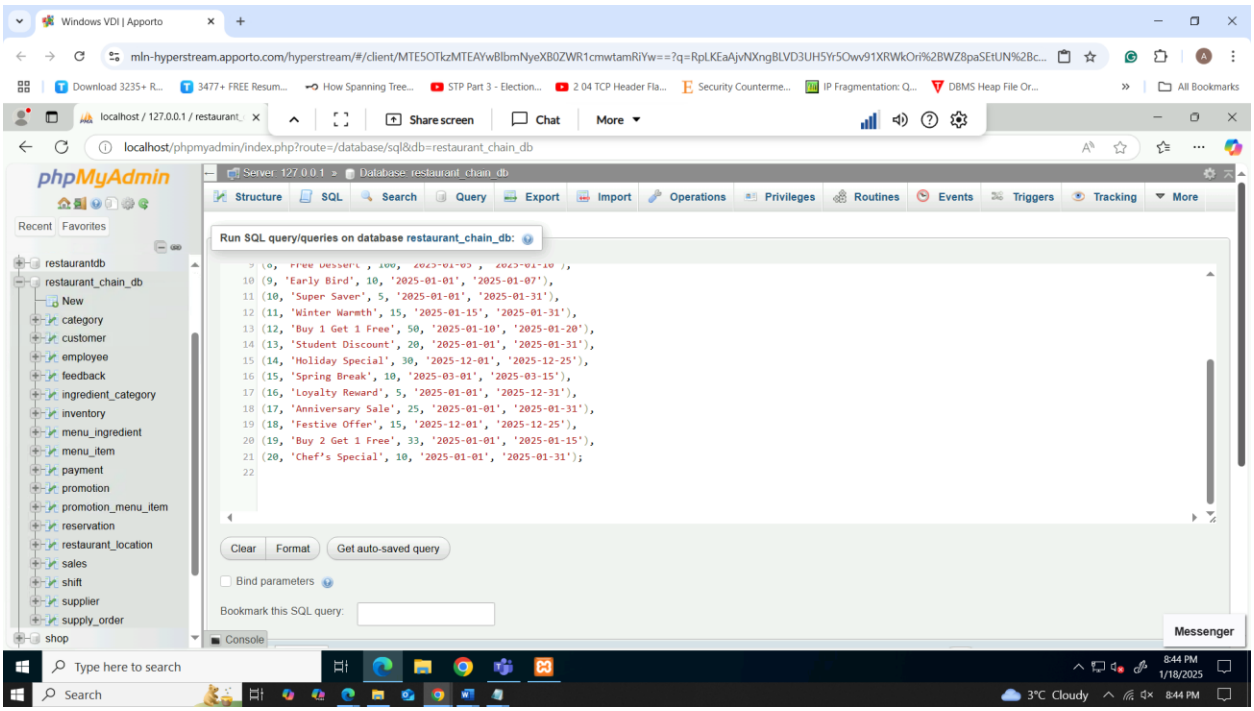


Figure 76: insert values to the promotion table

- Promotion_menu_item (junction table)

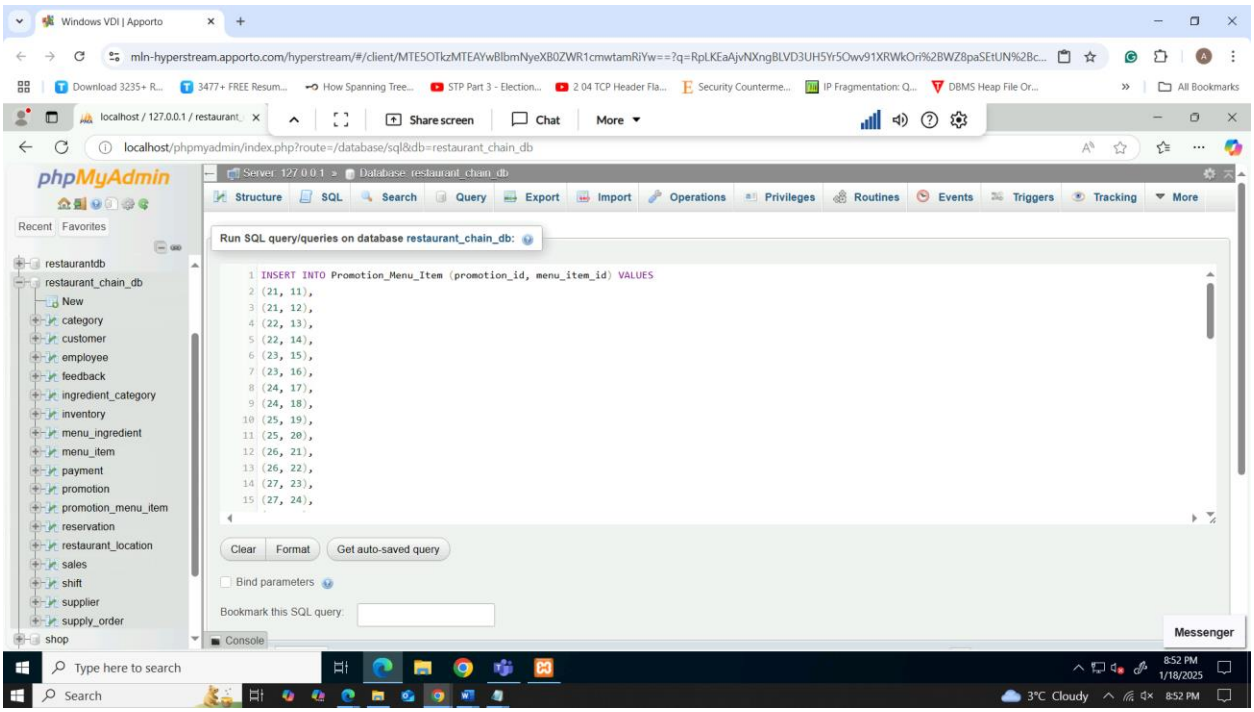


Figure 77: insert values to the promotion menu item table

5 Question and answers

- Total menu discipline for the different menu categories over a given date period

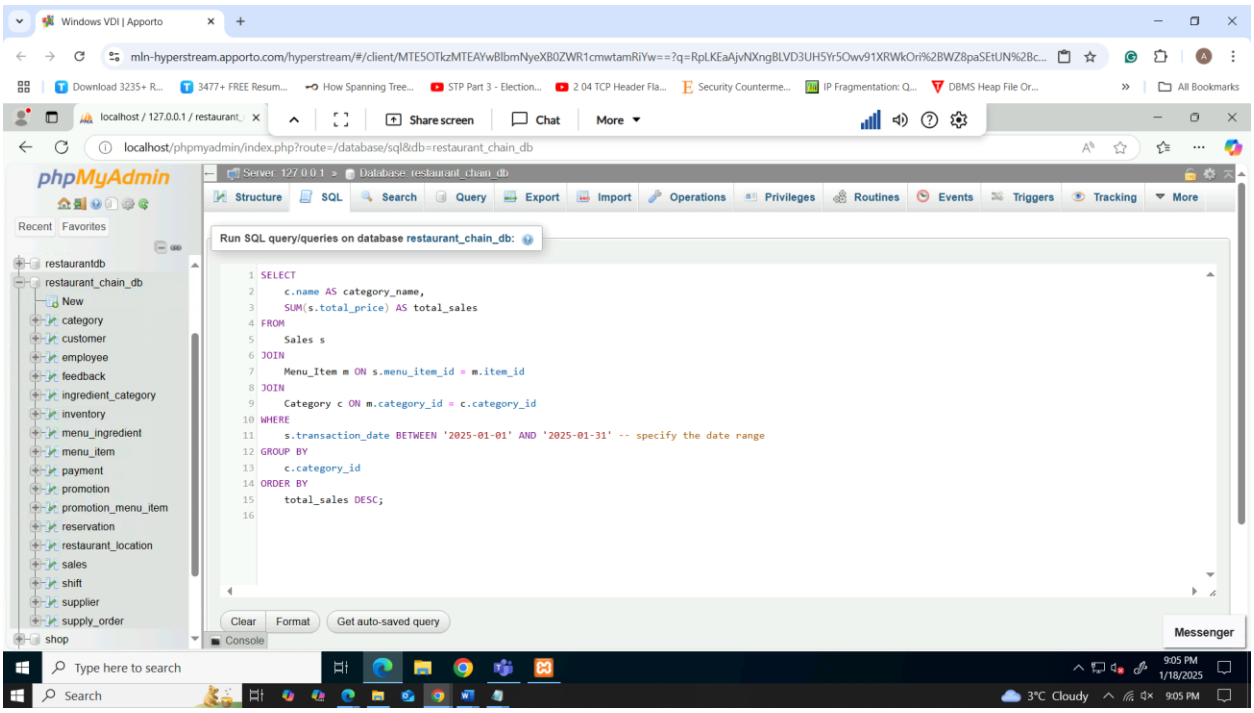


Figure 78: Total sales for each menu category

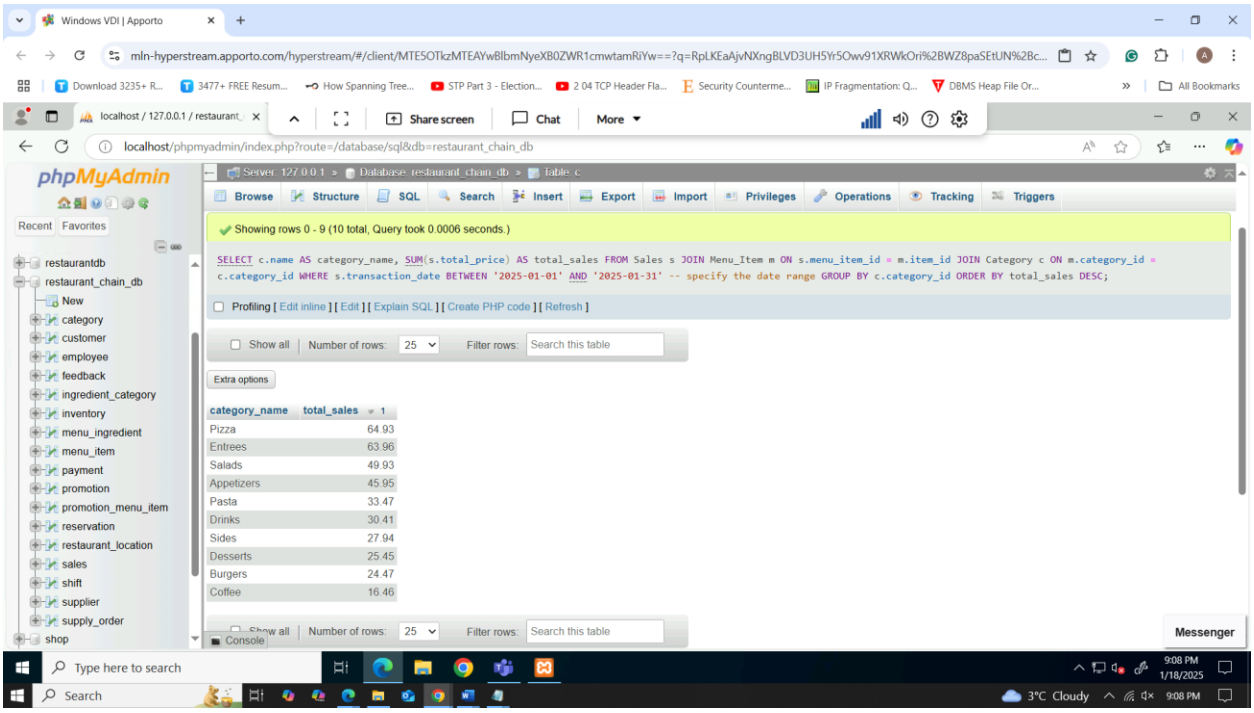


Figure 79: output of Total sales for each menu category

- Ranking of a total of 10 most ordered foods for specific days with breakdowns of name and type of food, total quantity sold

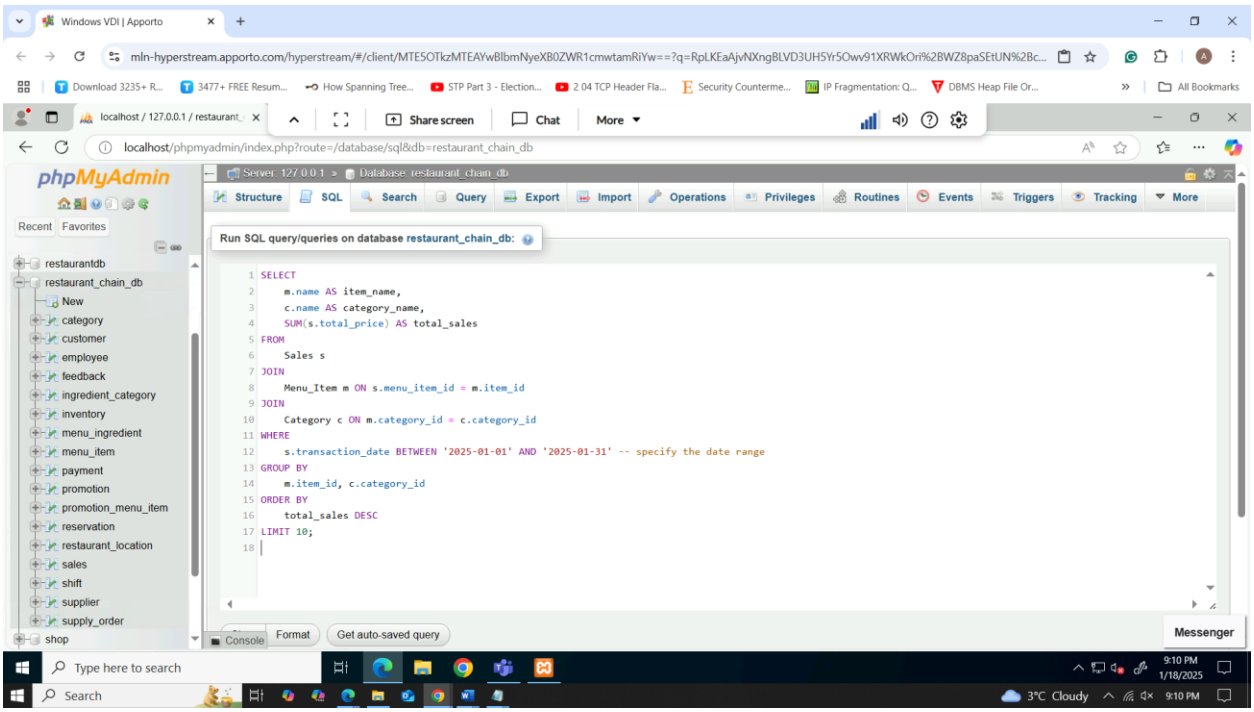


Figure 80: Top 10 most popular menu items

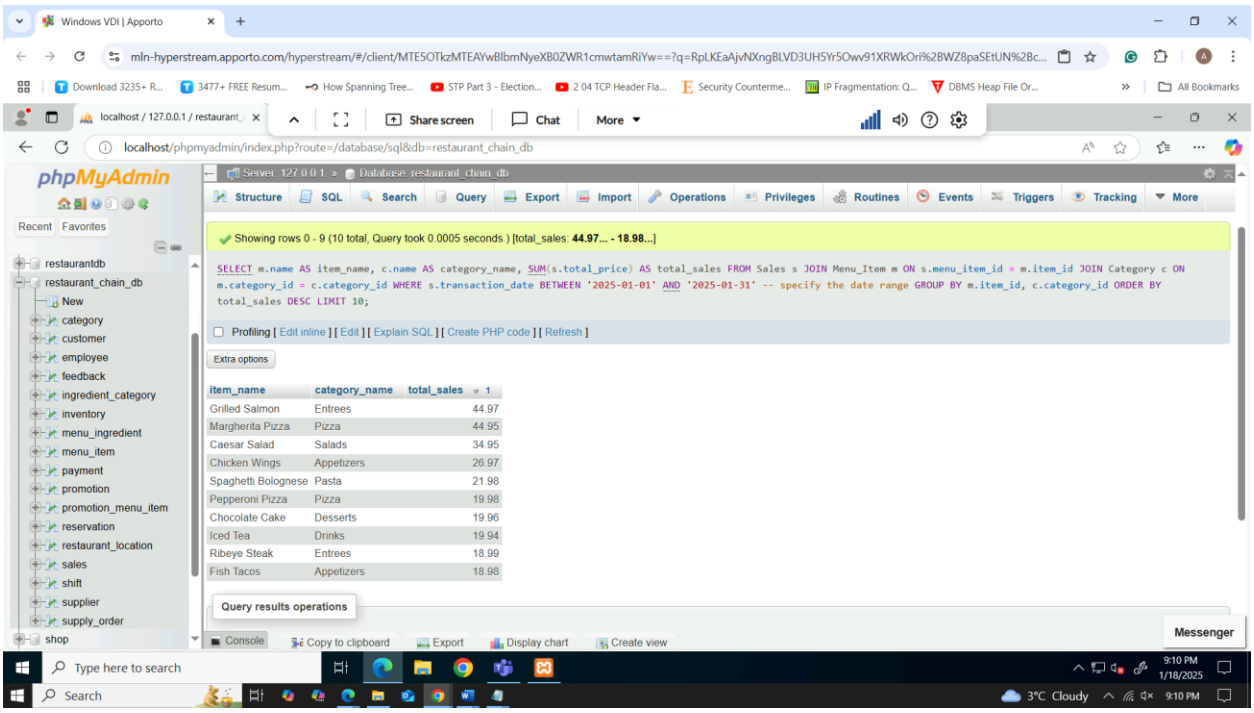


Figure 81: output of Top 10 most popular menu items

- Order information with the customer's first name and last name, phone number, number of bookings made within the last 30 days

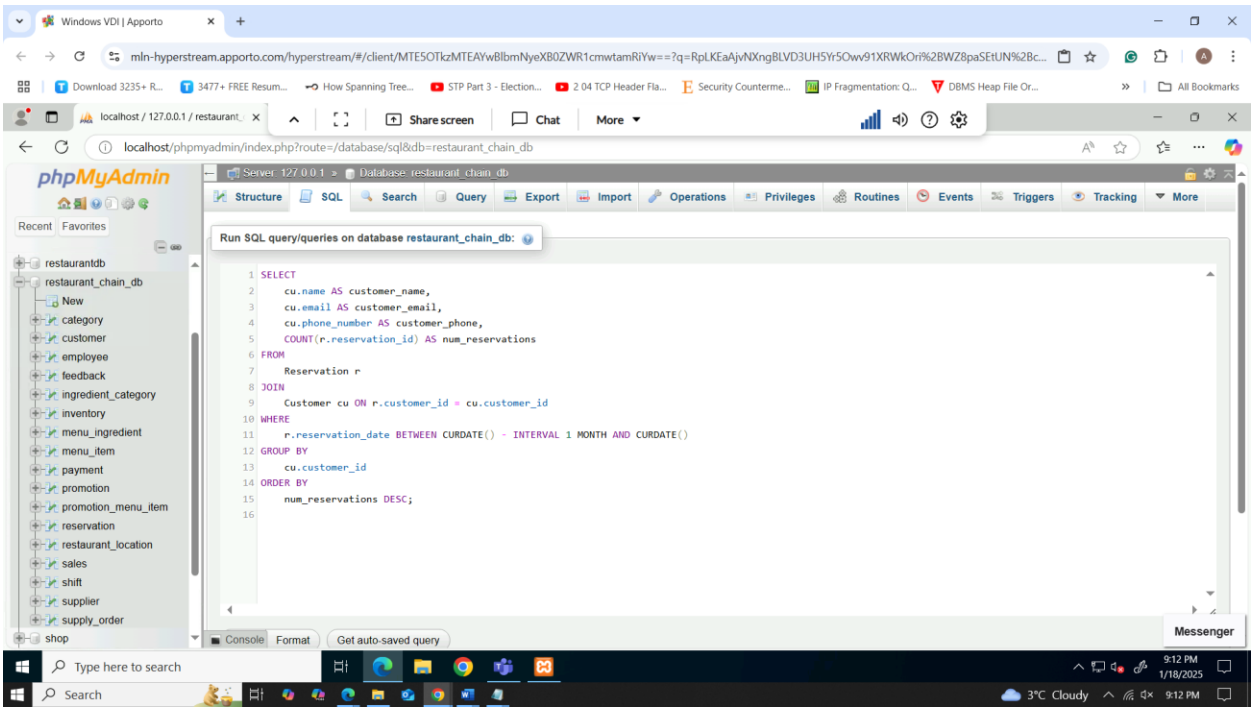


Figure 82: Customer reservation history

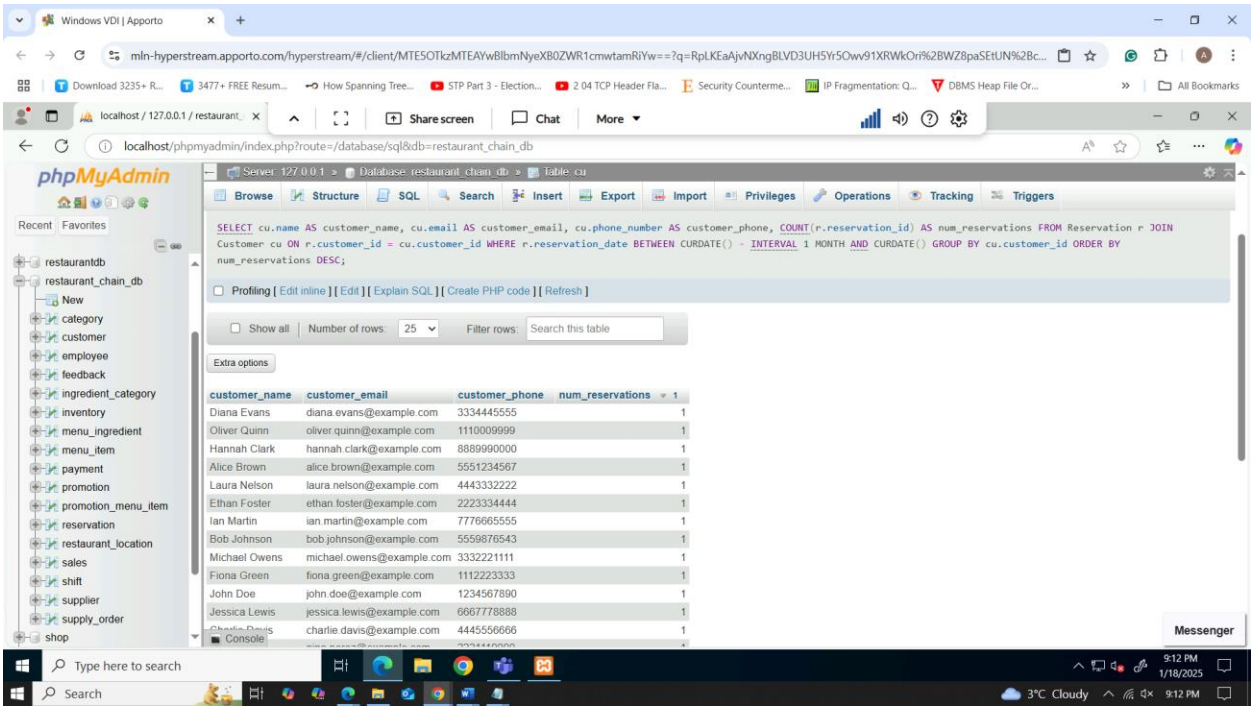


Figure 83: output of Customer reservation history

- Basic stock position of each ingredient: The ingredient name, the type of ingredient, and the amount it is currently in store.

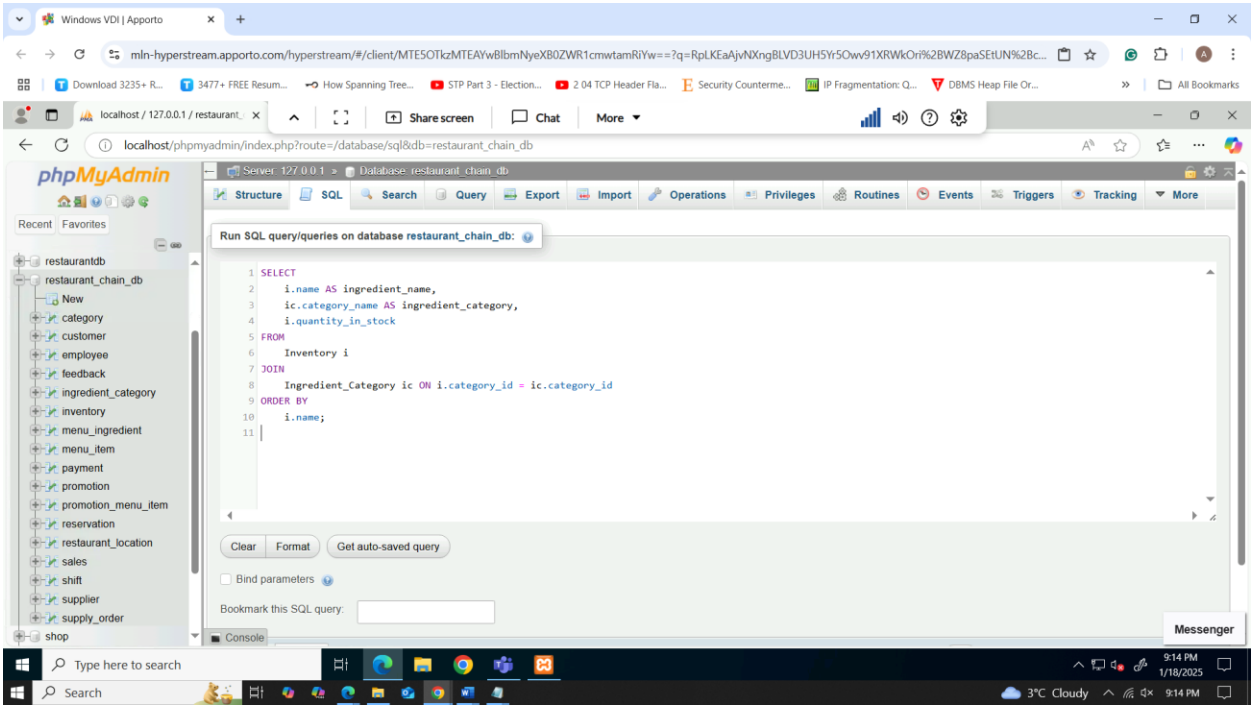


Figure 84: Current inventory levels by ingredient

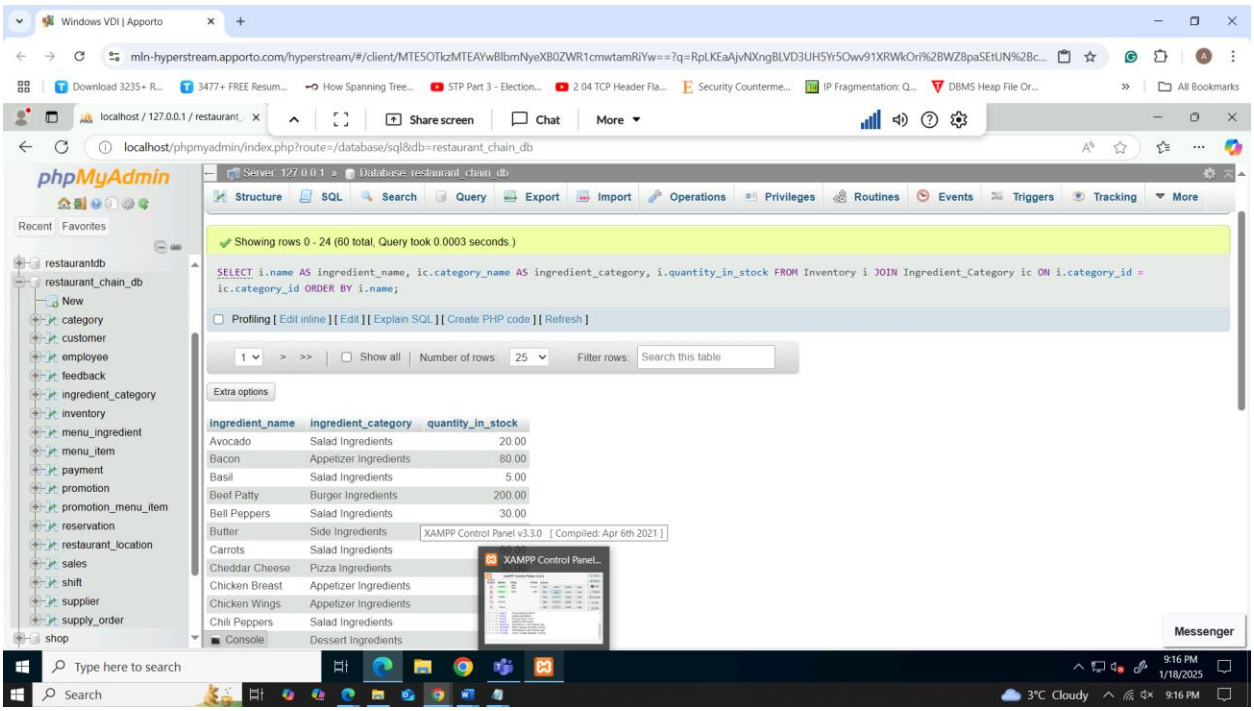


Figure 85: output of Current inventory levels by ingredient

- The total amount of money earned by each restaurant location in the past year along with the name locations

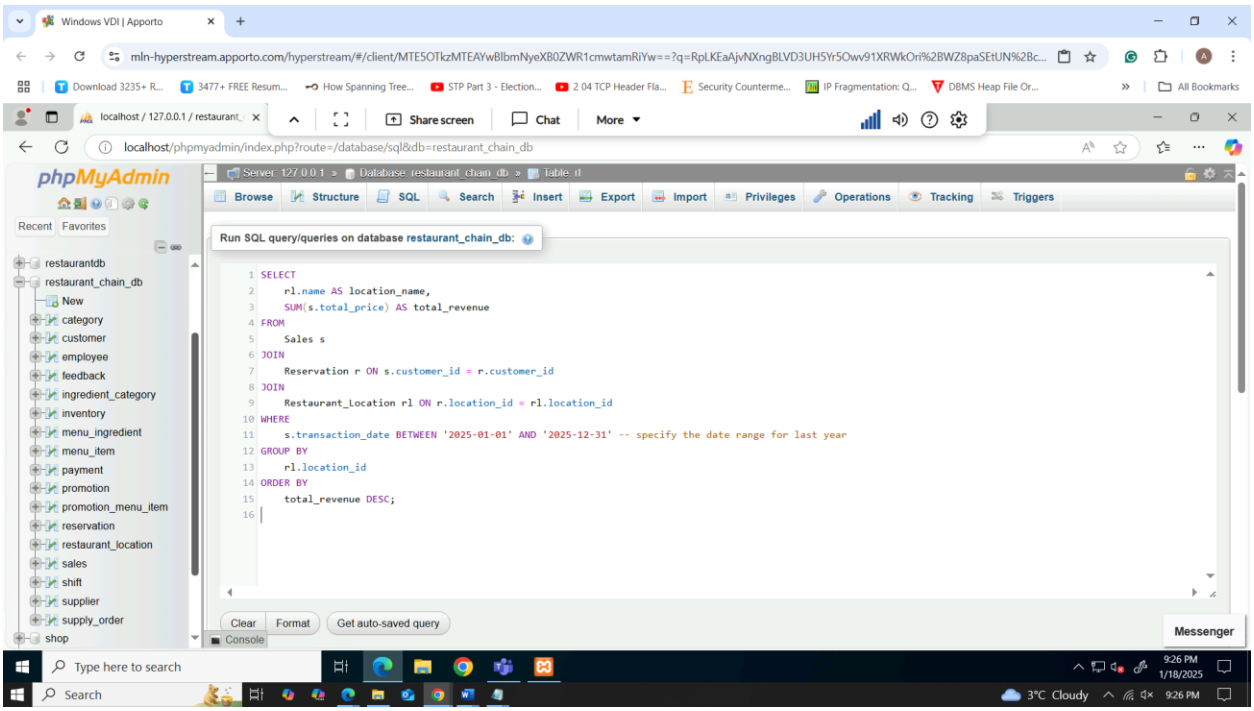


Figure 86: Revenue by restaurant location

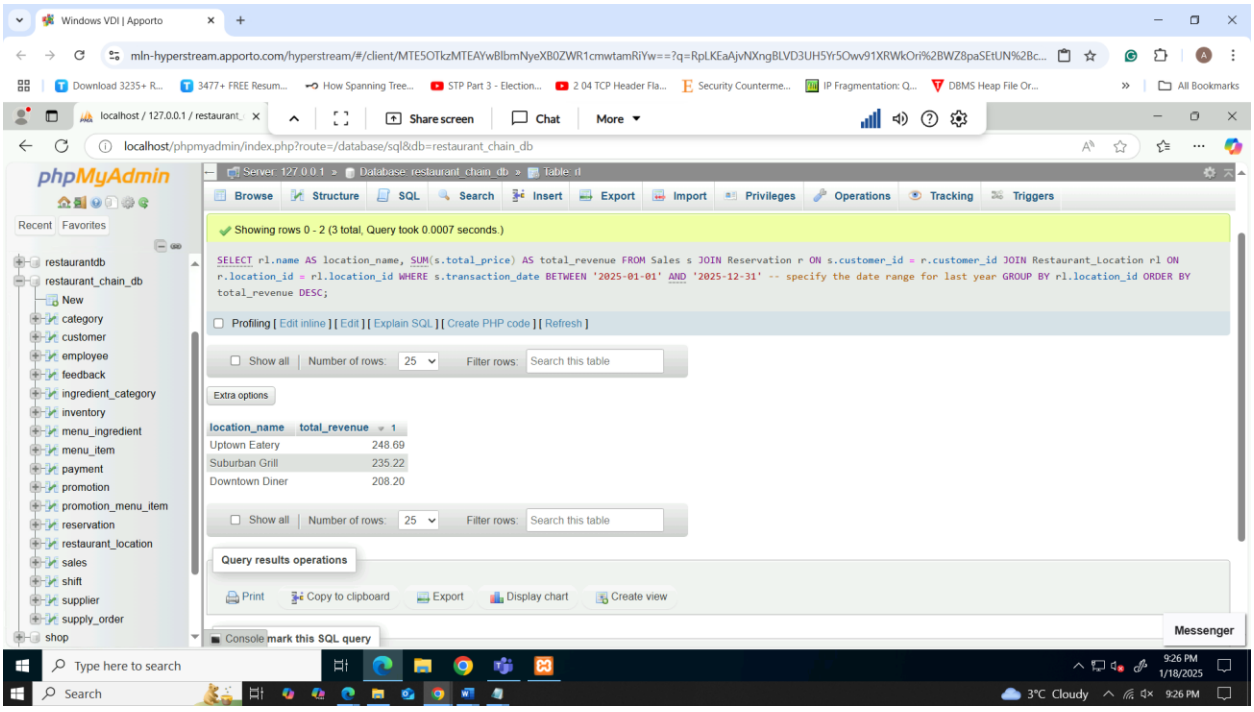


Figure 87: output of Revenue by restaurant location

6 Report and Analysis

- Daily Revenue report

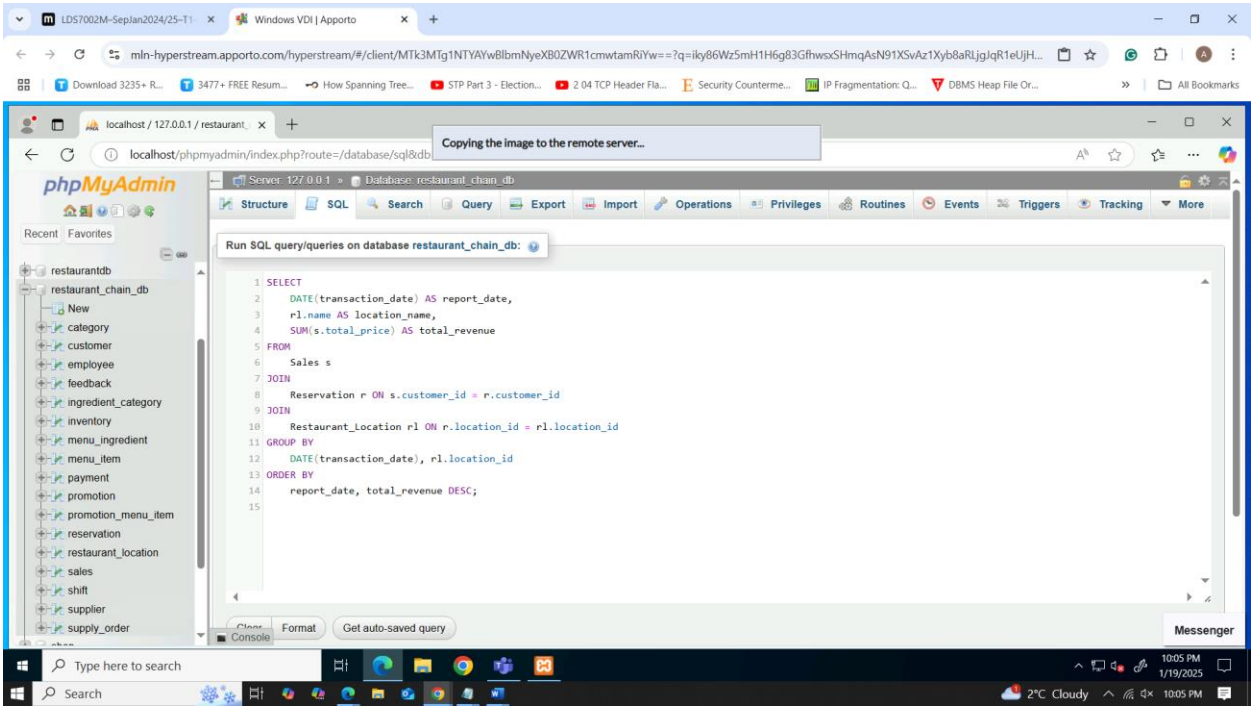


Figure 88: daily revenue report code

The SQL statement developed computes daily total revenues by restaurant location. It opts for the transaction date (transaction_date), and groups the sales, and links them with the restaurant locations through the joins. The Sales table contains records of sales and the Reservation table relates customers through customer_id to their chosen reservations. The Restaurant_Location table specifies where each of the reservations is to be made. When linking these tables it connects the sales with certain places or outlets in the restaurants. The SUM(s.total_price) accumulates a total of revenues for each location in each day. GROUP BY clause makes sure data is dumped by date and location while ORDER BY sorts the result set by date in ascending order and by decreasing revenue. This query gives a detailed, ordered, report on the daily performance of the total sales for each restaurant branch.

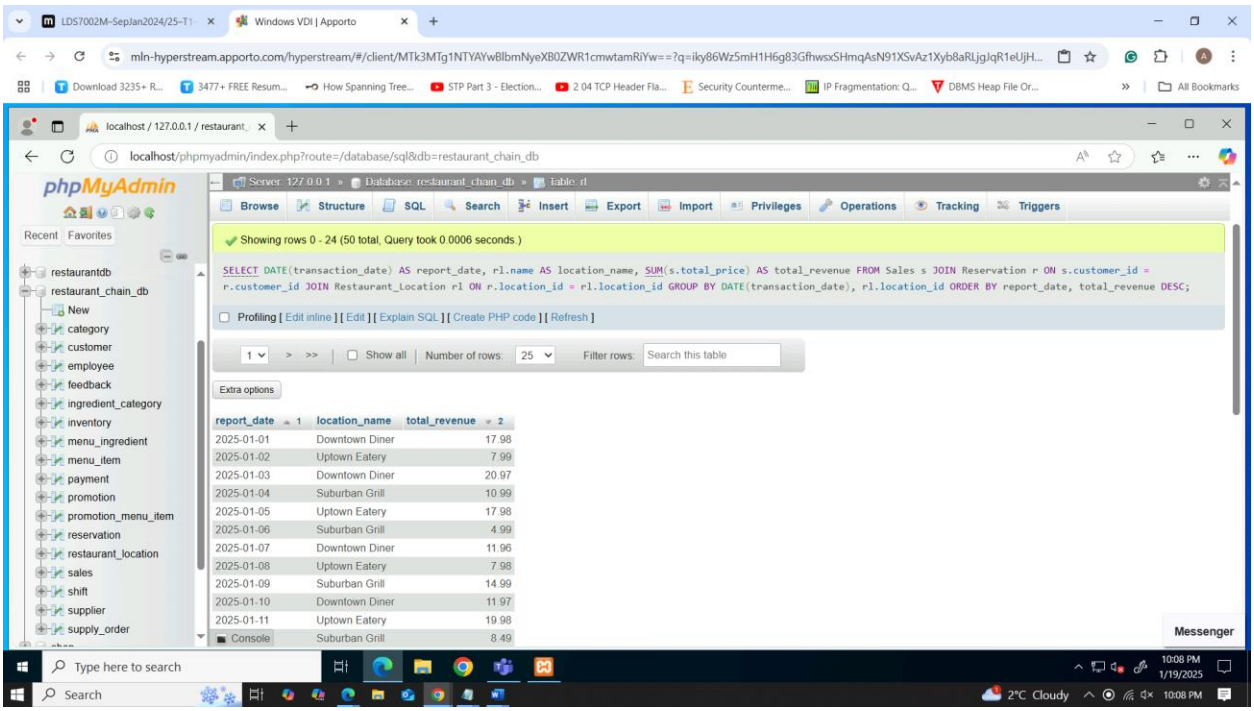
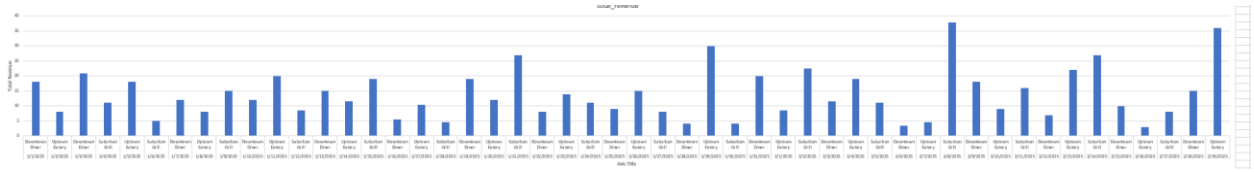


Figure 89: output of daily revenue code



The data shows daily revenue for three locations: Downtown Diner is a restaurant which implies some creation of it as a real centre of a town/square; Uptown Eatery looks like a creative and, let's say, progressive eating place in an upper-class area of a town; and finally, Suburban Grill which can be associated with a modern house in a suburban area or just a traditional diner. There are certain months when Revenues at the two restaurants vary, for instance, Suburban Grill, $ 37.98 in the 8/2/2025 and Uptown Eatery $ 35.97 in 19/2/2025. Downtown Diner steady but lower revenues are earned. It can be inferred that Uptown Eatery revenues rise towards the end of January, while the subtlety of Suburban Grill. This mean promotions could extend to Downtown Diner to encourage better performance, while at other areas, peak days could be optimally utilized. Comparing weekdays and weekends can also help to advance recognition of potential for the greatest total sales.

- Itemized Sales report by Category

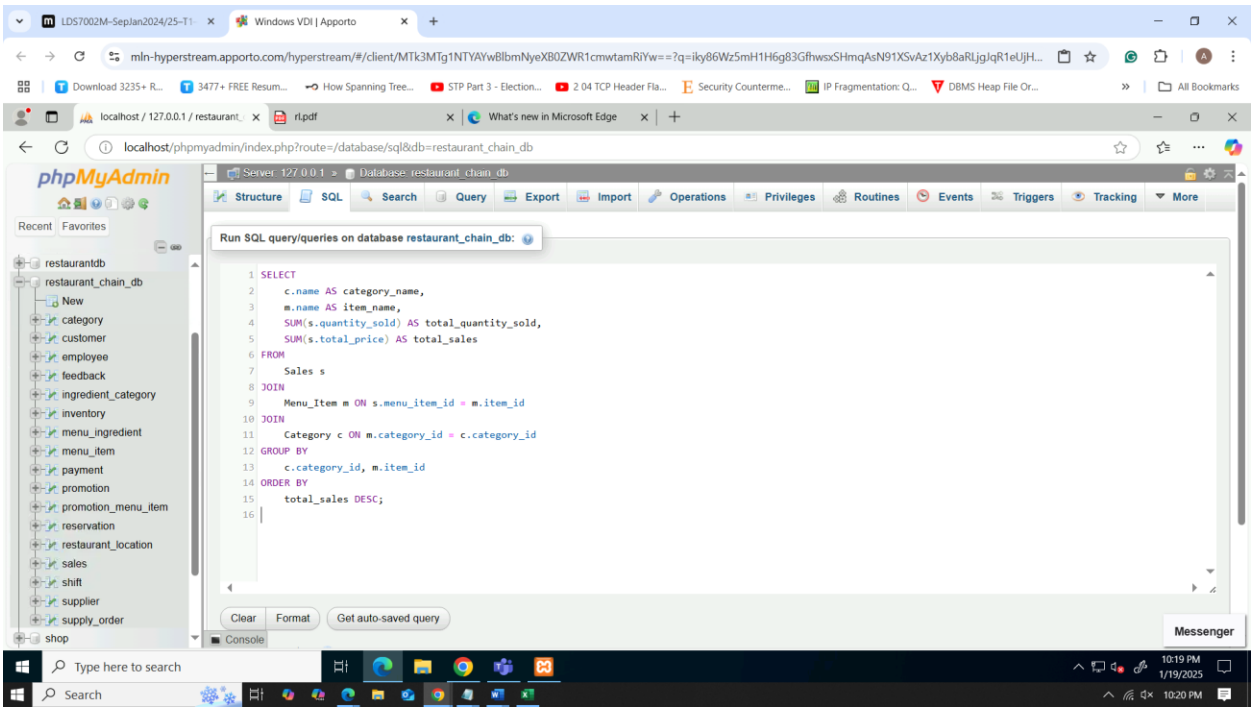


Figure 90: itemized sales query code

This SQL query creates a general report of the menu item and category sales with the emphasis on the number of items sold or total quantity and the total amount of money made from these items. The query selects the category name (c.name), item name (m.name), total quantity sold from table of status (SUM(s.quantity_sold)), and the total sales revenue from the same table (SUM(s.total_price)). It achieves this by joining three tables: Sales, that records some sales transactions, Menu_Item which associates the menu item with the sales records, and Category, to categorize the menu items. The GROUP BY clause filters data by category and particular menu item, to calculate totals, the operation has to be done for every item within the category. Plans and goods are sorted by total sales, which gives the list of items arranged in descending order and pointing at the items with the highest sales rates. This query is useful in determining market movers, category performance and planning for inventory, price changes and adding or eliminating products in the menu.

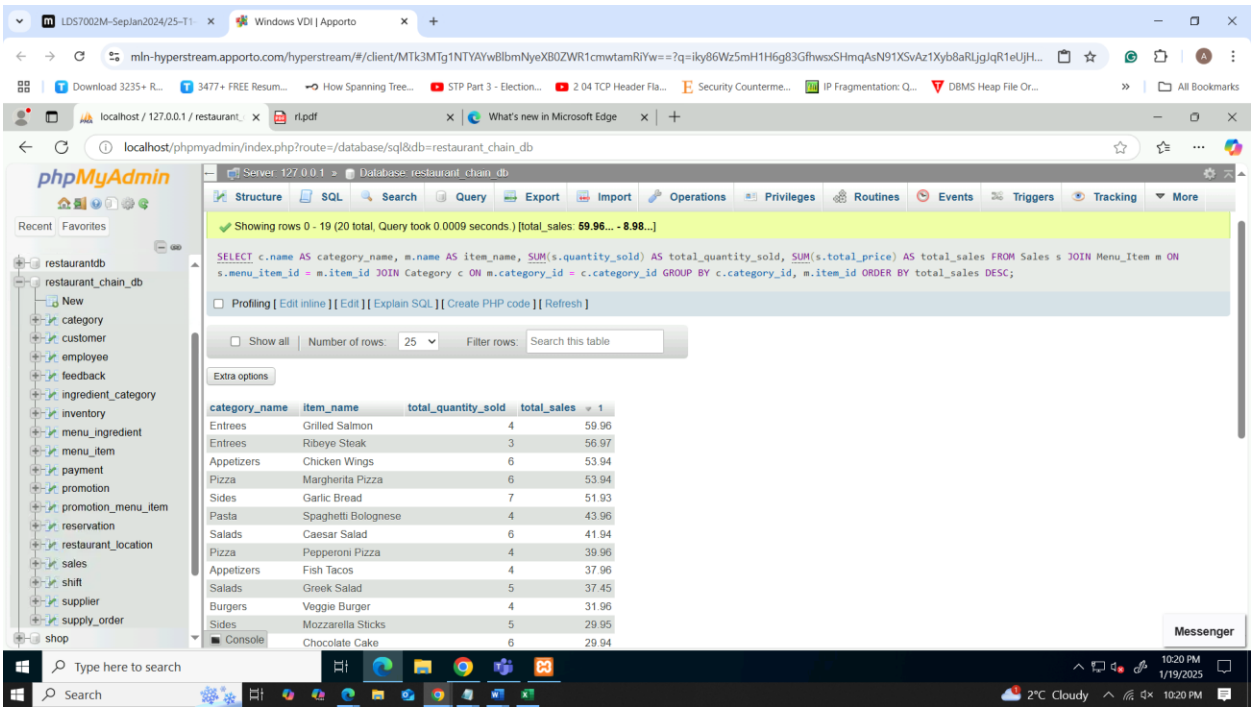
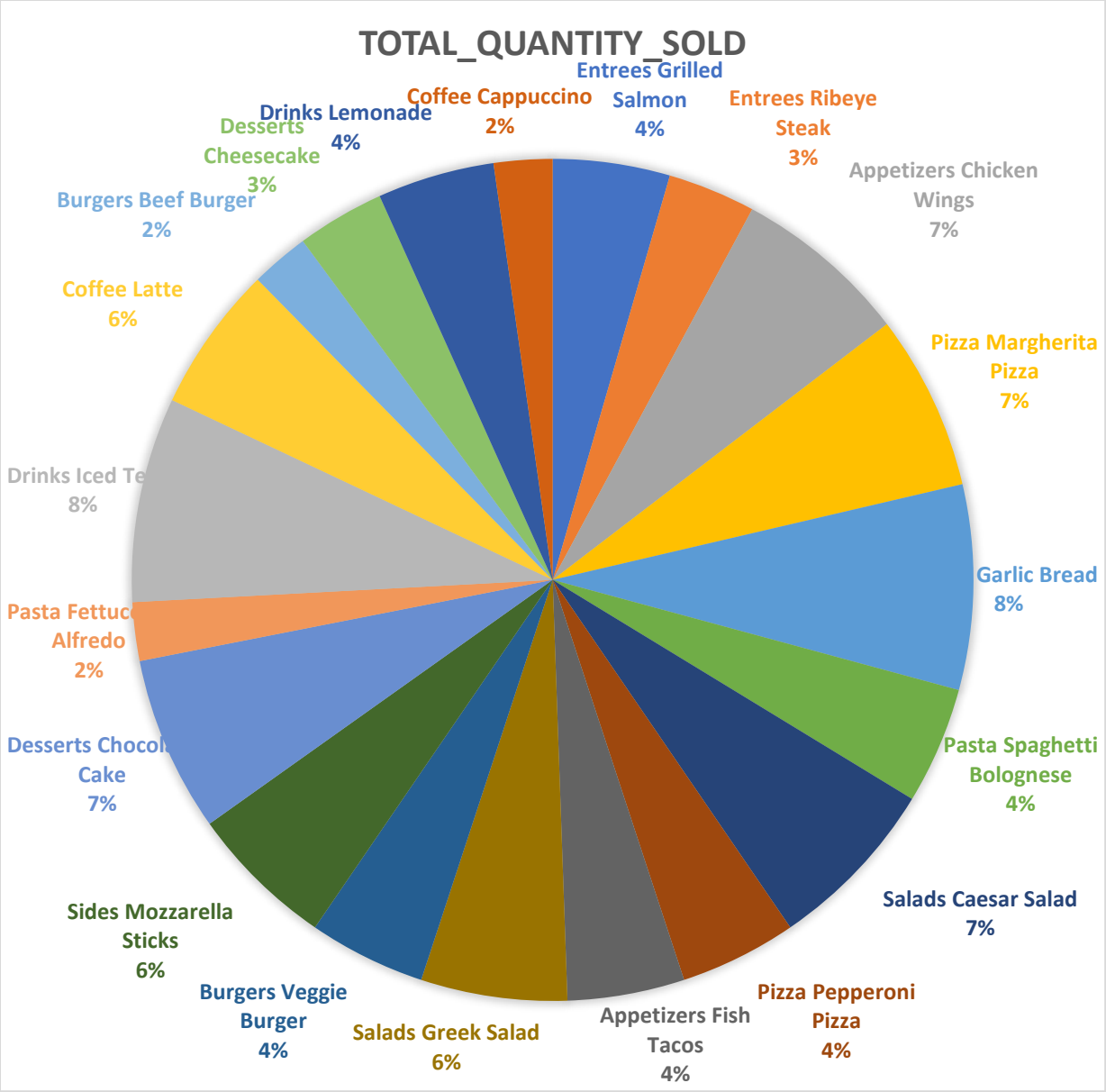


Figure 91: outcome of itemized sales



Using an intersection of sales summary by category in a pie chart displays the total contribution of each category to overall sales and show customers' favorite food items and those that are most popular. This perspective is relevant to striking the best balance of inventories in a restaurant and changing the menu with some items prioritized. Therefore, if appetizers or beverages tend to sell either higher than entrees, there could be promotions or new additions in that category boost revenues. As with poor quality, variable cost drivers may be altered: some products might be redesigned or rejected to increase gross margin. The calculation of the performance of the distinct categories enables restaurants to detect whether inventory requirements match consumption levels to offer customers preferred products, implement seasonal or promotional menus, and optimize total sales and cost.

- Inventory report

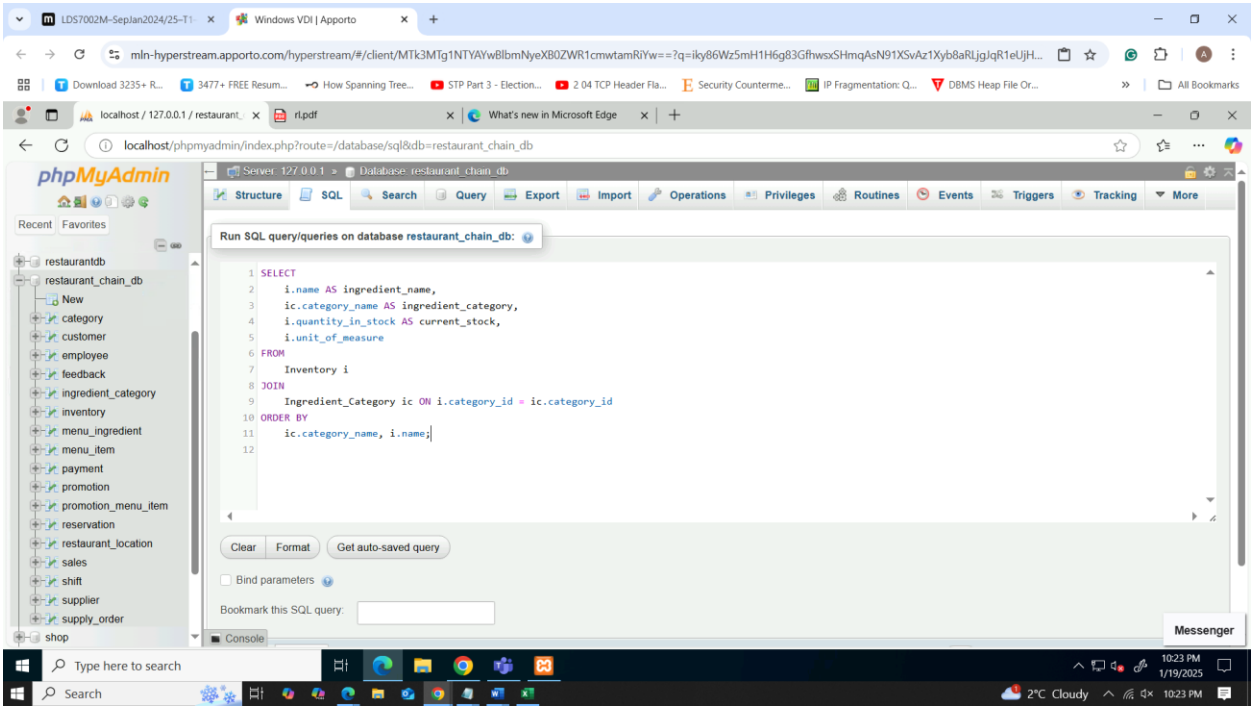


Figure 92: inventory report query code

The below SQL query gives the detailed inventory report and includes all the ingredients with their respective categories, quantity available and units. It brings back the name of the ingredient (i.name, called i for ingredient), the category to which the ingredient belongs (ic.category_name, called ic for ingredient category), the quantity of the ingredient in stock (i.quantity_in_stock), and the unit of measure of that quantity (i.unit_of_measure). The query links, through joining, between the table Inventory, which concern ingredient detail, and Ingredient_Category table which categorize the ingredient, based on the common identification number of category_id. Finally the last line orders the result set first by the name of the category in ascending alphabetical order then the ingredients inside each category. Such an arrangement makes the checking of inventories easier as far as categories are concerned and identification of certain ingredients easy. The query enables the organization to manage its inventory as it is precise and covers various aspects of the stock in detail to ensure that the businesses get an overview of the existing food ingredients that they need to order in order to restock.3.

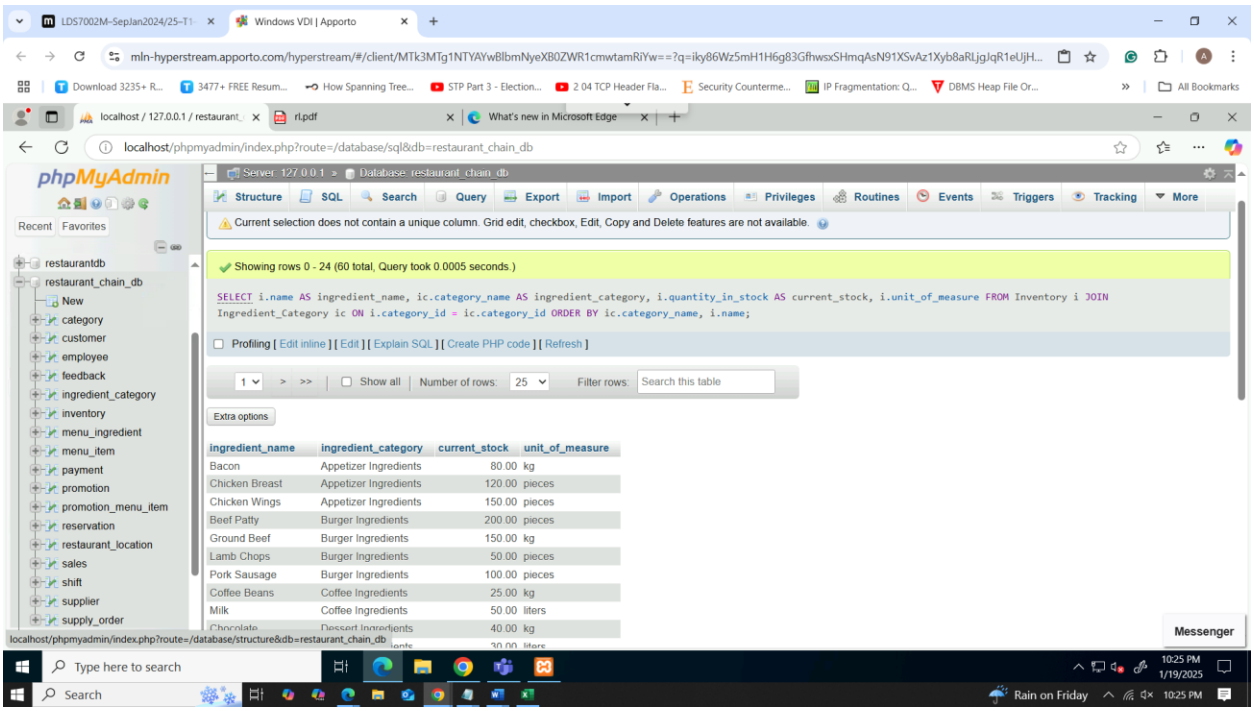
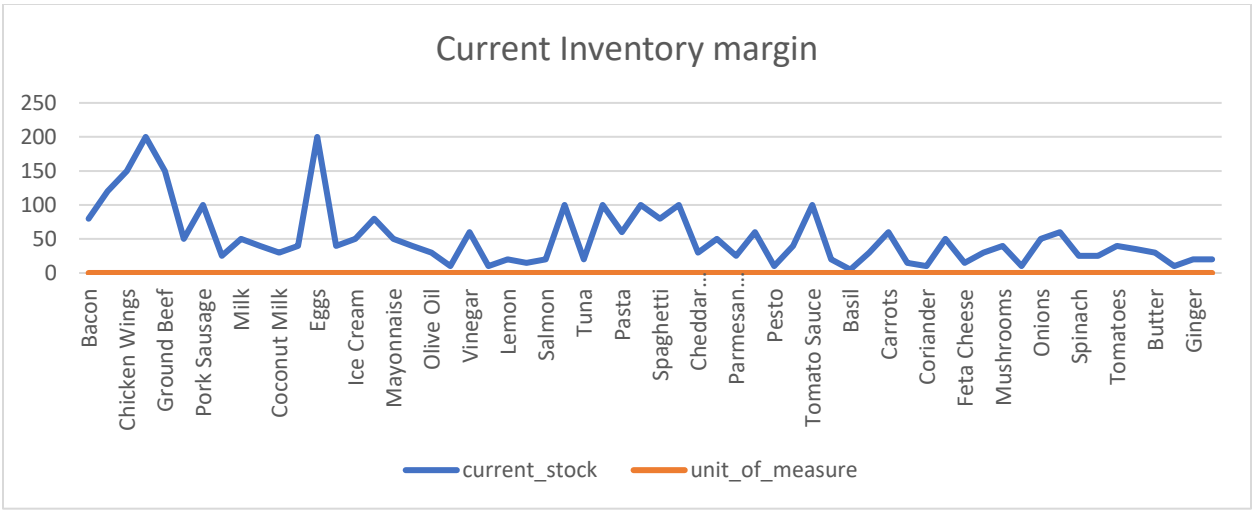


Figure 93: results of inventory margin



The inventory data provides current stock data for Appetizers, Burgers, Desserts, Drinks, and Salads, in terms of Current Stock. Products such as Chicken wings (150) and beef patty (200) show ample stock, on the other hand, sesame oil (10 litres) and worcestershire sauce (10 litres) may be stocked up loosely. This is because the degree of consumption is high in such categories as Burger & Salad Ingredients; hence require vigilance in order not to stock out. Stock levels can be presented in the form of a bar chart in order to easily pinpoint which items are critical. Subsequent tracking overtime can improve the purchasing decision since purchases will be made basing on the stock available rather than making assumptions and

end up with a surplus. The major idea of this strategy is that it helps to organize restaurants' work, controlling the constantly used products and giving proper attention to the products that are low in stock.